UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES
Section de physique

HAGRID in Space

-

High Accuracy GRB Rapid Inference with Deep learning in Space

Author:

Gilles Koziol

Department of Particle Physics

Supervisors:

Merlin Kole - David Droz

University of Geneva

A thesis submitted in fulfilment of the requirements for the Master's studies in Physics

Date of defence: 28^{th} of June, 2023

Contents

1 Acknowledgement

I would like to take this opportunity to express my appreciation to the following individuals and organizations who have played an important role in the completion of my master's thesis:
First, I want to address my gratitude to my two supervisors: Merlin Kole and David Droz for investing their time, energy and patience, spending countless hours (days) discussing with me and helping me in this work. This was highly appreciated.

More generally, I also acknowledge the POLAR group for their warm welcome and the good times I've spent with them.

Finally, I would like to thank my friends and family for their unwavering support throughout this journey. Special thanks to Maela for creating the beautiful logo, and to all those who contributed to the accomplishment of this work, which I truly enjoyed.

2 Abstract

Gamma-ray bursts (GRBs) are some of the most powerful and enigmatic events in the Universe. These brief, intense bursts of gamma radiation can last from a fraction of a second to several minutes, or even hours and are thought to be produced by some of the most violent events in the Universe, such as the collapse of massive stars or the collision of two compact objects like neutron stars.
Despite decades of research, many questions about the nature and origin of GRBs remain unanswered. One of the biggest challenges in studying these events is their rapid and unpredictable nature. GRBs can occur anywhere in the sky, and their positions are often difficult to determine accurately. These difficulties are, among other things, the reason it took decades to know that their origins were extra-galactic. Their location is also an important piece of information for other measurements, such as those which have to rapidly repoint to the GRB to measure it, or other measurements which require an incoming direction-dependent detector response.
Currently, the most widely used method for localizing GRBs uses comparisons (via a χ^2 fits) of the measured signal to a database of simulated GRBs from different incoming angles and different energy spectra. This method is fast but prone to systematic errors as the reconstructed location depends on the spectrum, therefore one needs the combination of every possible location and spectrum to perform this perfectly. There exist more advanced techniques like the Bayesian Low energy GRB Localization (BALROG) method, which uses statistical techniques to calculate the most likely position and spectrum of the source of a GRB based on the arrival times and energies of the gamma rays detected by instruments such as the Fermi-GBM or POLAR experiment. However, the BALROG method can be time-consuming compared to the χ^2 fits, and a faster method is needed to allow for rapid follow-up observations by other telescopes and instruments. In addition, despite its high reliability, the BALROG method is not suited for a space application because of the big computing power required.
In this Master's thesis, we present a machine learning-based approach for rapid localization of GRBs. Our method uses different types of neural networks to analyze the data collected by the POLAR detector modules and quickly calculate the most probable position of the source of a GRB. We have trained our models on a large dataset of simulated GRBs and achieved localization accuracy comparable to previously applied methods but with much faster processing times. In addition to rapid localization, our method also includes a spectral fitting component, which can provide important information about the energy and composition of the gamma rays emitted during a GRB.
Due to the positive results found in the localization and spectral studies, as well as the presence of a GPU on the China Space Station (CSS), it was decided to expand the results into a full GRB analysis algorithm. The goal of this algorithm is to automatically detect the GRB and subsequently calculate the location and spectrum for these. All of this will be performed using deep learning techniques. The whole algorithm was called HAGRID for *High Accuracy GRB Rapid Inference with Deep learning*. The result is of great interest due to the known rapidity of GPUs to make parallel computations such as predictions on trained deep learning models. If implemented this method is capable of providing quick alerts with accurate information.
This thesis will start with a basic introduction to GRB physics (section 3). This is followed by details on the theory part of particle detectors and particle physics in chapter 4 and 5, machine learning in 6 and random number generators in chapter 7. The simulations are described in chapter 8 and the analysis of the MC data in chapter 9 which details the localization studies and chapter 10 which describes the results of the spectral studies. After those preliminary steps, the POLAR data analysis is described in chapter 11, which details the GRB searching algorithms and 12 which handles the localization and spectral analysis. Finally, the full GRB analysis is described in section 13 this is followed by a discussion on the results and ideas for follow-up studies in chapter 14.

3 GRB Introduction

After the Limited Test Ban Treaty that was signed in August 1963 by the United States, the Soviet Union and Great Britain, the first pair of Vela satellites were launched by NASA. These were equipped with X-ray, gamma-ray and neutron sensors that are able to detect a dreaded nuclear explosion.

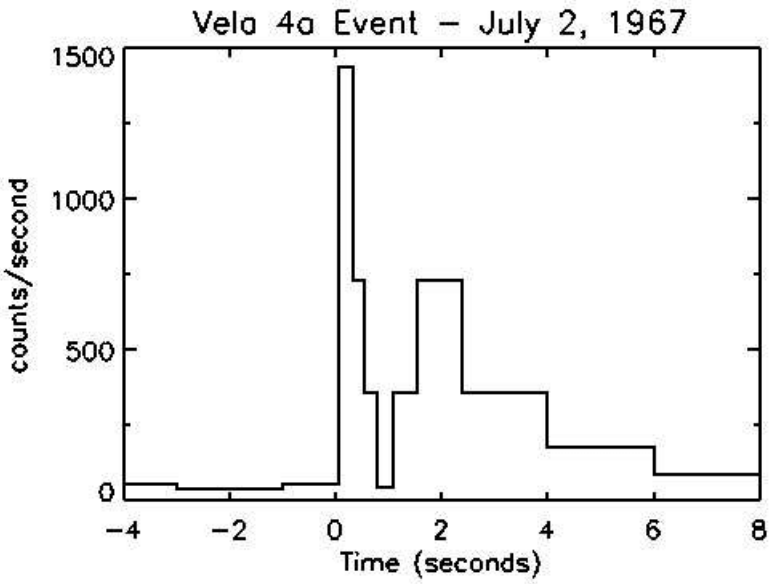


Figure 1: Historical measurement of GRB670702 detected by the the Vela-4 satellite. The double peak signal is clear.

Less than four years after the commissioning, a bright burst of γ-rays lasting less than an eighth of a second, followed by a longer peak of two seconds was simultaneously detected by two Vela-4 satellites (see figure 1). The hypothesis of a nuclear explosion was rapidly ruled out because the signature of such a weapon is expected to be very short [1]. It appeared that this burst was also detected by the Vela-3 satellites, but without the double peak shape because of the lower time resolution of Vela-3 detectors. After further research, it was also shown that no solar flare nor supernovae happened on 2^{nd} of July. This curious phenomenon was finally published in The Astrophysical Journal in 1973 [2], after which it was, due to being brightest in gamma-rays, called Gamma-Ray Burst (GRB). During the whole Vela program, the satellites were able to detect a total of 73 GRBs [33].

Almost 60 years after their discoveries, dedicated space-based observatories have been studying them and were able to detect a few thousand GRBs, helping to better understand this phenomenon. The great number of detections made it possible to produce some observational evidence, Some of the most important are:

[1] An event was recorded in 1979 by one of the Vela satellites with the clear signature of a nuclear explosion. See https://en.wikipedia.org/wiki/Vela_incident

[2] Fun fact: this paper was released exactly 50 years ago the month where this thesis is written

Isotropic distribution The *BATSE* mission, which was launched in April 1991 and took data for 6 years, was able to detect and locate 2704 GRBs [44], which makes it possible to have a good idea of their distribution in the sky.

As it can be seen in figure 2[a], it is clear that there is no preferred origin, which suggests an extra-galactic origin. If GRBs came from the milky way, their distribution would not be isotropic but would have a similar shape as the milky way. The extra-galactic origin was later confirmed by, among other instruments, the *Swift* mission that detected GRBs with a redshift up to 9.4 [44] deduced thanks to the UV measurements this instrument can perform.

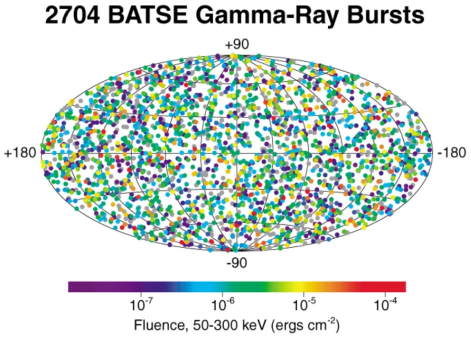


Figure 2: Map of the position of the GRBs on the sky detected by BATSE in Mollweide view. There seems to be no preferred origin, which suggests an extragalactic origin. [18]

The ability to accurately locate the GRB incoming direction is a crucial parameter to infer its position. The BATSE observatory had an uncertainty of less than a degree for the brightest GRBs but could be wrong by roughly 18 degrees in the worst cases [44]. This uncertainty is partly due to its design: it is a wide field-of-view detector. To achieve sub-degree precision, the field of view must be reduced as well as the observed wavelength. For example, *Swift* has a typical error of a few arcminutes and is sensitive to X-rays [44]. The downside of instruments with a narrower field of view is of course that they have a smaller probability to see the GRB, for this, they often rely on alerts from instruments like BATSE which can tell the instrument to repoint in the right direction.

Duration The duration of a GRB is quantified using what is called T_{90}. This parameter corresponds to the time interval during which 5 to 95% of the total number of counts occurs. The distribution of $\log(T_{90})$ is often described by a sum of two log-normal distributions.

The "short" GRBs (shorter than 2s) peak at $\sim 0.2s$, whereas the "long" peaks at $\sim 20s$ and can range up to a few thousand seconds. However, it is important to keep in mind that this distribution depends strongly on the hardness of the burst and the detector's sensitivity (which depends on the energy), as with a more sensitive detector it is easier to distinguish noise and faint signal [43]. The hardness H_{23} is defined as the ratio of the content in two given energy bins (the first one ranging from 50 to $100\,keV$ and the second one from 100 to $300\,keV$ [4]). Additionally, the spectral shape can evolve during a GRB, which means that the T_{90} you measure depends on the energy range an instrument is sensitive to. As it can be seen in figure 3, the short GRBs are often harder than the longer ones. The *Swift* multi-wavelength analysis indicates that those two different populations have different progenitors [44].

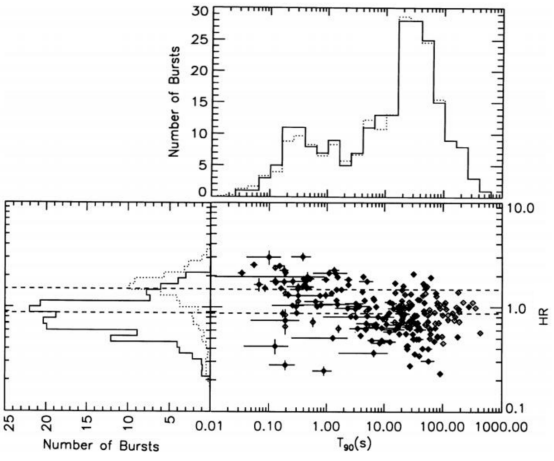


Figure 3: $2D$ scatter plot of the GRB duration vs its hardness. The dashed line in the hardness histogram is for the short GRBs whereas the solid line is for the long ones. In the T_{90} histogram, the solid line represents the raw data, and the dashed one represents the error-convolved data. [26]

[a]Image taken from:https://heasarc.gsfc.nasa.gov/docs/cgro/batse/

Band function spectrum The energy spectrum of thousands of GRBs has been studied by different groups and experiments. It appears that it is a very complex task to make a broad measurement of a GRB spectrum because it is a prompt emission, which implies that a lot of instruments have to point in the right direction while the burst happens. A GRB happens to be very bright in the $keV - MeV$ range, which makes it interesting to build instruments that have a large field of view, which makes it easier to detect the prompt emission in those energy ranges (even despite such detectors having higher background rates due to the wide field of view). Detectors observing in higher or lower (like GeV gamma-ray, X-ray, UV, optical, and radio) energy ranges need a smaller field of view, hence have a smaller probability to observe the burst. Such instruments therefore often rely on alerts from wide field-of-view gamma-ray detectors to re-point their own field of view towards the GRB in order to observe part of the prompt emission.
However, some experiments (*BATSE, Swift, Fermi, POLAR*) managed to measure GRB spectra. Such studies revealed that the spectrum is not only the result of a thermal process [29]. That means that the main process that generates photon radiation is not just black body emission but rather a process that involves the acceleration of charged particles. The reason is that a thermal process would not explain the high energy photons that are observed during the prompt emission. On the other hand, a non-thermal process is described by a power law (which decreases slower than the exponential behaviour of the thermal process) and could explain the observed high-energy photons detected.

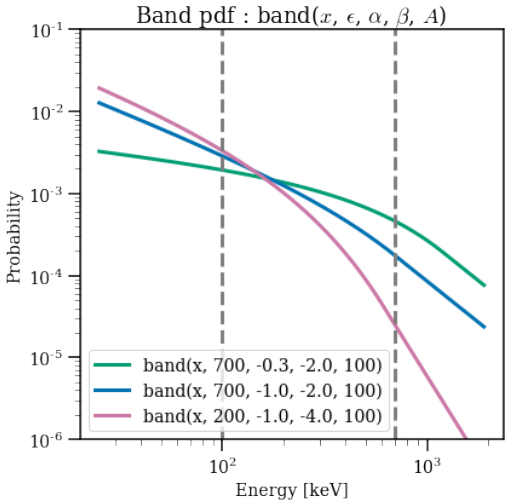


Figure 4: Three numerical applications of the band function as described in equation 1. Note that A was chosen such that the integral is 1.

However, even if only a small range of the spectrum is measured (typically between a few keV and a few tens of MeV), it appears that the most efficient way to describe a GRB spectrum is by using a continuously differentiable broken power law: the Band function [5], named after one of the major contributors of this research:

$$N_E(E) = \begin{cases} A\left(\frac{E}{100\,keV}\right)^{\alpha} \exp\left(-E/E_0\right) & (\alpha-\beta)E_0 > E \\ A\left(\frac{(\alpha-\beta)E_0}{100\,keV}\right)^{\alpha-\beta} \exp\left(\beta-\alpha\right)\left(\frac{E}{100\,keV}\right) & (\alpha-\beta)E_0 < E \end{cases} \quad (1)$$

This description is a purely phenomenological one and describes some physical models as particular cases [a]. In figure 4 are shown 3 different numerical applications of this function. Several theoretical models, such as synchrotron emission or photospheric emission, have managed to explain the Band function-like shape of the GRB emission. As several models explain it rather well it seems difficult to understand the origin of the emission using spectral information only.

[a] for example if $\alpha = \beta$, the power spectrum is recovered

Followed by an afterglow After the prompt emission, a multitude of lower energy photons (*ie.* UV, X-Rays, optical, and radio) are observed. This phenomenon is called the *afterglow*. It has been predicted even before the first afterglow measurement by [29]. The logic is that whatever the centre of the GRB is, the energy is so enormous in a small region that it has to create a *fireball* that expands into the interstellar medium at relativistic velocities. This medium is auspicious for inverse Compton and synchrotron radiation processes, which can explain the observed non-thermal emissions during the prompt emission. However, as the fireball slows down, the released energy decreases and the spectrum becomes softer, meaning that more low-energy photons are emitted [27]. The first GRB for which an afterglow was measured is GRB970228 and was detected by the Gamma-Ray Burst Monitor (GRBM) [11].

Progenitors The energy released during the prompt emission is comparable to the energy released by a sun-like star during its entire life, which makes it one of the most energetic events in the whole Universe. Their origin is thought to be the result of either the merger between two compact binaries (Neutron Star-Neutron Star or Netron Star-Black Hole), which are often associated with short GRBs, or the result of a Hypernovae which are thought to be the cause of long GRBs. One of the recent proofs of GRB progenitors is the detection of GW170817 1.7 seconds before GRB170817 [2] (*cf.* figure 5). This not only shows that the origin is a stellar process, but it also allows to put constraints on the speed of light or gravitational waves.

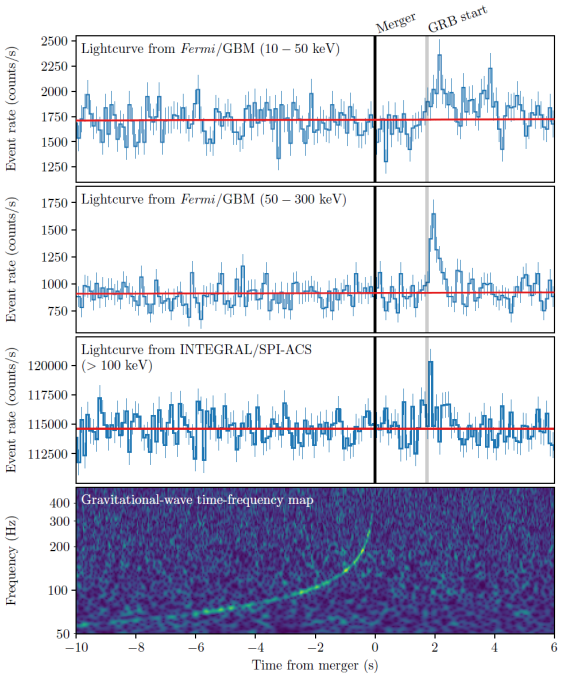


Figure 5: First joint multimessenger detection of GW170817 and GRB170817A. The GRB is detected 1.7 seconds after the merger [2]

Based on these observational pieces of evidence, a lot of models have been designed [3] To this date, the model that is most widely accepted to describe GRBs is the *fireball* model, first published by G.Cavallo and M.J.Rees [14]. The core idea of this model which was also detailed by P.Mészáros [27] is that the main engine of the GRB causes a sudden fireball that expands in the circumburst medium, producing forward and backward shocks in a region where strong magnetic fields are present[4]. Even if GRBs were discovered more than 50 years ago, a lot of open question remains open.

[3] By 1994, 118 models were trying to explain GRBs [44]

[4] In this work, we won't detail this model for the simple reason that it could be an entire work for itself.

3.1 GRB open questions

There are still a lot of things that are not fully understood about GRBs. In the following we review a few open questions from 2011 explained in [43] that remain open to this date.

What is the central engine? It should be noted that different types of processes can lead to the same central engine (the unknown object that causes the particles to be accelerated). It however needs to emit a luminosity similar to the observed GRB luminosity ($L_{iso} \sim 10^{47} - 10^{54}\, erg/s$). The most discussed type are the following ones:

- Spinning black holes are good candidates for the central engine of GRBs. However, to reproduce the observed GRB luminosity, the accretion rate should be in the range $0.1 - 1\, M_\odot/s$. The second constraint is on the spin, because the faster the black hole rotates, the higher the magnetic fields are, which can more efficiently accelerate particles. As a result, if a black hole is the main GRB engine, it should be rapidly spinning and its accretion rate should be high.
- A millisecond pulsar has a power that depends on its magnetic field, its radius, and its angular velocity: $L = B^2\Omega^4 R^6/(6c^3)$. In order to reach a luminosity close to the observed one, such an object should have a surface magnetic field of $10^{15}G$ and an angular frequency Ω of $10^4\, Hz$. The accretion power in such a system is also not to be forgotten. However, a high accretion rate would quickly turn the neutron star into a black hole (a few seconds for a rate of $\sim 1 M_\odot/s$). In the direct environment of the neutron star being filled with matter, the matter outflow could however be produced a few seconds after the shock.
- Other fancy central engines have been discussed like strange stars (even though they have never been observed) that could satisfy some conditions that are hard to satisfy for traditional stars.

What is the radiation mechanism responsible for the prompt emission? While the afterglow power spectrum is relatively well-modelled, the GRB prompt emission radiation mechanism is still not fully understood. The most probable mechanism is synchrotron radiation. Indeed, since the neighbourhood of the main engine is very probably highly magnetized, it bends the charged particle's trajectory (mostly electrons), which causes synchrotron radiation. However, for such a model, the α parameter of the Band power spectrum is expected to be -1, where the observations tend to show a tendency of $\alpha \sim -1.5$ [43]. Different modifications to the models have been made such as introducing a rapidly decaying magnetic field, but it also reveals other inconsistencies. Due to the initial difficulties in explaining the observed spectra, alternative models such as *photospheric emission* have been proposed. In these models, the emission observed on Earth is a result of thermal emission which is comptonized inside of the GRB jets. Overall, the emission mechanism is a very complicated process to model due to the combination of intense and turbulent magnetic fields interacting with highly energetic particles.

3.2 POLAR/POLAR-2 missions

Depending on the radiation mechanism and the distribution of the magnetic fields around the central engine, the polarization differs and is thought to be a powerful method to constrain the different emission models [45]. This is exactly what the POLAR detector was designed for. POLAR was a Compton polarimeter that was launched in 2016 on the Tiangong-2 Chinese space lab. After roughly 6 months, the high voltage power supply died and lead to the end of the mission. The Spacelab was de-orbited in 2019. The main goal of the POLAR detector was to analyse the polarization of the GRB prompt emission. The design of the detector is quite simple since it is a $40 \times 40 = 1600$ scintillator bar of dimensions $4 \times 4 \times 200\, mm^3$ array, grouped in 25 modules. The high energy photons interact with the scintillator bars through Compton scattering (the details on the physics of this process will be described in chapter 5). Depending on the number of interactions a scattering angle can be inferred and thus determine the polarization angle (PA) and polarization degree (PD).

Even though the mission didn't last very long, POLAR still managed to detect 55 GRBs, out of which 14 were bright enough to perform a polarization analysis [25]. As it can be seen in figure 6, the polarization degree is rather low and the POLAR collaboration claimed that the null hypothesis (*ie.* the beam is unpolarized) can not be rejected.

It is, however, important to understand that any polarization analysis can not be performed without knowledge of the GRB location in the sky because a detector response (as explained in chapter 4) is needed to compare observations to models. POLAR used to rely on other instruments' GRB localization. If these are poor this can lead to rather big errors, as the localization error propagates to the polarization. In some rare cases, POLAR was the only one to detect a GRB, in which case it has to rely on its own localization method.

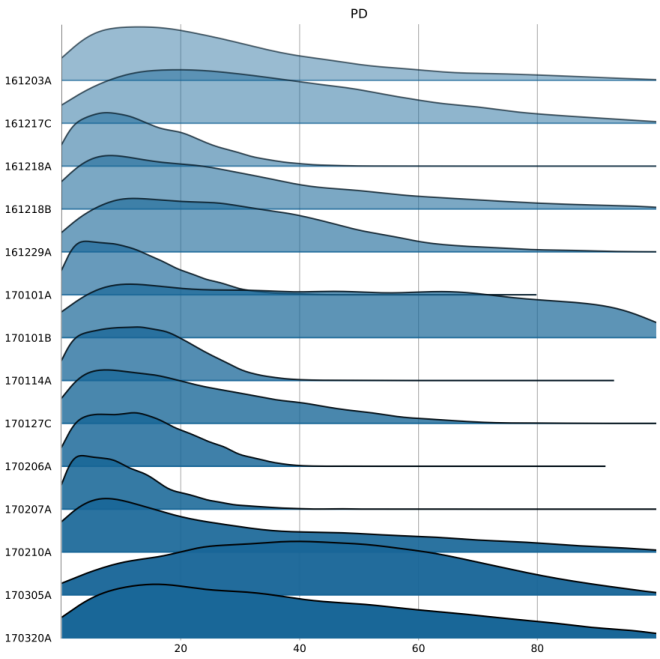


Figure 6: Posterior distribution of the polarization degree for the 14 GRBs that were bright enough to perform a detailed polarization analysis. [25]

The method that was used for POLAR is the following: Depending on the incoming beam direction and the spectral hardness, the trigger rate in each one of the 1600 bars changes. Note that there could be different situations leading to the same result, for instance, a soft GRB with a low θ value (see figure 18) would lead to the same trigger rate distribution as a harder GRB with a higher θ value because more energetic photons have a longer mean free path in the plastic that composes the scintillator bars.

The hitmap (the number of triggers per bar) is then compared through a χ^2 fit to a lot of hitmaps produced through simulations of GRBs coming from different directions and having three different spectra ("soft", "medium" and "hard")[41]. Once the minimal χ^2 value is found, it is possible to provide a location. Using a discrete set of spectra implies that unless a GRB has the same spectrum as the simulated one (which is never the case), the simulated hitmap will never be the one corresponding to the real GRB. This will thus induce systematic errors in both the spectral and localization analysis. A more precise method employed, for example by Fermi-GBM, is to follow this initial location measurement with a spectral measurement (which uses the best location fit found), this is then followed by a second localization study where the

simulated hitmaps are produced using spectra very similar to the best-fit result found in the spectral fit. This longer iterative process results in better location fit, however, it is still prone to systematic errors. This is because the initial localization relies on a poor spectral guess, resulting in a systematic error, which propagates to the subsequent spectral fit and again to the final localization. There are also other ways to proceed like the `BALROG` method [10] which is a joint Bayesian localisation and spectral analysis. This analysis fits the location and the spectrum at the same time, which eliminates the systematic errors of the χ^2 method (because it only compares the detector response to a simulation database). It also properly includes the uncertainties of the location in the spectral parameter, which makes it a more reliable localization method than the χ^2 method. It is to note that while the χ^2 method is faster, both methods are time-consuming and are relatively highly demanding tasks computationally speaking. It was not an issue for POLAR, because the data needed to be downloaded on Earth before being analyzed, the time this takes is significantly longer than the computation time.
This will, however, be an issue for POLAR-2 [24] that is planned to be launched in 2025 to the China Space Station (CSS)[5]. POLAR-2 is the next-generation GRB polarimeter that is supposed to be much more sensitive than POLAR (especially at low energies). Part of this improvement is due to the greater number of scintillator bars (6400 instead of 1600) and the use of silicon photomultipliers (SiPMs) that are more sensitive than the traditional PMTs, and they also don't require a high voltage source to work.
On the CSS, there is a computer with a GPU, which will not only allow performing real-time analysis but will also make it possible to alert other observatories, so that they can direct their instrument to the correct location in order to observe the GRB/afterglow. This can provide crucial information on the GRB mechanism, which makes a fast and reliable alert system of great interest. In order to do so, a fast localisation method is needed, and the GPU onboard makes it possible to run deep learning models, which can give predictions on data within a few seconds.

[5]There seem to be some issues so a delay is not to be excluded

4 γ-ray detectors

Since the whole thesis is on studying γ-ray, we have started by briefly explaining how gamma rays are detected and measured. All types of detectors work on the same principle :

"The transfer of part or all of the radiation energy to the detector mass where it is converted into some other form more accessible to human perception" [42, Chap. 5].

Unlike charged particles and as explained in chapter 5, γ-rays need to interact with matter to produce charged particles or excitation, which can be detected. The sensitive range of a γ-ray detector will have a direct impact on its design and material as the probability for a photon to interact highly depends on the atomic number Z (the number of electrons per atom). In a detector like POLAR, the goal is not only to detect photons, but they also need to undergo Compton scattering (POLAR is a *Compton* polarimeter), which means that the material that makes the POLAR detector should have a Z value high enough to maximize the probability for a photon to interact in the detector, but Z should also be low enough so that the interacting photons have a relatively high chance of being Compton scattered more at least once to perform a polarization analysis. This is the main reason why POLAR is made out of Vinyltoluene.

Even though a detector is meant to count the interactions, it is not as simple as that, because the reaction of the detector depends on a lot of parameters, including its shape, material, and the energies of the incoming particles. As a result, the counts per pulse height bin $c_\Omega(h)$ that the detector measures are derived from many underlying quantities and also depend on the number of sources [12] (by calling the wavelength λ):

$$c_\Omega(h) = \tau_{eff} \int_0^\infty \mathrm{d}\lambda D_R(h,\lambda) \left[\sum_i A_\Omega(\lambda, \hat{q}) s_i(\lambda) \right] \tag{2}$$

where τ_{eff} is the effective exposure time, $D_R(h,\, \lambda)$ is the function that encodes detector response per energy bin as a function of the input energy (also known as the RMF), $A_\Omega(\lambda, \hat{q})$ is the detector response (often called ARF) which includes things like the detector filters, the effective area and the quantum efficiency (the percentage of detected photons). This quantity depends on the source location $\hat{q}$. The term $s_i(\lambda)$ is the spectrum of the i^{th} source.

The RMF and the ARF can be obtained by detailed simulations which have to be verified with calibrations of the detector. As a result, it is possible, given a count per energy bin h, to reconstruct the detected spectral properties of the photons.
In the POLAR experiment, the ARF and RMF depend on the location, which means that for each simulated incoming angle there are different ARF and RMFs (one for each simulated energy and polarization). In order to perform a polarization measurement, the idea is to, given the localization invert equation 2 to have access to the polarization properties $s(\lambda)$ of the source. Since the localization has some uncertainties, this error propagates to the polarization measurement. The same is true when one wants to perform spectral measurements.

5 Particle physics theory

As mentioned in 2, a toy MC detector was built to both better understand the relevant interaction of photons in a detector, to gain an intuition for the following analysis, and to predict what results are to be expected from the real data which has a lot more unknowns. It is therefore important to know how the particles will interact in the detector and how they are measured in order to be able to simulate them. In this chapter, we will first introduce the mass attenuation coefficient, which explains the fact a flux of γ-rays is partially absorbed when going through a material. We will then detail its different components.

5.0.1 Mass attenuation coefficients

Let us take the ideal situation where the γ-ray source has a known intensity I_0 and is directed towards a material (the detector) of density ρ. The proportion of the beam that can go through a thickness t can be expressed as an exponential decay that depends on the absorption coefficient μ and the physical distance travelled t: $I(t) = I_0 \exp(-\mu t)$. In this context, μ can be seen as the inverse mean free path travelled by the particle. This equation is however often presented differently by introducing the *mass thickness* : $x = \rho \cdot t$ (in g/cm^2) and the mass *attenuation coefficient* : μ/ρ (in cm^2/g). As a result, the exponential decay can be rewritten as:

$$I/I_0 = \exp(-\mu/\rho \cdot x) \tag{3}$$

The coefficient μ can be derived from direct measurements but can also be calculated because it is simply proportional to both the material density and the total cross-section.
The cross-section gives information on the probability two particles interact. It can be calculated by using the Feynman diagrams and the laws of the standard model. There also exist a quantity called the *differential cross section*, which is formally defined as the probability of having N_s particle scattered in the infinitesimal solid angle $\mathrm{d}\Omega$, normalized by the particle flux F [42]:

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega} = \frac{1}{F}\frac{\mathrm{d}N_s}{\mathrm{d}\Omega} \tag{4}$$

A scheme of the representation of the differential cross-section concept is shown in figure 7. If we call A the molecular weight and σ the total cross-section, we have (note that μ depends on the energy):

$$\mu = \mu(E) = N\sigma = \sigma\frac{N_a\rho}{A} \tag{5}$$

The total cross-section is then the only part to evaluate. For X-rays and γ-rays, the three interactions are Compton scattering, photoelectric effect, and pair production.

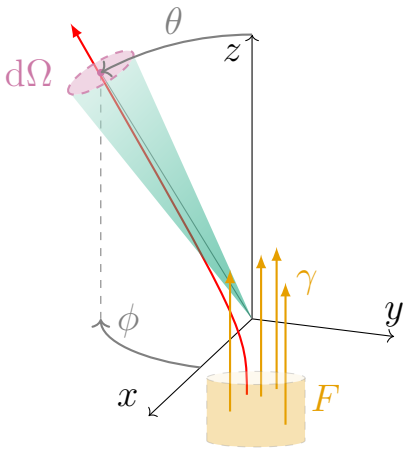


Figure 7: Scheme representing the concept of the differential cross-section. The interacting γ-ray is diffused in the $\mathrm{d}\Omega$ area element.

Compton scattering is the process in which a photon scatters on an electron (see figure 8) ($\gamma e^- \Longrightarrow \gamma e^-$).

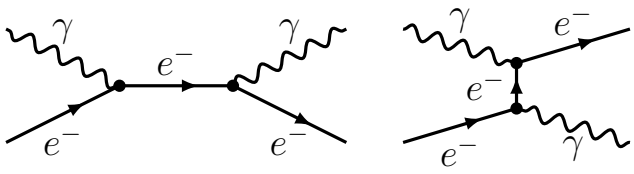


Figure 8: Two first-order Feynman diagrams of the Compton process.

After this interaction, the direction and the energy of both the photon and the electron are changed. If we call E the photon energy, θ the photon scattering angle, and γ the ratio between the photon energy and the electron rest mass energy $E/m_e c^2$, we can write the energy of the outgoing photon as

$$E' = \frac{E}{1 + \gamma(1 - \cos(\theta))} \tag{6}$$

The scattering angles are, however, not uniformly distributed, but follow the *Klein-Ninshina* [22] differential cross section. There are two interesting photon beam cases: the polarized case and the unpolarized one. Each individual photon in the flux is polarized since it is an intrinsic property of the photon [21], but a fraction of the flux can have the same polarization angle (PA), which makes the whole flux polarized to some degree, hence the name *Polarization Degree* (PD). The unpolarized case is reached when there is no preferred PA. The Klein-Nishina differential cross-section reads :

$$\frac{\mathrm{d}\sigma_{KN}}{\mathrm{d}\Omega} = \frac{r_e^2}{2}\left(\frac{E'}{E}\right)^2\left[\frac{E'}{E} + \frac{E}{E'} - 2\sin^2(\theta)\cos^2(\phi)\right] \tag{7}$$

where θ and ϕ are respectively the polar and azimuthal scattering angle, and r_e the classical electron radius. It is clear that the distribution is not isotropic both in the θ and ϕ angles. Moreover, it is to be noted that since the differential cross section depends on $\cos^2(\phi)$, the flux has a higher probability of getting scattered in a direction that is perpendicular to the polarization of the incoming one.

If the incoming photon flux is unpolarized, the cross-section can be obtained by taking the average over the ϕ dependant term: $\langle\cos^2(\phi)\rangle = 1/2$, leading to the following expression :

$$\left.\frac{\mathrm{d}\sigma_{KN}}{\mathrm{d}\Omega}\right|_{unpol} = \frac{r_e^2}{2}\left(\frac{E'}{E}\right)^2\left[\frac{E'}{E} + \frac{E}{E'} - \sin^2(\theta)\right] \tag{8}$$

As a result, when an unpolarized photon flux interacts via Compton scattering with the material, the photons can be scattered at an angle θ (given by the Klein-Nishina polarized cross-section), and their energy is modified according to equation 6. The remaining energy is the *deposited energy*. Since we won't do any polarization analysis in this work, we will be interested in the unpolarized case. There is a special case that particle physicists are often interested in, which is the case where the photon is backscattered (*ie.* $\theta = \pi$). As it can be seen with equation 7, this case has a rather high probability to happen, and the deposited energy is

$$E_{dep} = E - E' = E - E/(1 + 2\gamma) \tag{9}$$

As shown in figure 9, at this energy, there is a peak in the energy spectrum and its location is coherent with the above equation.

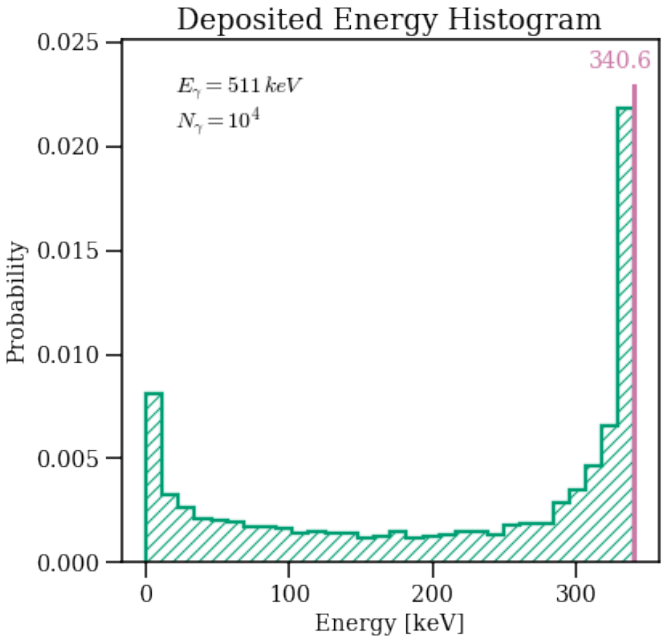


Figure 9: Histogram of the deposited energy for $N_\gamma = 10^4$ interacting photons, all of them having an energy of $511\,keV$. The last bin edge is located at $340.6\,keV$, which is coherent with equation 9, which leads to $340.6\,keV$ with $E = 511$ and $\gamma = 1$.

Photoelectric effect The photoelectric effect is the process in which an incoming photon kicks an electron out of the atom while being absorbed. It can be shown using four-momentum conservation that this process cannot happen with a free electron, which implies that this process necessarily takes place in an atom and the recoil momentum is absorbed by the nucleus. The energy of the kicked electron can be expressed as (BE is the binding energy of the kicked electron):

$$E' = E_0 - \mathrm{BE} \tag{10}$$

As one can expect, the binding energy depends on the atomic shell to which the electron belongs. The electrons in the outer shell are less bounded, which means that a photon with relatively low energy (typically a few keV, which corresponds to $X-$rays) will be able to kick those electrons off the atom easily, while more energetic γ rays are required to eject electrons from the most inner shells.

In the toy MC detector simulations, the photoelectric effect will be taken into account, but it is very difficult in practice to get numerical values for the differential cross section because the Dirac equation for the atomic electrons is very hard to solve [42]. There exist approximations but they are typically valid for low-energy photons (typically $E \ll m_e$). As it will be explained later, the cross-section for this process will therefore not be fully computed but will be deduced from the total cross-section. As it is explained later in this chapter, the cross-section will be converted from an online mass absorption coefficients database for various materials. Using equation 5, and the fact that the Compton cross section is analytically computable, it is possible to relatively easily and computationally quickly, infer a value for the photoelectric cross-section.

Pair production Is the process that corresponds to the Feynmann diagram shown in figure 10. It is straightforward to show that for this process to be possible the minimal photon energy needs to be greater than $E = 1.022\,MeV$, because this process creates two electrons having a rest mass of $511\,keV$. There exists development that allows computing the value of the pair production differential cross section but the computations are relatively complex. To keep things simple in the toy MC detector, the choice has been made to simply ignore this phenomenon by generating a photon with an energy smaller or equal to $1\,MeV$. Of course, in the true POLAR simulations, this process is taken into account by `GEANT4`, although its influence is minimal as the sensitivity range of POLAR and POLAR-2 is below 1 MeV.

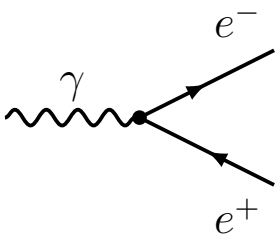
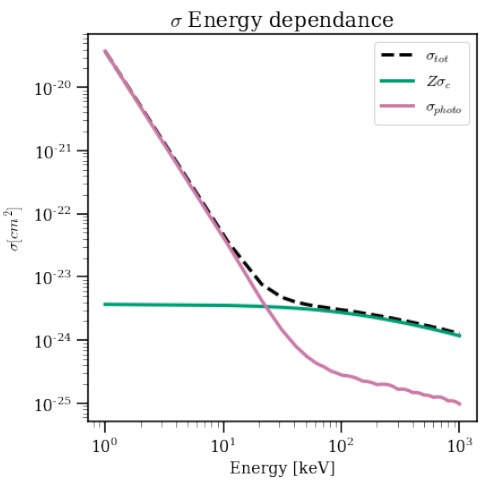


Figure 10: Feynman diagram of the pair production process.

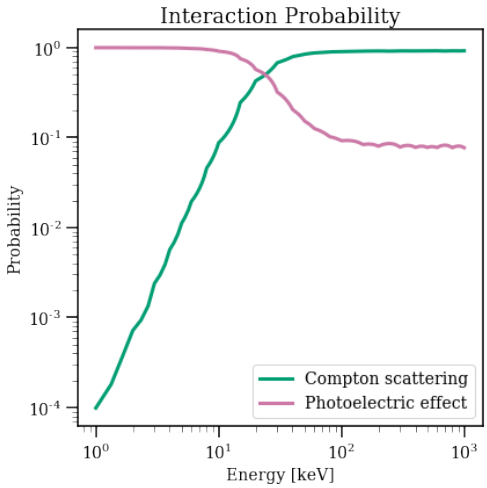
In the simulations that will be described in chapter 8, the total cross section will be taken from online tables found on https://www.nist.gov/pml/x-ray-mass-attenuation-coefficients (tables for "Vinyltoluene"). The Compton cross section will be computed analytically using 7 and since only energies lower than $1\,MeV$ will be generated (allowing us to ignore pair production), the photoelectric cross section will be deduced :

$$\begin{aligned}\sigma_{photo} &= \sigma_{tot} - \sigma_{Compton,tot} \\ &= \sigma_{tot} - Z\sigma_{Compton}\end{aligned}$$

where Z is the average number of electrons per atom in the material. Vinyltoluene is composed of 91.5% of Carbon and 8.5% of Hydrogen. It implies a molecular weight of $M_{scint} = 11.083\,g/mol$, an average number of electron per atom of : $Z_{scint} = 5.575$ and a density of $\rho_{scint} = 1.032\,g/cm^3$. Figure 11a are shows the graphs of σ_{tot}, $\sigma_{Compton}$ and σ_{photo} as a function of energy. In order to get data for all the energies in the range of interest, the data were interpolated with a first-order polynomial, which explains the fact that the function doesn't look smooth. Figure 11b shows the ratio $\sigma_{Compton}/\sigma_{tot}$ and $\sigma_{photo}/\sigma_{tot}$ as a function of the energy. It is clear that at low energies, the photoelectric effect dominates, whereas, at higher energies, it is the opposite.



(a) Total cross section and its different components as a function of the incoming photon energy. The total cross section is obtained thanks to the NIST tables



(b) Relative probability of interaction for both the Compton scattering and photoelectric effect. At low energies, the photoelectric effect dominates.

Figure 11: Plots of the cross-section and the relative importance of its components.

6 Machine Learning

Even though machine learning was originally developed in the late 50's [3], it is only in the last ten years that this field really exploded, with a tremendous number of applications such as license plate recognition, bank account safety checks, image recognition or even building chatbots like ChatGPT. Essentially, machine learning is an algorithm that learns a given dataset and its distribution, and can eventually do predictions on something the algorithm has never seen. In this thesis, supervised learning will be used to be able to infer and estimate the correct location and energy spectrum of a given GRB. For this thesis, models were trained using TensorFlow [1], a Python package developed by Google in 2015 that is very popular to train models in a relatively user-friendly manner. In the following, all the key components will be explained, but it won't be a complete description of the subject, as the focus is on its application to physics rather than machine learning. Only the basics of the concepts relevant to this work will be presented. All explanations are taken from [13] and [6] where a more complete description of all the covered topics can be found.

In this Master's work, Deep learning will be used for two tasks: Regression and binary classification. Regression can be defined as the task that consists to approximate a function's output, given an input feature. It will be used to find the incoming direction of a GRB as well as its spectral Band function parameters α, β and E_0. This task will be investigated using both fully connected neural networks and convolutional neural networks. On the other hand, a classification task is, as its name already explains, to classify an event given the input features. If there are only two classes, it is called *binary classification* and is widely used in image recognition algorithms (the ones that recognize if the input image represents a cat or a dog). The binary classification will be used to detect GRBs in a lightcurve (which is the number of detected photons by a detector as a function of time). As it is a time-dependent quantity, recurrent neural networks will be used. All those model types will be explained later on.

6.0.1 Basis

First, and as it is often a source of confusion, it is a good idea to emphasize the difference between "machine learning" and "deep learning". Machine learning is simply the ensemble of algorithms that are available in order for a machine to learn a dataset, whereas deep learning is one specific way. Deep learning uses an Artificial Neural Network (ANN) to learn the data. Artificial neurons are meant to reproduce neurons from the human brain which are interconnected in multiple layers. We here label the layers with index L, and the reaction of one neuron depends on the reaction of the other ones.
Each neuron has a trainable *weight* w_j^L and a trainable *bias* b_j^L that are summed and are the input of the activation function σ that modulates its output a_j^L (j is the label of the neuron index in layer L and k labels neurons in layer $L-1$). As a result, the output of a given neuron can be written as :

$$a_j^L = \sigma(z_j^L) \quad , \quad z_j^L = \sum_k w_{jk}^L a_k^{(L-1)} \tag{11}$$

At the creation of the model, the weight values are randomly initialized. The task that the algorithm has to perform in this work is, given a training set $\mathcal{D}$ of pairs (x, y) (x = detector response and y = localization/spectral parameters) the model will have to approximate a function f such that the function applied to the input and the trained parameters w^* ($\hat{y} = f(x, w^*)$) is a good estimator. As one could expect, the goodness is computed with a function $\mathcal{L}$, which should ideally be as small as possible (and vanishes when $y = \hat{y}$). The learning phase of the algorithm consists of updating the weights w such that the function $\mathcal{L}(w^*)$ is minimal. The model is trained for a given number of *epochs*, which is the number of times the algorithm looks through the whole dataset.

Losses The loss defines what quantity should be minimized, and is, therefore, a crucial component of the model. In the context of localization, for example, it can be useful to set the loss as the angular distance, because the physical angular separation between the actual incoming direction and the predicted one should be as small as possible.
It is good to note that in practice, the loss is computed by taking the average over a small sample called *batches*, with L being the loss computed on the target and predictions:

$$\mathcal{L}(w) = \frac{1}{N}\sum_{n=1}^{N} l_n(w) \quad \text{, where} \quad l_n(w) = L(f(x,w),y) \tag{12}$$

where N is the number of samples in a batch. The loss used in a model should be coherent with the task the model has to fulfil (classification, regression, etc...). Since the main goal of this work is to perform a regression, let's introduce the losses used:

1. Mean Absolute Error (MAE) is the most straightforward loss function in a regression problem. As its name already explains, it is computed by taking the mean of the absolute errors between true labels and predictions:

$$MAE = \frac{1}{n}\sum_{i=0}^{n} |y_{true} - y_{pred}| \tag{13}$$

   where y_{true} is the correct value and y_{pred} is the one predicted by the model. Note that this function is already pre-defined in TensorFlow.

2. Mean Squared Error (MSE) is another widely used loss in deep learning. It has the advantage to grow fast, hence big errors are more penalized. The analytical expression is the following:

$$MSE = \frac{1}{n}\sum_{i=0}^{n} (y_{true} - y_{pred})^2 \tag{14}$$

   This loss is also pre-defined in TensorFlow since it is quite a quite common one.

3. Angular distance is the physical angular separation between two points on a sphere. For the work performed here this quantity, which is a derived physical quantity, can be used as a loss. It is defined as :

$$\theta = \arccos[\cos(\theta_1)\cos(\theta_2) + \sin(\theta_1)\sin(\theta_2)\cdot\cos(\phi_1 - \phi_2)] \tag{15}$$

   This relation comes from the scalar product of two vectors pointing in two directions (θ_1, ϕ_1) and (θ_2, ϕ_2). If the angle between the vectors, the argument of the arccos is close to 1, whereas if they are directed "back to back", the argument is -1. This loss is used in directional regression problems as can be seen in [36]. This quantity has the advantage to be a single number and if it is small (*ie.* the two points are close one to another, it necessarily implies that the angles θ and ϕ, which define the two points, are close to one another.

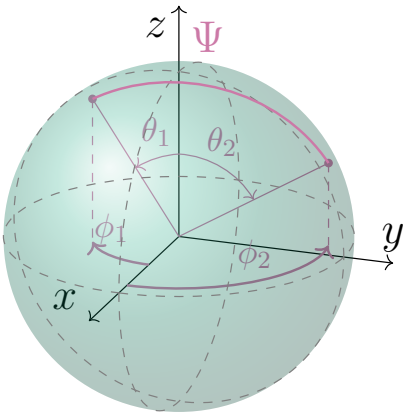


Figure 12: Representation of the angular separation Ψ between two points defined by spherical coordinates (θ_1, ϕ_1) and (θ_2, ϕ_2).

4. Binary crossentropy Is the most widely used loss for binary classification (true/false or 1/0) and is therefore also pre-defined in TensorFlow. Its analytical expression is the following:

$$BCE = -\frac{1}{N}\sum_i y_i \log(p_i) + (1 - y_i)\log(1 - p_i)$$

Even though this equation seems complicated, it is relatively simple to understand its working principle. Suppose the target label y is equal to one. As a result, the second part of the equation vanishes and the first part simplifies to a simple *log*. p_i is the probability of having a label predicted as 1. The loss of the predicted event to be closer to 0 should be large while a predicted label close to 1 should have a small loss. This need justifies the $-\log$ because it exactly satisfies those properties ($-\log(x)$ is very large for x close to 0). This loss is therefore also often called the *log loss* [16].

A range of other pre-defined losses exist, an overview can be found on the

Gradient descent Once the loss is computed, the weights have to be updated. The way to proceed is to compute the gradient of the loss with respect to all trainable parameters w, and to adapt the weights from a small quantity in that direction:

$$w_{n+1} = w_n - \eta \nabla_w \mathcal{L}(w_n) \tag{16}$$

where the w_n index represents the weights at the n^{th} iteration and η is the *learning rate*, which allows the user to set the speed at which the minimization will be performed. η is one of the parameters the user can change, which are called *hyperparameters*. It is relatively complex to set a value for η because if it is too small, the process will take a lot of time and can be stuck in a local minimum (defined by w_n such that $\nabla_w \mathcal{L}(w_n) = 0$, but the value of $\mathcal{L}(w_n)$ is not the minimal one). On the other hand, if it is too large, it can oscillate around the minima, without reaching it, so the weights will never minimize the loss.
There exist algorithms called *optimizers* like `Adam` (it is not an acronym [7]) that have the ability to compute rapidly the gradient descent with minimal memory requirement, with the possibility to adapt the learning rate as the model is training. This optimizer is known to be very fast and efficient [13] and will therefore be used in this work.

Activation Functions Are a class of functions that modulate the output of the neuron based on its inputs.
Let's call σ the activation function and x_i, w_i respectively the output and weights of the previous neuron (n in total). As a result, the output of the considered neuron can be expressed as $\sigma(z)$, with:

$$z = \sum_{i=1}^{n} x_i w_i + b$$

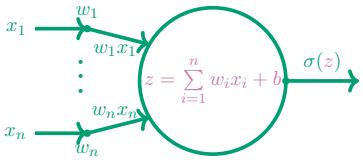


Figure 13: Scheme describing how the activation function is related to the output of the previous neurons and their weights.

b is the bias element used to shift the whole output. Figure 13 is a visual representation of this equation. Activation functions are very important since they can help the model to converge faster and they can also help to catch some of the non-linearity of the problem. The choice of the activation function often depends on the task the model has to fulfil (classification/regression), and it is even possible to define custom functions. Nevertheless, there are a few well-known functions, shown in figure 14. Each one has its pros and cons, but the `ReLU` (Rectified Linear Unit) is a common usage for all regression problems.

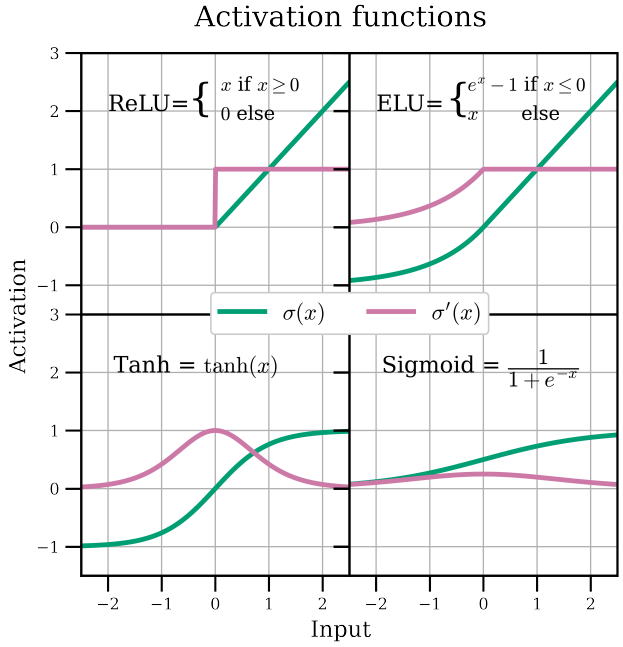


Figure 14: Plots of four different activation functions widely used in deep learning.

Backpropagation is the core of deep learning, because it is the mechanism that allows to update the weights in the correct direction. Once the loss is computed, the weights need to be updated, and backpropagation answers the question: "How much should the weight of a given neuron be changed?".
Since the neurons are interconnected, their output depends on the previous weights and biases. The effect on the loss of the activation of neuron a_k in layer $(L-1)$ can be determined through the chain rule :

$$\frac{\partial \mathcal{L}}{\partial a_k^{(L-1)}} = \sum_j \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial \mathcal{L}}{\partial a_j^{(L)}} = \sum_j w_{jk}^{(L)} \sigma'_j(z^{(L)}) \frac{\partial \mathcal{L}}{\partial a_k^{(L)}} \tag{17}$$

As a result, the dependence on the weights $w_{jk}^{(l)}$ on any layer l can be recursively computed by :

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^{(l)}} = a_k^{(l-1)} \sigma'(z_j^l) \frac{\partial \mathcal{L}}{\partial a_j^{(l)}} \quad \text{, where} \quad \frac{\partial \mathcal{L}}{\partial a_j^{(l)}} = \sum_j w_{jk}^{(l+1)} \sigma'(z_j^{(l+1)}) \frac{\partial \mathcal{L}}{\partial a_j^{(l+1)}} \tag{18}$$

This task is quite demanding in terms of computational resources, and the required computing power was only reached a few years ago, which explains why machine learning became a hot topic in the last ten years, whereas it has been developed in the 60s.

Sample division It is standard in machine learning to divide the dataset into three different subsets: training, validation, and testing. The main objective of any deep learning problem is to ensure good training. By separating the full dataset it is possible to test the model on the validation sample during training (hence the name validation), which makes it possible to monitor the effects of the model on a different set. It is important to note that the validation set is only to monitor the performances and the training is only done on the training sample. The measure is called a *metric* and is not necessarily the same function as the loss, it can be MSE for the loss and MAE for the metric for instance. The test set is used only for the final test with data that the model has never seen before. No optimization is done for this sample. A quite standard ratio is to use 80% of the dataset for the training, 10% for the validation, and the rest for the testing sample. Note that it can vary a bit, the relevant parameter is the order of magnitude.

Overfitting Over the epochs, the model is trained to reduce the loss. What can happen is that the model manages to be so accurate that it begins to train on irrelevant data (like statistical fluctuations). In this case, the model finds a local minima that is different from the validation set minima. The result is that the models get worse at predicting the validation data, and the monitored quantities will grow. As a result, it is quite standard to define the point at which the validation performances are at their best as the point where the model should stop training.
There exist different architecture types one can build. In this work, three types of networks will be explored: Fully Connected Neural Networks (FCNN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN), which will all be explained in greater detail in the next paragraph.

6.0.2 Fully Connected Neural Networks

Fully connected neural networks are the simplest types of neural networks one could think of, as they are composed of N_L layers, each one of them made of a given amount of neurons all connected, where each layer can have a different activation function. The fact that they don't do any special operation (unlike the convolutional networks which will be discussed next) makes them a good choice to start experimenting with neural networks. It is, however, possible to make more complex architecture by separating the network into different branches, which only changes the complexity of the network, not its working principle. A typical use of such a separation is for a model that has multiple outputs.
A network is always separated into three different stages: the *input* layer is the one that receives the data, followed by the *hidden* layers: The number of layers in the hidden part of a network can vary as well as the number of neurons in each one of them. Their aim is to catch the underlying relations between input data. The last part is the output layer, where the model provides the final prediction. As one can expect, there are many different types of layers that one can use in the hidden part of the network. However, it is possible to distinguish three layers which are the most widely used: `Dense`, `Dropout` and `BatchNormalization` layers which are defined as:

1. Dense layer: is the most simple type of layer one can define. It is an ensemble of neurons, all of them having learnable weights and biases. The activation function is the same for all neurons in one `Dense` layer. This type of layer will naturally be the main building block of the FCNN that will be trained in this thesis.

2. Dropout is a type of layer that has no learnable parameters. It has the same number of neurons as the previous layer, and its only task is to turn off a given percentage of the neurons at each training step (obviously, when the model is trained, all the neurons are turned on). The use of such a technique forces the model to learn with different paths. It is known to prevent overfitting [37].

3. BatchNormalization is a type of layer that has two learnable parameters per input dimension (the shape of the array that constitutes the model's input) and is meant to modify data provided to a given layer. This layer type is widely known to accelerate the learning process [8]. It is used to solve the fact that deep neural networks are very sensitive to the distribution of the inputs by standardizing the data provided to a given layer at each batch. Its learnable parameters `gamma` (the scaling factor) and `beta` (the offset factor) are used to normalize input data after the training phase is done. It is hence important to perform inference on data that has similar statistics as the training data [38].

6.0.3 Convolutional Neural Networks (CNN)

These types of networks are well known for their great performance on image recognition problems. They achieve those good results by having the possibility to have a complex architecture while keeping a reduced amount of parameters. As their name explains, their working principle is based on convolutions. It is a mathematical operation on two functions f and g that is defined as (for the continuous case, the sum becomes an integral over da):

$$(f * g)(c) = \sum_a f(a) \cdot g(a - c) \tag{19}$$

This operation is often compared [28] to a test function (f) on which a kernel (g) is moving and all possible configurations are summed over. In the context of neural networks, it is exactly how it works, except that the function f is an image (it can be in general a N-dimensional array, but in practice, it is mostly used for $2D$ arrays and rarely on $3D$ arrays). The function g is an array with an equal number of dimensions and is called a *kernel*. For $2D$ arrays, the product is the scalar product and the kernel moves in two dimensions, along the horizontal axis but also along the vertical one. The output of the convolution is a $2D$ array. Note that convolutions are also used by image treatment programs for many applications, like blurring an image or detecting the edges, for which the following kernel is applied [15]:

$$g = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

This kernel has a size of (3×3), which is the *kernel size*. Note that several kernels can be applied to the same image (*eg.* detecting the edger, then blurring). The reason why CNN can keep a relatively low number of weights while being deep is that the weights (the numbers in a given kernel) are used multiple times over the full image.
When analyzing $2D$ arrays with CNN, there are usually two types of layers that are combined: the `Conv2D` and the `MaxPool2D`.

1. Conv2D is the basis layer for $2D$ convolutions. It allows to define the number of filters that will be applied on the $2D$ array (it determines the output dimension), the kernel size, but also the activation function. It is also possible to set parameters such as `strides` and `padding` that are useful to make the dimension of the output the same as the input dimension as explained in [39].

2. MaxPool2D is a type of layer that is often placed after the `Conv2D` ones. It is used to reduce the dimension of the output by returning the maximal value of the $2D$ array over a window of a given size. There also exists the `MinPool2D` or `AveragePooling2D`, whose names are self-explanatory.

After the convolutions and pooling layers, one generally places a fully connected neural network to analyze the convolutions. However, it needs to have a $1D$ vector as input, which justifies the use of a `Flatten` layer that converts any multi-dimensional input into a $1D$ one.

6.0.4 Recurrent Neural Networks

Recurrent Neural Networks (RNN), are widely known for their ability to analyze/forecast time series [40]. The core idea of RNN is that the output of the neurons is also part of its inputs (it is *folded*) as can be seen in figure 15. Since the neuron weight is the same for the whole time series, it is easy to understand that if the time series are long ($\gg 1$) and the weight is different from one, the neuron's output can either explode or vanish. This is known as the *vanishing/exploding gradient* problem. As a result, RNNs have difficulties learning long time series. There exist solutions such as the LSTM (Long Short Term Memory) networks [20] that are designed to hold information for a longer period. Because of the great results it showed, LSTM can be a subject on its own, but it would go far beyond the scope of this work. To learn more, one can visit the blog of Olah [28]. In this work, RNN/LSTM will not be used differently from a fully connected neural network since TensorFlow features both the `simpleRNN` and the `LSTM` layers that can be combined easily with the `Dense` layers. They both accept an integer as a parameter that represents the number of neurons in the layer. The input x of an RNN/LSTM cell is composed of its time evolution, each one being a number x_t. The same is true for the output y.

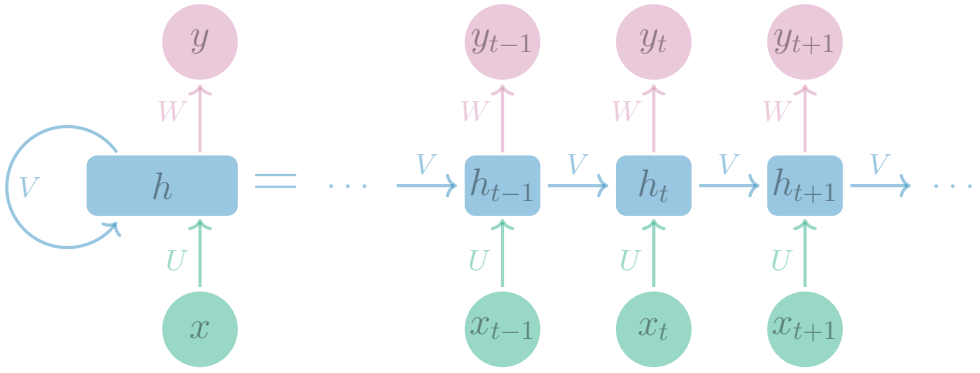


Figure 15: Scheme of an unfolded recurrent neural network. The input is labelled as x and the output y. Since x is a time series, it consists of several time dependents components x_t (t is the time), and so does y. U, V, and W are respectively the weights of the input, the feedback loop, and the output. Note that the weights are the same for the whole sequence but the state h_t depends on all the previous elements of x.

7 Random number generation

Since we need to build a Monte Carlo simulator, many random quantities have to be computed. The main problem in those simulations is to draw random numbers according to a given probability density function (*pdf*) $f(x)$ defined on the interval $[a, b]$. There are two popular methods to do so: The *Accept-Reject* method (also known as the *Von Neumann* method) and the *inverse transform* method [31]. Both are described below.

Inverse Transform This method is computationally speaking the fastest because all computed random numbers are used. Let's assume that the integral of $f(x)$ can be computed on the range $[a, b]$. As a result, its Cumulative Density Function (CDF) is expressed as:

$$F(x) = \int_a^b f(t)\,\mathrm{d}t \tag{20}$$

Since $f(x)$ is a *pdf*, it is a non-negative function whose integral is equal to 1, hence the CDF is a non-decreasing function on the interval $[0, 1]$.

Assume now that $U \sim \mathcal{U}(0, 1)$ is a uniform random variable between 0 and 1. Then, according to lemma 2.4 of [31], $F^{-1}(U)$ has the same distribution as F.

As a result, it is possible to generate random numbers according to a distribution $f(x)$ as long as the integral can be computed. This is however not always the case. A simple example of a commonly used distribution is the normal distribution $f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right)$ whose integral cannot be computed easily. For such functions, the accept-reject method can be used instead.

Accept-Reject This method, which is illustrated in Figure 16, allows to draw of numbers according to $f(x)$, without the condition that the integral of f must be computable. Let's note m the max of f on the interval $[a, b]$. The problem is to generate a set of two random variables :

$$(X, U) \sim \mathcal{U}\{(x, u) : 0 < u < f(x)\} \tag{21}$$

It, however, appears that to sample directly from this joint distribution is often hard, and the problem can be solved in a much simpler manner. It can be done by sampling (x, U) on a bigger set (say (Y, U)) that is easier to simulate[6] and then select or discard events that satisfy a given condition. More precisely, it means that it is easier to generate the pair (Y, U) such that $Y \sim \mathcal{U}(a, b)$ and $U \sim \mathcal{U}(0, m)$. If only the pairs where $u < f(y)$ are taken, then Y has the same distribution as X. The reason is that, by the definition of Y, we have :

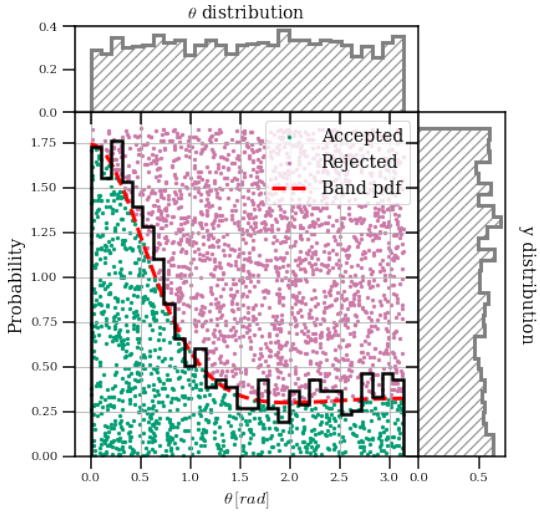
$$\begin{aligned} P(X < x) &= P(Y < x \,|\, U < f(Y)) \\ &= \frac{\int_0^x \int_0^{f(y)} \mathrm{d}u\,\mathrm{d}y}{\int_a^b \int_0^{f(y)} \mathrm{d}u\,\mathrm{d}y} = \int_a^x f(y)\,\mathrm{d}y \end{aligned}$$

where the integral on the numerator represents the proportion of elements accepted and the integral on the denominator represents all the elements generated.

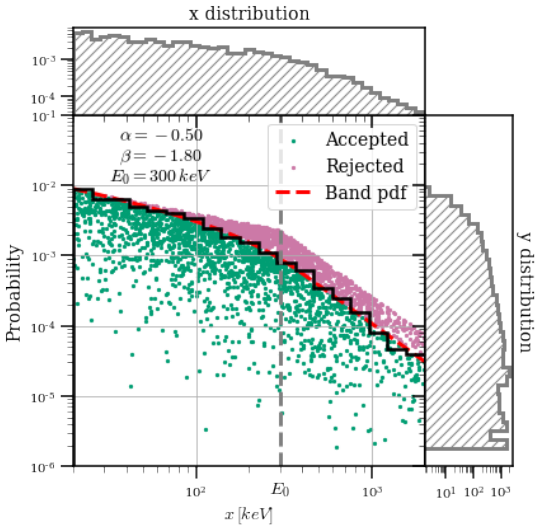
Note that this algorithm also works if X is sampled from a distribution g that satisfies $g(x) \geq c \cdot f(x)$ (with $c \in \mathbb{R}$) from which it is easy to sample from and $y \sim \mathcal{U}(0, g(x))$. This property can be very useful when the desired distribution is not *close* to the uniform distribution, which means that a lot of points are rejected. in this work, it will be used to sample from a Band distribution (see 8.1.1)[7].

[6] It often means to sample from a uniform distribution

[7] The idea is that where the function g is low, the probability to draw a low number U is high (because $\mathcal{U} \sim g(x)$) but there is also a lower probability for X to happen for the same reason.



(a) Illustration of the accept-reject algorithm as described in the first part of 7 using the unpolarized Klein-Nishina to build a probability density function (as explained in 8.1.1). All the 3000 θ coordinates are generated uniformly between 0 and π, while the height is distributed uniformly between 0 and 1.05 times (for visual purposes) the maximal value reached by the probability density function on the interval. The points above the function are rejected and the ones above are accepted. The **black** line is obtained by binning the θ coordinates of the accepted points.



(b) Illustration of the accept-reject algorithm as described in the second part of 7 using the Band probability density function in red. All the 3000 energy (x) values are generated as a power law whose spectral index α, β, and E_{break} are derived from the Band parameter such that the power law is always greater than the Band function. The points above the function are rejected and the ones below are accepted. The **black** line is obtained by binning the energy values of the accepted points.

Figure 16: Accept-Reject method applied for two different distributions used in the simulations.

8 Toy Monte Carlo detector

Before applying ML methods to complex instrument data a toy MC was built. This allowed to produce test data that is fully controlled and misses the complexities from real data or POLAR simulation data which, in some cases cannot be distinguished from issues in the ML analysis. The aim of the Toy MC is therefore not to be a better version of the official POLAR simulations, but rather to be more familiar with the detector physics, to control the data more easily than the true POLAR simulations (that are far more complicated) and to develop intuitions on the topic.
Even though the aim is not to reproduce `GEANT4`, the simulations have to respect the laws of physics. The two important elements that will be simulated are the sources (*ie.* the background and the GRB) and the detector. In practice, the background photons are coming from outer space, which includes solar flares, activity in the galactic plane, and extragalactic sources like blazars. However, most of the photons come from particles decaying in the higher layers of the atmosphere and emitting X or γ-rays. Therefore, the background will be considered as homogeneous and isotropic (*ie.* the density of γ-rays is the same in any direction). For the real POLAR simulations, since the background data will be real flight data, this won't have to be simulated.
The final goal of the project is to, given counts per bar and the energies measured during a GRB, predict accurately its incoming direction and spectral parameters. Those data will be obtained from the simulations that simulate the effect of GRBs having different incoming directions in the detector.
Here is the list of elements that the simulations will have to fulfil:

Sources

- The incoming direction of the population of GRBs must be isotropic (*ie.* there is no preferred incoming direction).
- The GRB spectrum is defined by the Band function as defined in equation 1.
- The GRB is simulated by a disk larger than the detector that emits photons in the direction of the detector. Thereby ensuring that the full detector is irradiated during the simulations.
- The background is homogeneous and isotropic and its spectrum is defined as a power-law spectrum with a fixed index.

Detector

- The detector is composed of a multitude of small units that represent the scintillator bars, this ensures that the detector resembles POLAR and allows it to perform localization analysis in a similar method as POLAR.
- The bars are made of Vinyltoluene, the same material as the real POLAR scintillator bars.
- Photons that enter the detector can interact via Compton scattering or the Photoelectric effect.
- The detector has an uncertainty of 10% on the energy measurement.

As there are many elements to verify in those simulations, two toy Monte Carlo detectors were built. The goal is to verify things separately in order to avoid numerous unknown errors. The first of these detectors is made of $4 \times 4 \times 4$ scintillator bars. All photons (background and source) have an energy of $50\,keV$. Photons can enter the detector and interact with the material through Compton scattering or the photoelectric effect. The detector, however, only records whether an interaction has taken place in a bar, it doesn't record the deposited energy.

The second detector is the final version and looks more like POLAR. It is composed of 8×8 scintillator bars. The background photons have energies defined by a power law of index $n = -3$ and the GRB photons have energies defined by the Band function. The distributions of the α, β and E_0 parameter will be described later. Photons can interact through Compton scattering and via the Photoelectric effect. The energy is recorded and has a 10% error.

8.1 Code description

In this section, we describe all the classes and functions used in the code. However, only the meaningful methods will be described (that means, not the traditional getters[8] and drawing methods.) In order to make the code much faster (at least 10 times) and maintain a reasonable computing time, using Numba[9] is recommended.

8.1.1 Random Generators

In the following are described the different random generators used in the simulations. For more details on random number generation, see chapter 7.

Spherically Uniform distribution The goal of this generator is to generate random N incoming directions (θ, ϕ) of the photons with respect to the detector. This is a well-known problem and the key is to generate ϕ uniformly from 0 to 2π and θ as $\arccos(1-2u)$, where u is a random number between 0 and 1. The $\arccos(1-2u)$ is obtained by the inverse transform method by imposing that the density on a small area dA around any point v on the sphere is uniform, which means :

$$f(v)\,\mathrm{d}A = \frac{1}{4\pi}\,\mathrm{d}A = f(\theta,\,\phi)\,d\theta\,\mathrm{d}\phi$$

Because $\mathrm{d}A = \sin(\theta)\,\mathrm{d}\theta\,\mathrm{d}\phi$, and the ϕ angle has to be uniformly generated, it follows that $f(\phi) = 1/2\pi$ and $f(\theta) = \sin(\theta)/2$. By applying the inverse transform method, it is then clear that $\theta \sim \arccos(1-2u)$. A random point on the sphere is hence defined as $(\theta,\,\phi)$ with $\theta \sim f(\theta)$ and $\phi \sim \mathcal{U}(0,\,2\pi)$.

If the θ angle was chosen uniformly between 0 and 1, the points would have been accumulated on the poles as explained in [34].

Unpolarized Compton Since photons entering the toy Monte Carlo detector can scatter, a proper angle generator has to be implemented. The probability function is derived from the unpolarized Klein-Nishina differential cross section (see *eq.* 8) by first computing the total cross section:

$$\sigma_{KN} = \int_0^{\pi} \left(\int_0^{2\pi} \left.\frac{\mathrm{d}\sigma_{KN}}{\mathrm{d}\Omega}\right|_{unpol} \sin(\theta)\,\mathrm{d}\phi \right) \mathrm{d}\theta \tag{22}$$

where the integral over $\mathrm{d}\phi$ can be expressed as $\frac{\mathrm{d}\sigma_{KN}}{d\theta} = 2\pi \frac{\mathrm{d}\sigma_{KN}}{\mathrm{d}\Omega} \sin(\theta)$. Note that the total cross-section is the same for both the polarized and unpolarized cases. Finally, $f(\theta)$ can be obtained by simply dividing the differential cross-section by the total cross-section :

$$f(\theta) = \frac{1}{\sigma_{KN}} \frac{\mathrm{d}\sigma_{KN}}{\mathrm{d}\theta}$$

This random number generator was done with the accept-reject method because $f(\theta)$ has no simple integral.

[8] The easy method that allows getting access to the class instance

[9] See https://numba.pydata.org/

Mean free path In order to simulate the path of a photon in the detector we need to generate a mean free path. To do so, equation 3 has to be used. For a given energy, the ratio $I/I_0 \leq 1$. That means that in order to generate a random path t, it is sufficient to generate $u = I/I_0 \sim \mathcal{U}(0,1)$ and invert the relation. The path travelled by the photon reads :

$$t = -\frac{\log(u)}{\mu(E)}$$

As a reminder, the μ value is obtained from a database found online, where the μ/ρ values for the Vinyltoluene are given as a function of the photon energy (see chapter 5.0.1).

Cosine distribution This distribution is very useful to generate background photons. For spherical volumes, as mentioned in [32]: *"Isotropic angular emission from the surface leads to non-isotropic fluence in the volume"*. This is not the case if the angles follow the *cosine law*. This law is explained in [17] where the probability density functions for θ and ϕ can be derived from the constraint that the flux dn across a surface A follows this specific law (let N_0 be the total flux):

$$dn = \frac{N_0}{\pi}\cos(\theta)\mathrm{d}\Omega \tag{23}$$

where $d\Omega$ is the infinitesimal solid angle element. Since the above doesn't depends on ϕ, it can be generated uniformly : $\phi \sim \mathcal{U}(0, 2\pi)$. On the other hand, $f(\theta)$ can be found by integrating the above on $\mathrm{d}\phi$: $f(\theta) = \sin(2\theta)$. By applying the inverse transform method, θ can be generated $\theta \sim \arcsin(\sqrt{u})$.

Broken power spectrum This kind of random generator is particularly useful when generating energies from background or source photons. Since the integral of a power law $f(E) \propto E^{-n}$ is well known (*ie.* $\propto E^{1-n}$), it is possible to use the inverse transform method for this case because a broken power law is simply defined as two power laws of indices α and β separated by a breakpoint E_{break}. The total integral is simply given by the sum of the two power laws $I = I_1 + I_2$, and their relative weights w_1 and w_2 can be computed easily: $w_1 = I_1/I$ and $w_2 = I_2/I$.

The only difference with respect to the other previous uses of the inverse transform method is that in order to generate N random numbers it is necessary to generate two uniform samples (one for each power law) with relative length $n_1 = N \cdot w_1$ and $n_2 = N \cdot w_2$ and apply the inverse transform to both lists. The list formed by the concatenation of list 1 and 2 has a broken power law distribution as shown in figure 16b, where the broken power law distribution was used to sample from a Band distribution.

Band Even though it is possible to integrate the Band function[10], the integral is defined with the incomplete Gamma function in which one would have to provide negative arguments. There is a solution, however, it is faster to use the accept-reject method. Since the band function is essentially defined by power laws, it is easy to understand that a lot of computing power is wasted, because there is a very low acceptance rate (because the function drops very fast on the very wide range being used: $[1\,kev, 1\,MeV]$). The solution is to generate the pair (Y, U) more cleverly than using a uniform distribution. The idea is the same as what is described in chapter 7. Hence, both the Y and U random variables were sampled from a broken power law that was close to the desired shape. The result is a fast and reliable Band power spectrum random generator as can be seen in figure 16b.

[10] With `Mathematica` it is possible to use clever assumptions to make this integration easier.

8.1.2 Classes

Since the toy Monte Carlo simulations involve different components, oriented object programming is a good way to proceed[11]. Apart from the function libraries, there are four different classes: `Photon`, `Source`, `Volume`, and `Detector`. Those classes are described below. Since different detectors were built, the code was different. The classes and functions explained below are the complete versions.

`Photon` class is the class that describes a photon, its direction, and its energy. The position is simply defined by two vectors, namely the initial position $\vec{\mathbf{x}}_0$ and the direction $\vec{\mathbf{x}}_d$ (both in Cartesian coordinates). As a result, the trajectory can be computed with

$$\vec{\mathbf{x}}_\gamma = \vec{\mathbf{x}}_0 + t \cdot \vec{\mathbf{x}}_d \quad , \quad t \in \mathbb{R} \tag{24}$$

The principal component of this class is the `scatter` method, which allows the photon to scatter (*ie.* change its direction and energy according to equations 6 and 8).
The way to proceed is to generate a random unit direction $\vec{\mathbf{x}}_{scat,0} = (1, \theta_{kn}, \phi_{kn})$ with θ_{kn} and ϕ_{kn} random angles generated with unpolarized Compton random number generator (see 8.1.1). Converting the photon direction $\vec{\mathbf{x}}_d$ in spherical coordinates directly gives the $\vec{\theta}_\gamma$ and $\vec{\phi}_\gamma$ angles. The direction of the scattered photon $\vec{\mathbf{x}}_scat$ is obtained by rotating the random direction $\vec{\mathbf{x}}_{scat,0}$ of angles θ_γ and $\phi\gamma$ about a given point using rotation matrices[12]:

$$\begin{aligned} \vec{\mathbf{x}}_{scat} = R_{\theta,\phi} \cdot \vec{\mathbf{x}}_{scat,0} &= R_\phi \cdot R_\theta \cdot \vec{\mathbf{x}}_{scat,0} \\ &= \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \cdot \vec{\mathbf{x}}_{scat,0} \end{aligned} \tag{25}$$

In order to verify this method, $N_\gamma = 10^4$ `Photons`, all of them having an energy of $511\,keV$ were created and scattered. The deposited energy ($\Delta E = E_\gamma - E'_\gamma$) was then computed using equation 9 for all of them. The result is shown in figure 9. We see two peaks. The first one comes from the higher probability that a photon gets forward scattered, which implies a small deposited energy. The origin of the second peak (the Compton edge) is the same, except that the probability is higher. Its position can easily be computed with equation 6 and in this case, it should be at an energy of $340.67\,keV$, which is the case.

`Volume` class is the class that represents the scintillator bars. They are defined by an Axis-Aligned Bounding Box (AABB, which is widely used in video games), itself defined by two vectors: the position of the bottom left corner and the one of the upper opposite corner. The methods are the following :

`ray_intercept` takes a photon as a parameter. If the cube is on the path of the photon, the method returns the point (a vector) where the photon gets in contact with the cube. On the other hand, if the photon does not hit the cube, the method returns `None`. The principle is to solve the intersection between the ray and the AABB in each direction ($\hat{x}$, $\hat{y}$, $\hat{z}$). For example for the x direction, the system to solve is the following :

$$x_\gamma(t_{1,x}) \stackrel{!}{=} x_{min} \quad \text{and} \quad x_\gamma(t_{2,x}) \stackrel{!}{=} x_{max}$$

[11] The code was written in Python. After using and making the code more complicated, it appeared that it was too slow, which required the use of `Numba`, a package that allows compiling Python functions. It made the code ~ 10 times faster but it is not optimal. If I had to do those simulations again, `c++` for speed purposes.

[12] In general this point is the location of the position of the photon but for testing purposes, this point can be defined by the user

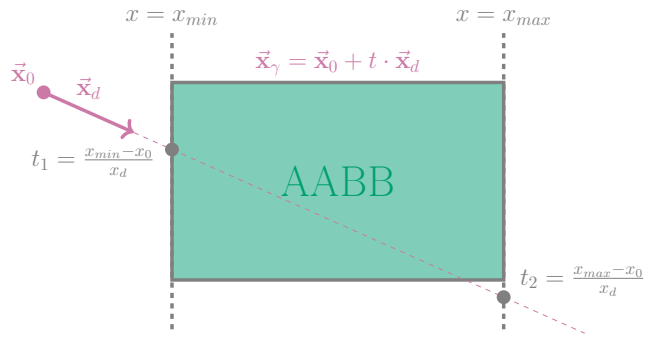
which leads to [13]:

$$t_{1,x} = \frac{x_{min} - x_0}{x_d} \quad \text{and} \quad t_{2,x} = \frac{x_{max} - x_0}{x_d}$$

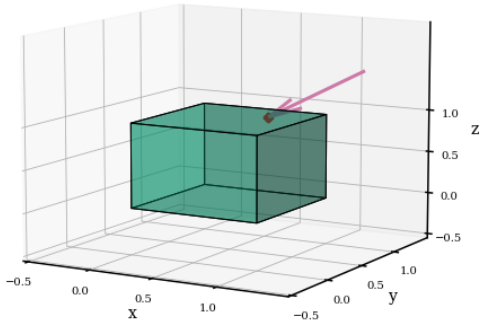
A visual representation of the computation is shown in figure 17a.

After performing this computation on the x, y, and z direction, we can check if the photon has entered or not by checking that in all directions, $t_2 > t_1$. If it is not the case, the photon does not collide with the AABB and the function returns `None`. If there is a collision, we can compute the (x, y, z) coordinate of the collision by finding the minimal time value and computing the photon position with equation 24.

In order to verify that this method worked as expected, an instance of both `Photon` and `Volume` was created. The 511 keV photon has its origin at $\vec{x}_0 = (1.2, 1.2, 1.5)$ and is directed along $\vec{x}_d = (-1, -1, -1)$. The AABB has a cubic shape and has its low left corner located at the origin. All the sides have a length of 1. In figure 17b we can see the `Photon`, the `Volume` and the intersection point. Everything is working as expected.



(a) Scheme of a $2D$ Axis-Aligned Bounding Box and a photon.



(b) Intersection of an AABB and a photon. The red dot shows the calculated intersection

Figure 17: Visual representation of two AABBs interacting with a photon

`is_In` is a function that returns `True` if the point that is given as a parameter is in the volume, and `False` if not. In order to do so, it checks for the $\hat{x}$, $\hat{y}$, $\hat{z}$ directions if the corresponding component is in the range defined by the two edges that define the AABB. For instance, in the $\hat{z}$ direction, it checks if:

$$z_{BL} \leq z_{point} \leq z_{TR}$$

where z_{BL} is the z coordinate of the bottom left corner and z_{TR} the one of the top right one.

`Detector` class inherits from the `Volume` class (because a detector is composed of several bars). It is defined by an integer that represents the number of bars by side and two floats that represent the height and width of a bar. At the creation of a `Detector` instance, a `Volume` with the same size as the `Detector`. It represents the shell of the detector and is useful for speed purposes as it will be explained. The functions are the following:

`inWhich` takes a photon as parameter. It loops over all the `Volume` instances that the detector is made of and applies the `is_In` method. The output of that function is the bar index (the number that determines the detector element) in which the photon is. If there is no photon in the detector, `inWhich` returns `None`.

[13] Before doing the division we check that none of the direction components is zero. If it is the case there are different ways to get around. we simply replaced the zero with infinity as explained in https://pyrr.readthedocs.io/en/latest/_modules/pyrr/geometric_tests.html#ray_intersect_aabb

`rayCapture` is the generalised version of `ray_intercept`. It also takes a photon as an option. It starts by checking if the detector shell is in the direction of the photon, which is faster than doing this check for each bar in the detector. If it is the case, the `inWhich` method is called and the function returns both the bar index in which the photon is and the photon's position. If the detector shell is not on the photon's path, the function returns `None, None`.

`absorbRay` is the most important method for the `Detector` class. It allows to track the photon in the detector, taking into account the Compton process and photo-absorption.

The first step is to know where the photon comes in the detector (if it does so). For that, the method `rayCapture` is used. If the result is not `None`, it means that the photon hits the detector. If it is the case, a mean free path is generated (as explained in 8.1.1, it depends on the energy of the photon). Additionally, a random number between 0 and 1 is drawn. This number is then compared to the probability for the photon to interact via the Compton process. This probability is simply obtained by the following ratio :

$$P_C = \frac{Z_{scint} \cdot \sigma_c}{\sigma_{tot}}$$

where σ_c is the Compton cross section and Z is the average number of electrons per atom in the scintillator bars. If the generated number is smaller than this probability, it means that the photon interacts via Compton, otherwise it interacts via the photoelectric effect.

Let's suppose the photon interacts via the Compton process. The procedure checks that after travelling the random path, the photon is still in the detector, in which case the photon is scattered, and the deposited energy is measured with a 10% accuracy. Then, a path and a probability are generated and the loop continues until the photon gets out of the detector or the photoelectric effect is chosen. In this case, a check is performed to verify that the photon is in the detector, and the deposited energy is measured with an accuracy of 10%.

`Source` **class** is the class that defines all the sources that "emit" photons, namely the background and the GRBs. As a result, this class has two main methods, one for each kind of source.

GRBs are simulated by a disk of radius r (bigger than the detector size) and distance d. The disk centre is located at spherical coordinated (d, θ, ϕ).

A point on the circle can be defined by $\vec{x}_{rand} = r_{rand}\left(\sin(\theta_{rand})\hat{\theta} + \cos(\theta_{rand})\hat{\phi}\right)$. Using the fact that the infinitesimal number of emitted photons per unit area is $dn = \rho \cdot \mathrm{d}A$ (with ρ the constant density), it can be shown (using the inverse transform method) that $\theta_{rand} \sim \mathcal{U}(0, 2\pi)$ and $r_{rand} \sim \sqrt{u} \cdot r$, where $u \sim \mathcal{U}(0, 1)$.

All the photon's initial position can be defined by the sum of the disk centre and the rotated random direction (in order for the random direction to stay on the tilted disk) :

$$\vec{x}_0 = \vec{x}_{disk,cent} + R_{\theta,\phi} \cdot \vec{x}_{rand}$$

All photons have $\vec{\mathbf{x}}_d$ given by (in spherical coordinates) $\vec{\mathbf{x}}_d = (-d, \theta, \phi)$ so that they are all directed towards the detector. Their energies are distributed according to the Band function (see equation 1 and chapter 7).

background is the method that generates a given number of background photons. In order for the background to be homogeneous and isotropic near the detector, the photons are emitted from a sphere of radius d_{BG}, centred at the origin. The photons are emitted from random points chosen uniformly on the sphere, once the random point is defined, a random

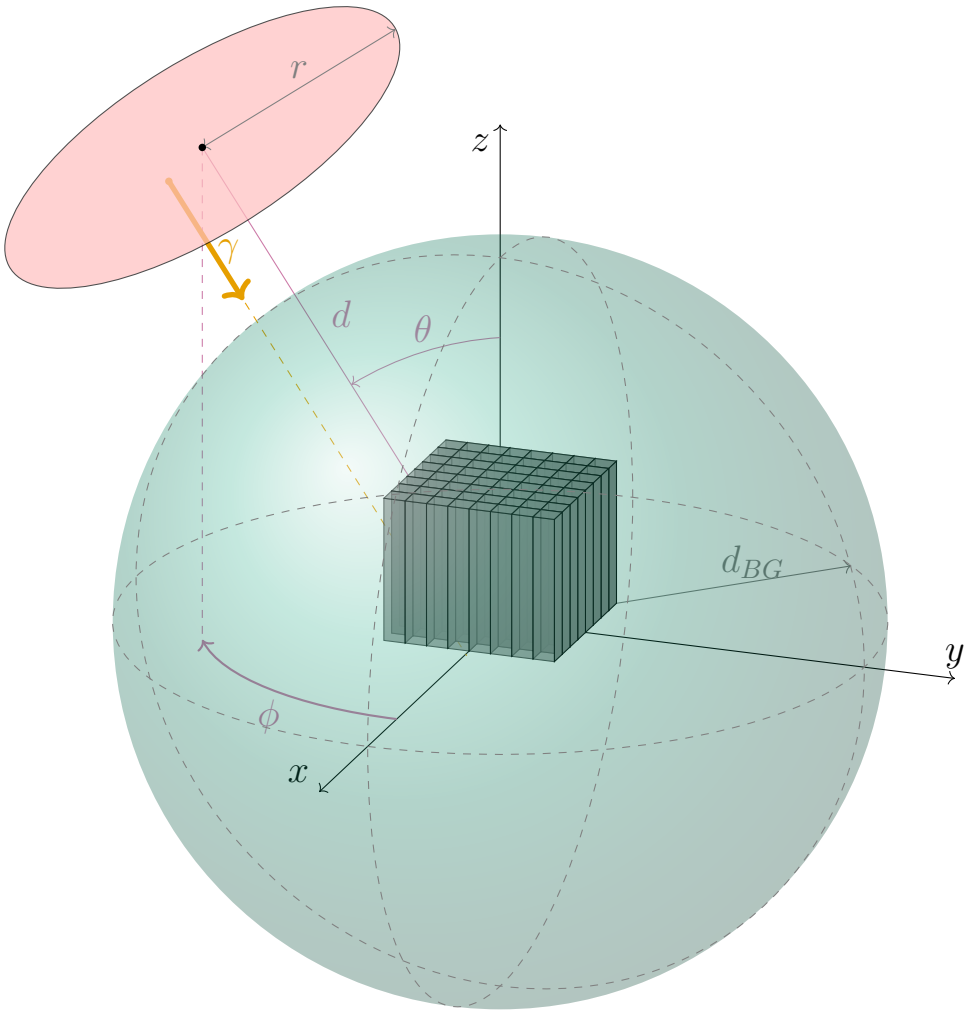


Figure 18: Scheme of the different sources defined in the `Source` class. In green is the sphere that emits the background photons in a homogeneous and isotropic way. The pink is the disk that emits the GRB photons, whose energies are distributed according to the Band function 1. At the origin is shown an 8×8 bar detector.

direction is drawn according to the *cosine law*, then rotated such that the generated photons tend to be directed towards the centre of the sphere. The energy of the background photons is distributed according to a power law of index $\alpha = -3$ which matches roughly the power law spectrum as seen by POLAR.

8.1.3 GRB generation

Using all the classes presented above, it is relatively straightforward to generate a lot of GRB simulations that will be passed to the different models as training samples.
In order to keep things simple, the only parameters the user has to provide are:

1. The sample size (The number of GRBs).
2. The number of GRB photons that are requested to interact with the detector.
3. The ratio allows the program to compute the number of background photons that will interact with the detector.

In real conditions, the background can not be selected and depends on the GRB incoming angle. Here the simulations are done with a fixed ratio such that it is possible to analyze the sensitivity of the different models on the ratio.

To be faster, the script starts by building a background photon database so they don't have to be generated at each iteration. The detector must be on the path of all photons in this database such that they all interact with it. The size of that database contains typically tens of times the number of background photons that interacts with the detector in each GRB simulation.
Each GRB simulation is done by first selecting a random pair (θ, ϕ) uniformly on the sphere. This point will be the location of the GRB. Three random values for the Band spectrum are also generated as follows [14]:

$$\alpha_{Band} = \{\alpha \mid \alpha \sim \mathcal{N}(-1, 0.48^2) \ \wedge \ (-3.1 < \alpha \leq 1.5)\}$$

$$\beta_{Band} = \{\beta \mid \beta \sim \mathcal{N}(-2.3, 0.9^2) \ \wedge \ (-3.1 < \beta \leq \alpha_{Band})\}$$

$$E_0 \sim \mathcal{U}(300, 1000)$$

Then, batches of $2 \cdot 10^4$ GRB photons are generated, and their interaction are computed (it appeared that this size is a good time/efficiency balance). If the desired number of interacting GRB photons is not reached, batches can be generated in consequence. Once the desired number is reached, the process is the same as the background photons, except that they are drawn randomly in the database.
Once these steps are done, all that is left is to count the interactions and deposited energies in each one of the scintillator bars. The count per bar is represented by a single number whereas the energies per bar are binned in a histogram with 32 logarithmic bins. The result of a typical simulation is shown in figure 30.

[14] Those are the distributions of the Band parameter used in the POLAR simulations that will be described later

9 Localization analysis with the toy detector

In this section, different deep learning models are trained using the data provided by the code described in section 8. The main objective of this section is to study the details of the various models by applying these to well-understood data and testing the analysis methods that will be applied to the real simulations.

As there are many parameters to understand/optimize, the choice was made to first build a simple Fully Connected model that allows to make the first predictions. The model architecture and the hyperparameters are not that relevant in the beginning because those simulations are not meant to make new discoveries.

Additionally, since two detectors were built (see chapter 8), it is relevant to show results for both of them as long as it is not redundant. Hence, the basis of the analysis will be developed using the first detector (the $4 \times 4 \times 4$ detector with photons having fixed energy and that doesn't record the energy per bar, but only the counts). The spectral analysis methods will be developed in section 10 using the second detector which looks more like POLAR.

9.1 Training data

The training data depends on which detector is used (one uses the deposited energy, and the other does not). Here we describe the more complex data, which are being analyzed by the 8×8 detector. Since the detector response varies with the incoming GRB direction and its spectrum, it is important to provide both the rate per bar and the measured energy to the model. As there are 64 bars, the rate is provided to the model as a vector of length 64. On the other hand, the measured energy in a given bar is stored in a 32 logarithmic bin histogram. Consequently, the energy data are represented by a 64×32 array.

The data that are provided to the models need to be reprocessed, for example by normalizing it, in order to simplify the learning process. This procedure is quite standard and in almost any example that one could find online (for instance the MNIST dataset), the data are modified before being provided to the model. Standard techniques are to take the maximal value reached in the dataset (call it M) and divide all the others by that. This way, all the model will experience is values between 0 and 1. This was the first tested normalization method. More formally, if we call c_i the counts in the i^{th} channel, we have:

$$c_{i,norm} = \frac{c_i}{M}$$

9.2 Model definition for localization studies

A basic fully connected neural network was built in order to compare the other models with respect to it, but also to make all analyses in the same way. The architecture of this model is presented in figure 19. There is no particular reason to use this kind of architecture, except that it has worked for the first tests done on these datasets. Note also that all models are initialized using the same random seed to be able to reproduce the results.

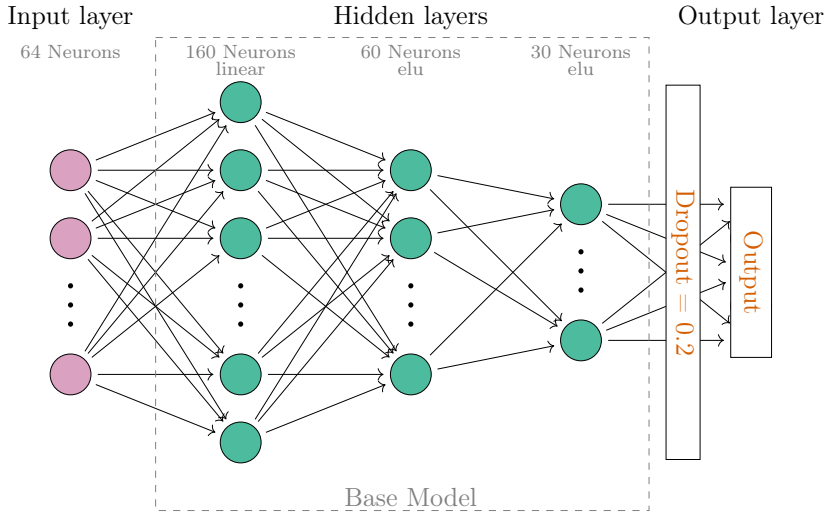


Figure 19: Scheme of the neural network architecture used for the various analyses in this project. The base model is the same for all models used. The only differences are the input and output layers which will vary depending on the model being used

9.3 Target selection

Since the neural network has no physical interpretation by itself, we can define the target the way we want as long as it provides good and physical results. Three possibilities for this have been tested.

θ and ϕ in one array A straightforward choice is to minimize θ and ϕ directly. However, by doing this, the model wouldn't understand that the ϕ coordinate is cyclic (*ie.* that $\phi = 0.01$ is actually close to $\phi = 2\pi$). Therefore, it is a good idea to minimize the angular distance because this measures the physical angular separation between two points on a sphere (see equation 15).

It appears that directly setting the angular distance as the loss leads to predict `NaN` (Not a Number)[15]. Many alternatives were therefore investigated ~~tried~~, but none of these was successful. The decision was made to change the loss because the whole information is contained in the argument of the arccos, which is only useful to convert the scalar product int an angle, but any arbitrary quantity that is small when $x = \cos(\theta_1)\cos(\theta_2) + \sin(\theta_1)\sin(\theta_2) \cdot \cos(\phi_1 - \phi_2)$ is close to 1 (*ie.* the two points are close) and big when x is close to -1 is fine. This motivates the use of the following function :

$$f(x) = e^{\tanh(1)} - e^{\tanh(x)} \tag{26}$$

It will be shown later that using this loss was not enough to get good results. A graphical comparison between f and the angular distance is shown in figure 20.

[15] An hypothesis for this issue is that the arccos has a derivative that diverges at $x = 1$, which is the exact point we want the model converges to.

$\cos(\theta)$, $\cos(\phi)$ and $\sin(\phi)$ in one array In order to get cyclic values (*ie.* to make 2π close to 0) it is convenient to take the sin and the cos of the desired quantities, namely θ and ϕ. However, since the θ angle lies in the range $[0, \pi]$, it is entirely determined by $\cos(\theta)$, hence it is sufficient to set $\cos(\theta)$ as the target. The ϕ value can be obtained from the cos and sin with the `arctan2` function and by taking the modulus to achieve a result between 0 and 2π :

$$\phi = mod(\texttt{arctan2}(\sin(\phi), \cos(\phi)), 2\pi)$$

As a result, a model that tries to fit trigonometric identities has 3 output neurons instead of 2.

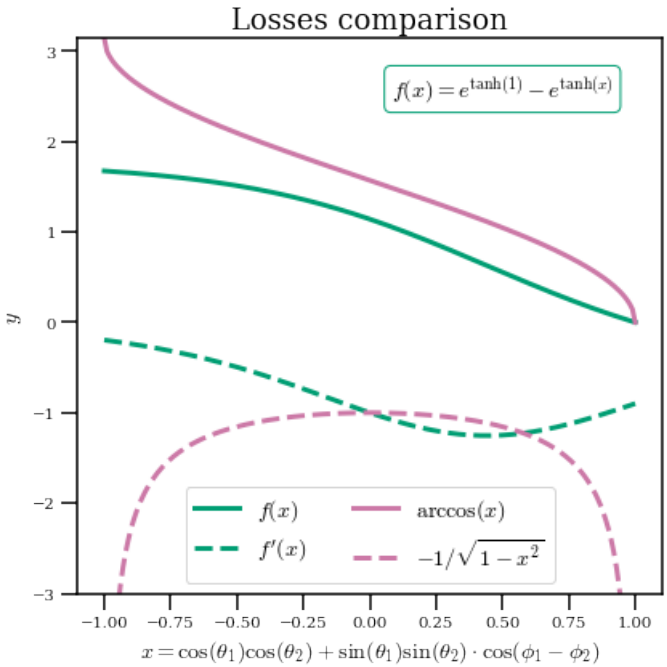
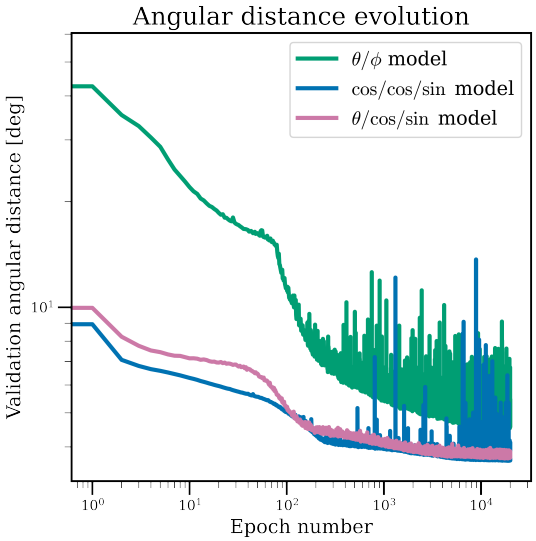


Figure 20: Comparison between the angular distance as defined in equation 15 and the custom loss function as defined in equation 26. Their derivatives are the dashed curves.

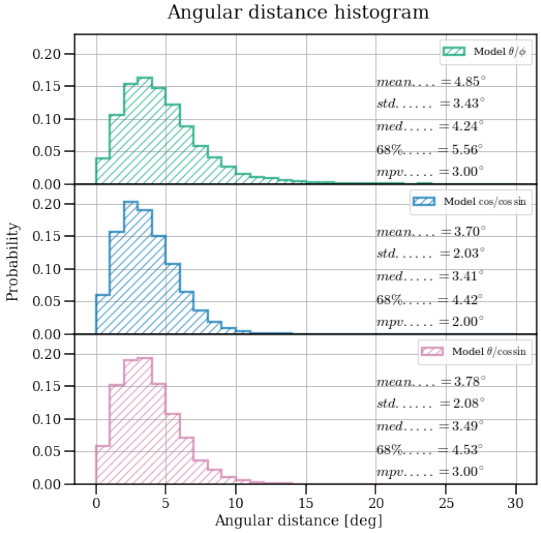
θ, $\cos(\phi)$ and $\sin(\phi)$ in two arrays The architecture of the model is not different than the cos / cos / sin model, except that the two output leads to two output arrays: one for the θ value and one for the $\cos(\phi)$ and $\sin(\phi)$. It will be shown later that the two previous models have some difficulties in accurately analyzing GRBs located at the zenith (with respect to the detector). Having two different outputs for the same network can help to make better predictions since each output can have its own loss. The downside is that the two outputs don't have access to the other, so it forces to use independent losses on each output, like the *mean squared error*.

To know which solution works the best, three different models, all having the same architecture (except the output layer that differs from one to another) have been trained. The training data consists of 200′000 GRB simulations, with 1000 source photons and a signal-to-noise ratio of 3.33 (*i.e.* 300 background photons). The number of epochs has been chosen such that the model converges without overfitting. After each epoch, the mean angular distance was computed on the validation sample, which is very useful to quantify the accuracy of the model even though they have different outputs. It is also useful to verify that the chosen losses effectively minimize the angular distance (which is the desired physical effect). The dependencies of the angular distance as a function of the epoch number are shown in figure 21a. It is clear that the θ/ϕ model is the one that performs the worst. The two other models seem to be better, and the $\theta/\cos/\sin$ model

is the one that has the lowest noise (the fluctuations on the monitored quantities evaluated on the validation sample). A general behaviour that is easily observable here and will also be in the next analysis is the sudden drops in the monitored quantities. The reason for this is thought to be the combination of two elements: the log scale and the `adam` optimizer. This optimizer has the ability to adapt its learning rate, which changes the improvement rate.



(a) Evolution on the mean angular distance between target and predictions on the validation sample for the three different model architectures.



(b) Distribution of the angular distance for the three different models, with some statistical quantities. It should be noted that the MPV (most probable value) depends on the number of bins in the histogram.

Figure 21: Study of the angular distance between target and predictions on the testing sample for the different architecture investigated.

In order to further analyze the results, it is worth considering the distribution of the angular distance between the target and predictions on the validation sample (figure 21b). For each model, the different statistical quantities such as the mean, the standard deviation, and the 68% percentile are shown. As one could imagine, the smaller those values are, the better the model performs. Note however that the most probable value (MPV) is computed using the coordinate of the highest bin, which is not very representative and explains the big difference between the cos / cos / sin model and the θ/ cos / sin one. It is here again evident that the models cos / cos / sin and θ/ cos / sin are the best ones. The reason why the θ/ϕ model performs the worst becomes evident with figure 22 because it tends to predict a wrong location for events near $\phi = 0$. That means that the model didn't properly learn that the data are cyclic, which is the case for the two other models.[16]
Even though the cos / cos / sin model has better statistical values, the preferred model is the θ/ cos / sin model because the noise during the fitting procedure is much lower. Indeed, the noise can be very high and reach values of tens of degrees.

[16]Note that this kind of map allows guessing the shape of the detector because it is harder to predict events coming from the edge of the detector so at $\phi = 45°$. It is also a nice verification that both the simulations and the analysis make sense.

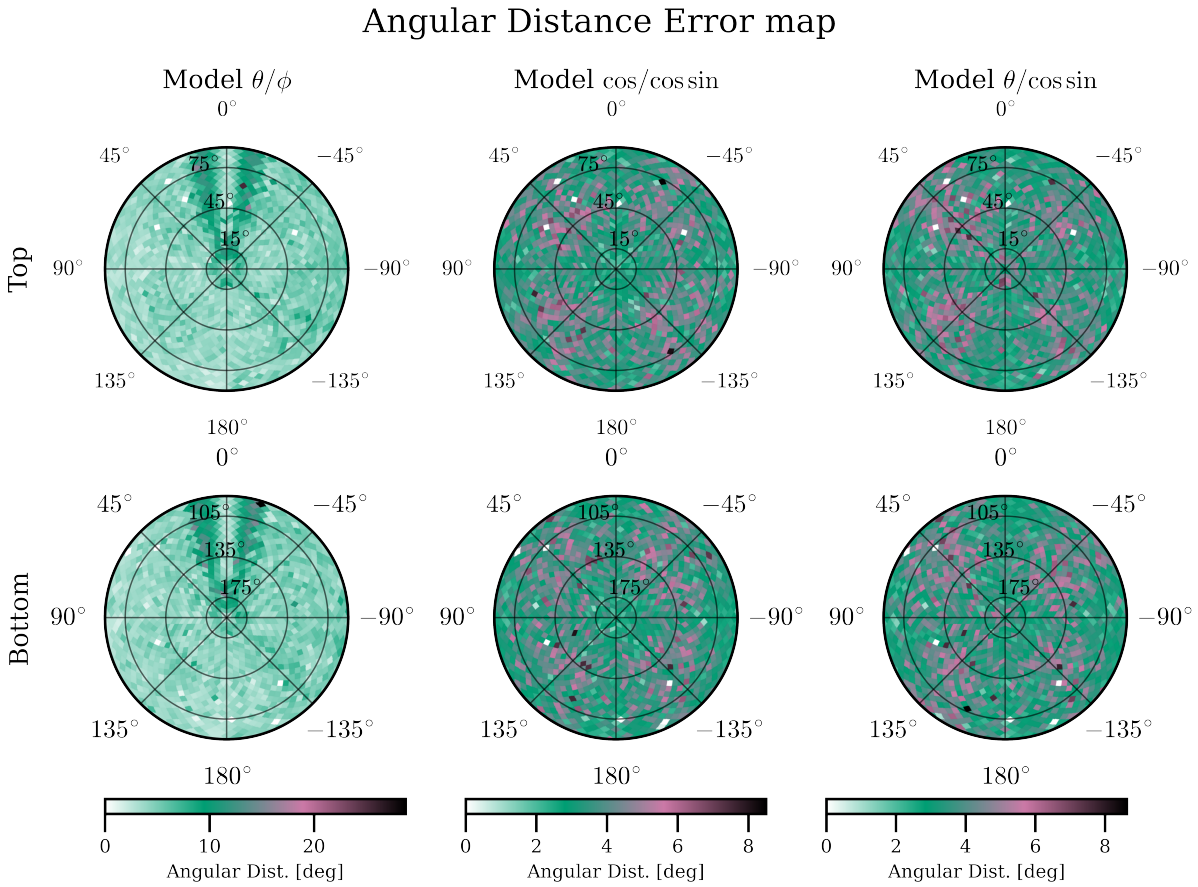


Figure 22: The angular distance between target and prediction as a function of the incoming direction. It is evident that the θ/ϕ model has big difficulties in accurately providing the correct location for GRBs near $\phi = 0$, which justifies the use of trigonometric identities instead. The 'Top' row is for the upper half sky whereas the 'Bottom' one is for the lower half sky.

9.4 Training sample dependence

Once the model is selected, an important thing to know is whether more training data should be generated. For the toy detectors, generating more data is easy but with the POLAR simulations, it takes more time and if the results do increase with the training sample size, it is a simple way to increase the model's performance.

To do so, different training sample sizes are generated and for each one of them, a model is trained. The simulations are the same as the ones used in the previous analysis. Each model is trained for the same number of epochs and the model is saved at the epoch where it performs the best. At the end of the training, the models are evaluated on the test sample, and the performances are computed. The important criteria are the statistical quantities of the angular distance histogram. As it can be seen in figure 23, it is clear that the bigger the training sample, the better the model performs.

As it can be seen, all the statistical quantities seem to be decreasing at the training sample size used in this chapter (*ie.* 138′285), which means that the results could be improved. However, this is not the aim of those simulations and the training size will be the same for all the other analyses done on the toy MC detectors. This analysis will be performed again with the real POLAR simulations since qualitative results will be expected.

Finally, as the training sample should typically contain hundreds of thousands of simulations, it is clear that the analysis of POLAR data needs to rely on simulations, for the simple reason that the 55 detected GRBs won't be enough to train the model.

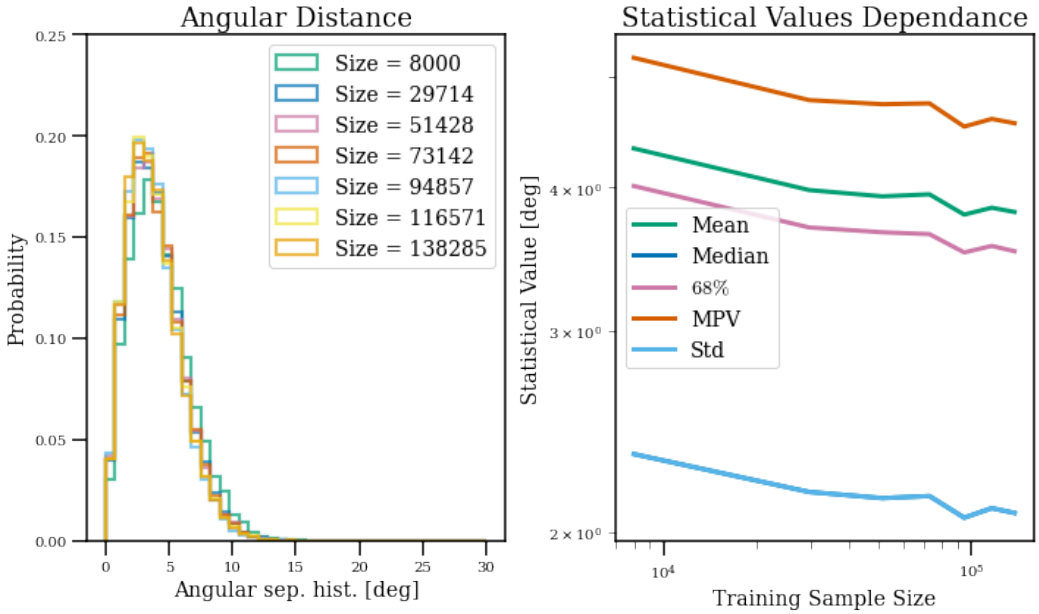
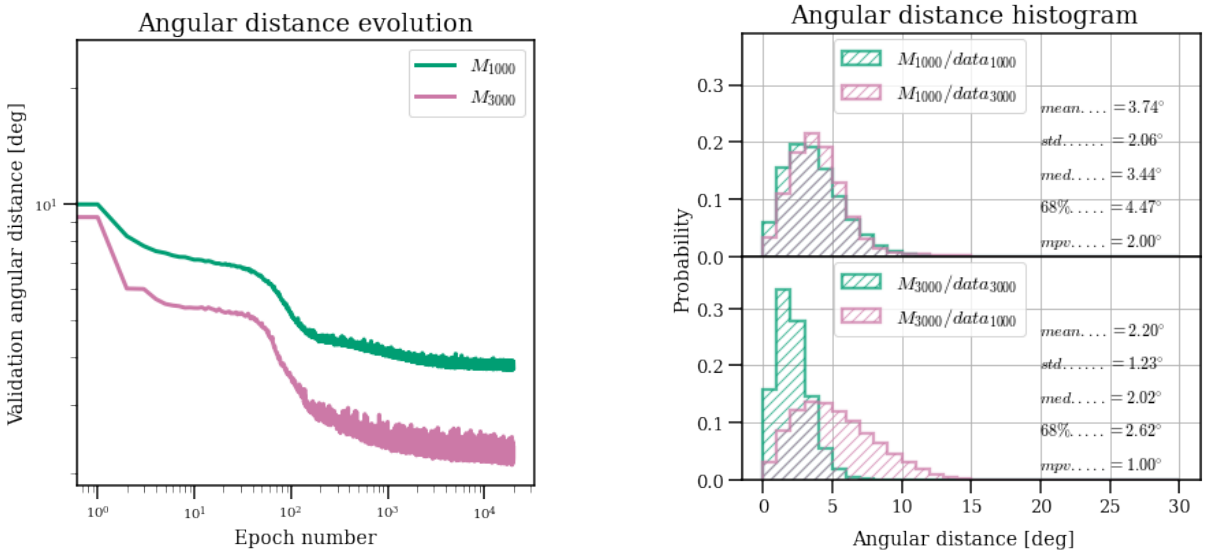


Figure 23: Evolution of the distribution of the angular distance between target and predictions on the testing sample. The bigger the training sample is, the lower the statistical values are. The right graph shows their evolution as a function of the training sample size.

9.5 Signal to noise ratio dependence

As it was already explained in chapter 8.1.3, all the simulations are done with a fixed signal-to-noise ratio in order to be able to study the effects on the model predictions. It is expected that the higher the ratio, the easier it is for the model to make accurate predictions. In a real case, the signal-to-noise ratio is different for each GRB, so it is important to know whether the model can make accurate predictions on different GRB ratios.
To test this dependency two models were trained on different datasets. The first one is trained on data that results from simulations of 200′000 GRBs where 1000 source photons and 300 background photons interact with the detector while in the second dataset, 3000 source photons and 300 background photons interact with the detector. For clarity, let's call M_{1000} the model that has been trained using the first dataset (that we will call $data_{1000}$) and M_{3000} the model that has been trained using the second dataset (whose name will be $data_{3000}$). The background is the same in both cases, which leads to a signal-to-noise ratio of 3.33 in the first case and of 10 in the second one. The evolution of the mean angular distance between the targets and prediction on the validation sample is shown in figure 24a. It is evident that the higher the ratio, the more accurate is the model. This is also verified by the statistical quantities of the distribution of a model (M_{1000} or M_{3000}) on 'its' dataset ($data_{1000}$ or $data_{3000}$). Those statistical quantities are shown in black in figure 24b and refer to the green histograms. Since the two models are trained using a fixed SNR, they should perform worse if a prediction is made on a different dataset. To verify that, it is interesting to make predictions on $data_{1000}$ using M_{3000} and vice-versa. The distributions are shown in magenta in figure 24b. Interestingly, the behavior is very different: the predictions of M_{1000} on $data_{3000}$ (labelled as $M_{1000}/data_{3000}$ on figure 24b) are significantly more accurate than the predictions of M_{3000} on $data_{1000}$.
To be complete, it is also interesting to see where M_{3000} performs the worse. It is also relevant to compare it to predictions and errors of M_{1000} as shown in figure 25. Overall, as one could have expected, GRBs having a higher signal-to-noise ratio are much easier to predict accurately. Additionally, it is important, considering figure 24b that the training sample includes GRBs having a low SNR because they are able to analyze GRBs with a higher SNR quite accurately.



(a) Evolution of the mean angular distance between target and predictions on the validation sample, as a function of the training epoch. It is evident that the SNR has a big impact on the angular distance. The green curve represents the evolution of M_{1000} (trained on $data_{1000}$) and magenta shows the evolution of M_{3000}

(b) Distributions of the angular distance between target and prediction on the test sample. Green shows the distribution for the predictions on data that are similar to the training data (*e.g.* the predictions of M_{1000} on the testing sample of $data_{1000}$). Magenta shows the result of when the trained model is used on the different dataset (*e.g.* the predictions of M_{1000} on the testing sample of $data_{3000}$)

Figure 24: Study of the angular distance for the different signal-to-noise ratios that have been investigated

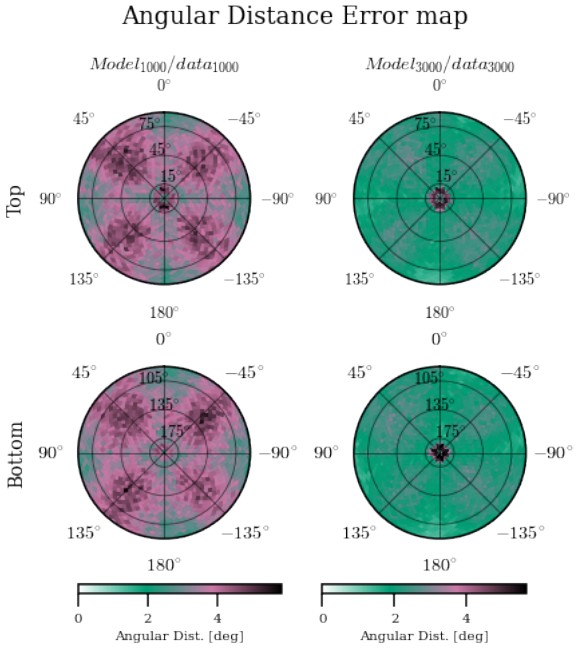
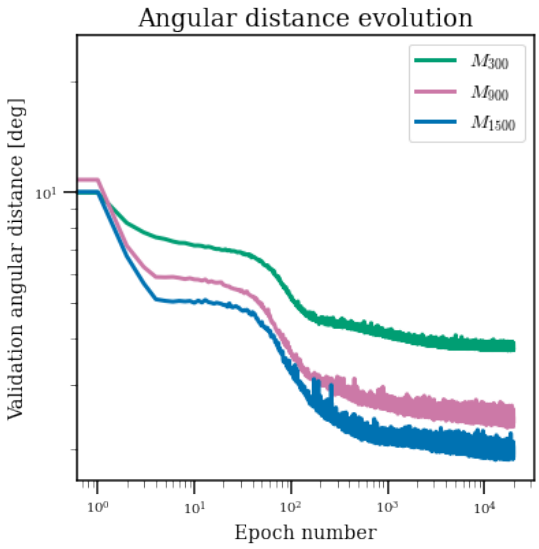


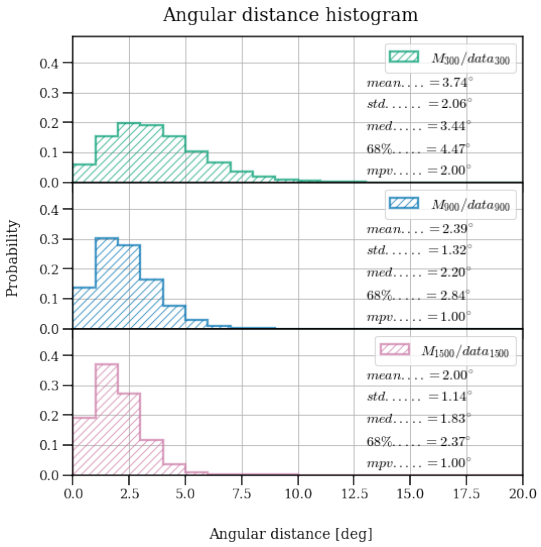
Figure 25: Angular distance between target and prediction as a function of the incoming GRB direction for the two different models M_{1000} and M_{3000}. It is clear that both models have difficulties to predict events at the poles. However, M_{3000} performs best on average. While for the M_{1000} model the poorer performance at ϕ angles corresponding to the edges of the cube is also observed.

9.6 Photon number dependence

Similar to the signal-to-noise analysis, it is important to know how the model reacts to different rates with a fixed signal-to-noise ratio. The reason why this is important is that the GRB prediction will be performed on the integrated counts over the full GRB duration, which means that a very short and bright GRB can have the same number of counts as a longer and fainter one. In this context, it is important to study its impact to ensure that all GRBs are well-predicted. Therefore, three models were trained using 200′000 GRB simulations: the first one with 1000 source photons per simulation, the second one with 3000 GRB photons per simulation, and the last with 5000 GRB photons per simulation. The signal-to-noise ratio is fixed at 3.33 for all simulations.
In order not to create confusion with the previous chapter and to make it easier, the first model will be denoted by M_{300}, the second one M_{900}, and the last one M_{1500}. They have been trained respectively on $data_{300}$, $data_{900}$ and $data_{1500}$. Those names refer to the number of background photons in each simulation.
The evolution of the mean angular distance as a function of the training epoch for the three different models are shown in figure 26a. Without surprise, the model that performs the best is M_{1500}, which was trained on $data_{1500}$, which has the best statistics since for each simulation a total of 6500 photons are interacting, whereas 1300 and 3900 photons are interacting for respectively $data_{300}$ and $data_{900}$.



(a) Evolution of the mean angular distance between target and prediction on the testing sample as a function of the training epoch. The best model is the one trained with $data_{1500}$.



(b) Comparison of the distribution of the angular distance between target and predictions on the testing sample for M_{300} (top), M_{900} (middle), and M_{1500} (bottom)

Figure 26: Study of the dependency of the total number of photons with a fixed SNR.

As before, it is relevant to see where the models have difficulties to predict events accurately. As it can be seen in figure 27, for the three models, events at the poles are difficult to predict.

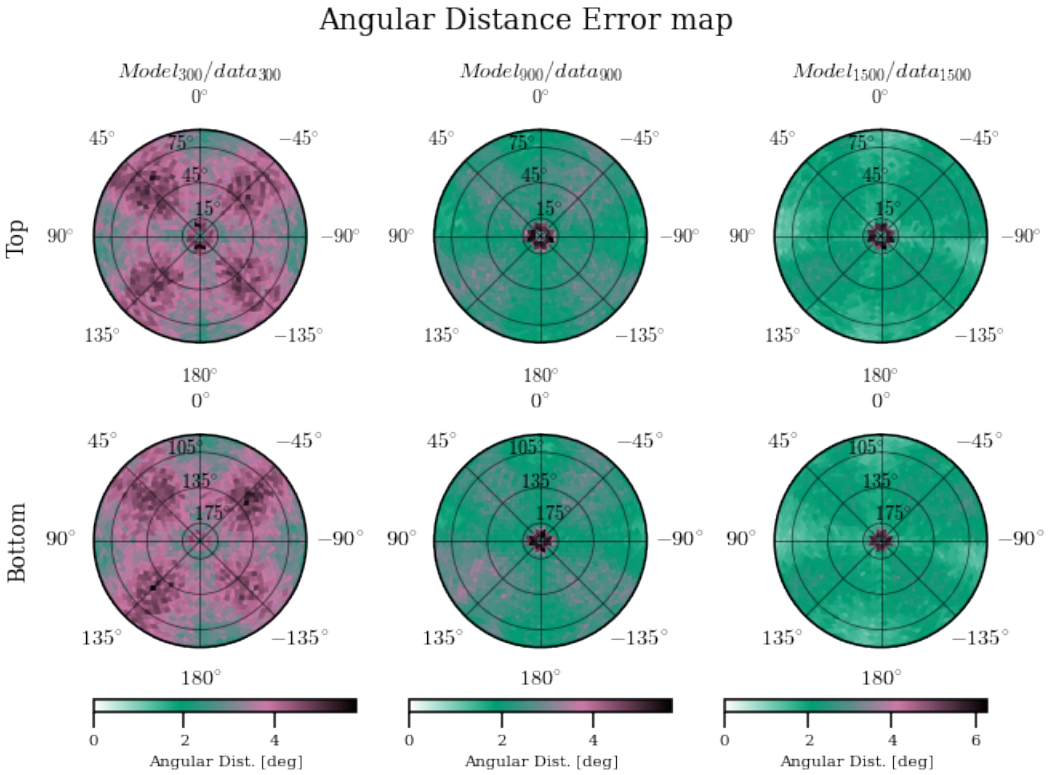


Figure 27: Angular distance between target and prediction as a function of the incoming GRB direction for the three different models M_{300}, M_{900}, and M_{1500}. It is clear that all the models have difficulties to predict events at the poles. The poorer performance at ϕ angles corresponding to the edges of the cube is also observed.

9.7 χ^2 fit comparison

An important thing to do is to compare the results obtained by the deep learning models with the standard method, which is basically the χ^2 fit method which was applied to POLAR data in the past [41]. To do that, a database of 10^6 background photons are computed, as well as a database of the counts per bar for 3072 GRBs, one for each `Healpix` bin. `Healpix` bins are regions in the sky defined in such a way that they all have the same surface. They are derived from spherical harmonics and are a common use in astrophysics, because they are defined the same way for everyone [19]. Each simulation computes the counts per bar for $5{\cdot}10^4$ GRB photons (as we want the detector responses to be free of statistical uncertainties). Then, given a ratio, it is possible, using `ROOT`[17] to add two histograms with respective weights. The first histogram is the counts per bar due to the background photons. The second one is the counts per bar due to GRB photons coming from a given `Healpix bin`, without background. Since the SNR ratio is known, it is possible to compute the weights accurately, using the number of photons from the background (n_{BG}) and from the GRB (n_{GRB}):

$$w_{BG} = \frac{n_{BG}}{n_{BG} + n_{GRB}} \qquad w_{GRB} = \frac{n_{GRB}}{n_{BG} + n_{GRB}}$$

Once it is done for all `Healpix` bins, the result is a database of detector responses for each healpix bin. Any detector response can then be compared to those expectations with the χ^2 test function `Chi2Test` function. The database element that minimizes the χ^2 value is the best fit and thus gives a location. This operation was done on a testing sample of 20′000 GRB simulations with different background values to check the dependency of this quantity on the predictions. In all

[17] The useful function to do that is `Add`

cases, the SNR ratio is fixed at = 3.33. The knowledge of the best fit and the true incoming direction makes it possible to draw the distribution of the angular distance between those two quantities. Those histograms are shown in figure 28 for the different datasets.
The deep learning models perform better since the corresponding distributions are more centred than the ones for the χ^2 method. This is crucial because it motivates the use of such deep learning models to make localization analyses on POLAR data.
It is also interesting to know where the χ^2 method has difficulties to make accurate predictions, which is shown in figure 29. Two things are to be noted. The first one is that the general shape is the same as the one obtained with the deep learning models (see figure 27 for instance), which means that both the simulations and the analysis make sense. The second thing to note is that the χ^2 is more accurate at the poles than the deep learning models, indicating that this inaccuracy is an artifact of the deep learning method.

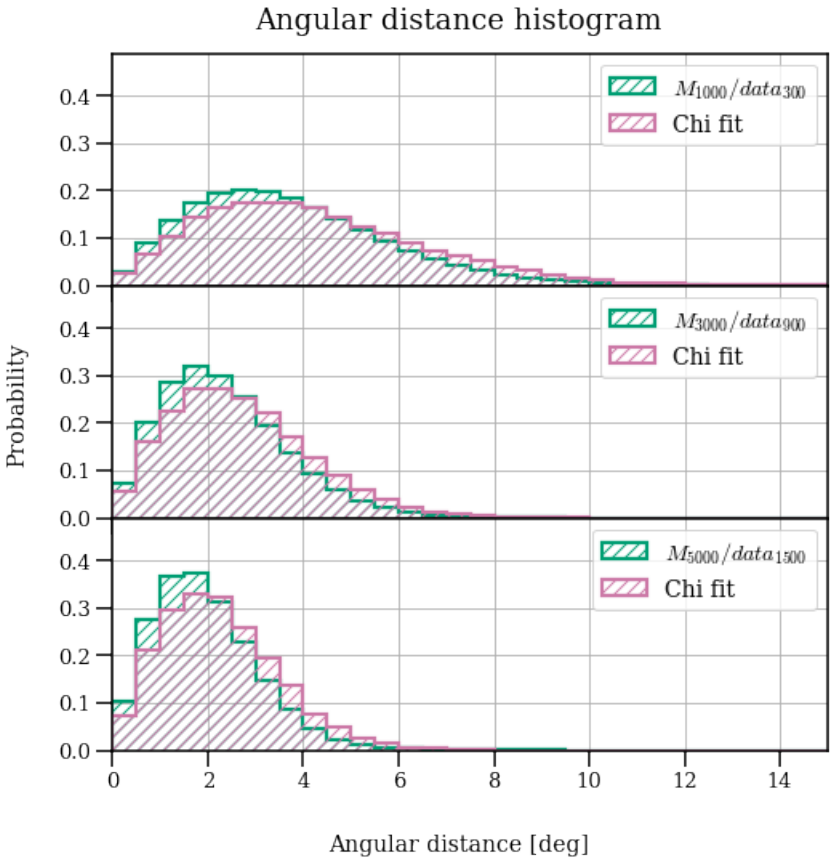


Figure 28: Comparison of the angular distance between target and prediction for the deep learning models (in green) and the standard χ^2 fit method (in magenta). The deep learning models perform better for the 3 different signal-to-noise ratio samples it was tested on here.

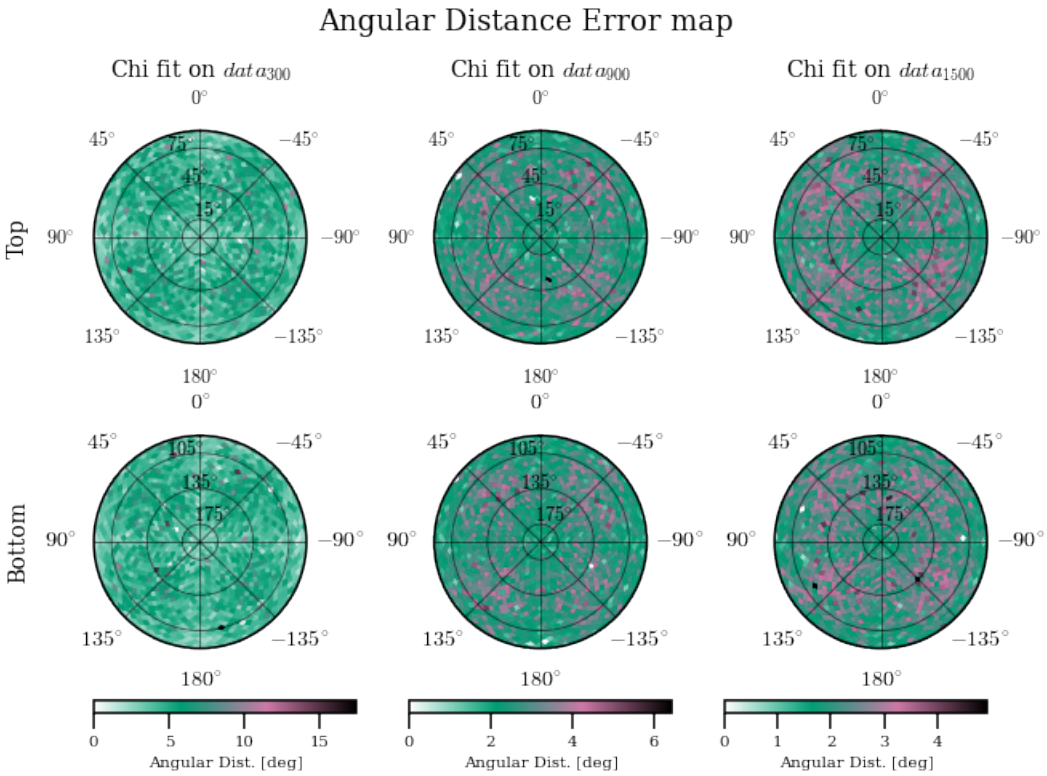


Figure 29: Map of the angular distance as a function of the incoming direction for the χ^2 method and different datasets.

10 Spectral analysis with the toy detector

This section describes the analysis methods for the toy detector that looks more like POLAR. Precise differences with respect to the $4 \times 4 \times 4$ detector are explained in chapter 8, but the biggest one is that in this case photon energies are recorded and distributed as a power law for the background photons and as a smooth broken power law for GRB photons.
200′000 GRBs were simulated for this analysis, and the distributions of the GRB parameters are shown in figure 30. Each GRB simulation is the result of the interaction of 3000 GRB photons and 900 background photons. The count normalization being used is the same as for the first toy detector. For the energy, the method is the same: all values in the dataset are divided by the maximal one.

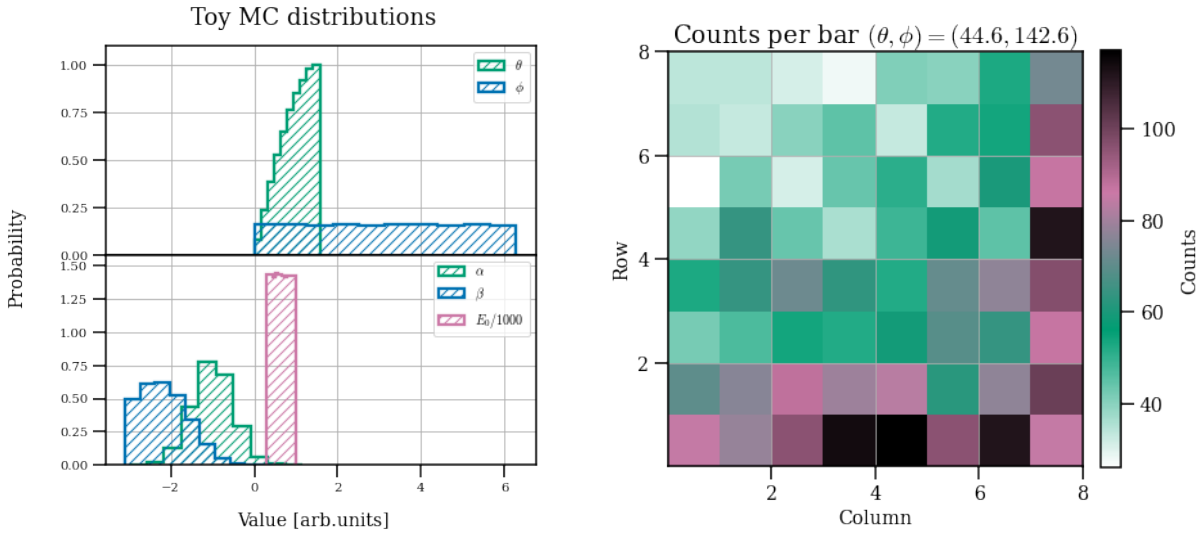


Figure 30: Left: Distribution of the different GRB properties as simulated in the toy Monte Carlo. The ratio is fixed and the spectral parameter are chosen such that they look like the real POLAR simulations. Right: Typical hitmap resulting from a GRB simulation.

Even though the full analysis (meaning localization and spectral analysis) could be done using a single model with multiple outputs, it was decided that the spectral analysis will be performed by a separate model to have more flexibility. Since the data are generated differently, it is also interesting to see how the localization prediction changes with a more complex dataset, however, no further localization analysis will be performed in this chapter, however, the effect of the new data format used for spectral analysis on the localization is studied. In terms of model architecture, there are major differences since the detector records the energy. As the localization depends on both the incoming direction and the spectrum, it is important to provide the counts (64 numbers) but also the energies (64 histogram containing 32 bins) to the model. In order to keep the number of neurons relatively low, a few layers were added just after the inputs, as shown in figure 31. The rest of the model is the same as in section 9.

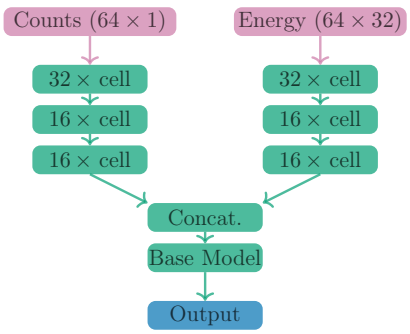
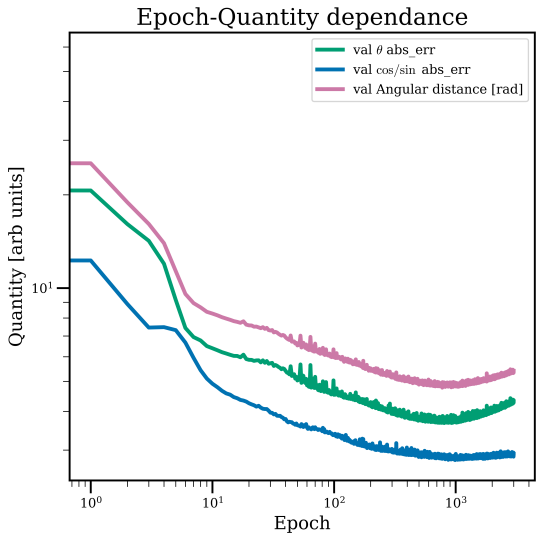


Figure 31: Architecture of the model taking into account both the counts per bar and the deposited energy per bar. A few `Dense` layers have been added to keep the number of neurons low.

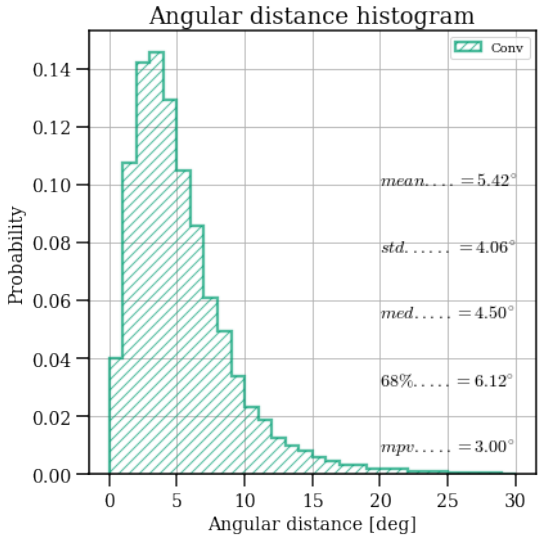
On the other hand, for the model that does the spectral predictions it is irrelevant to provide more than the measured energies to the model. In order to make the model converge faster (and it also showed better results) the energy bins are all summed such that the model's input is a single 32 bin histogram. The output of the model that predicts the spectral parameters will be detailed later in this chapter.

10.1 Localization results

The evolution of the mean absolute error of those quantities over the epoch is shown in figure 32a, which shows that the model has converged.



(a) Evolution of the monitored quantities over the training epochs. It is clear that the model shows some clear overfit signs from epoch $\sim$ 9000, but the model has been saved where the mean angular distance between the target and predictions on the validation sample is at its lowest.

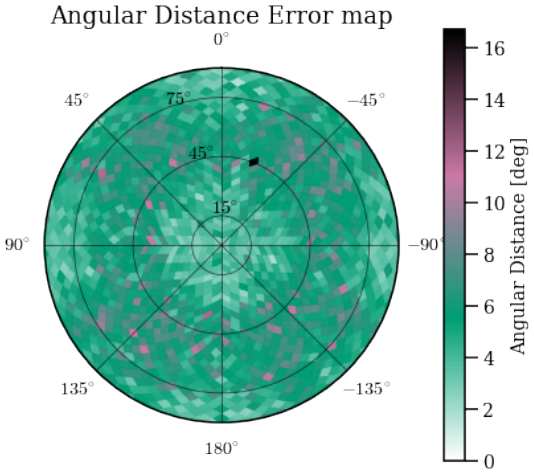


(b) Distribution of the angular distance between target and predictions on the testing sample. It should be noted that the distribution is wider than the one shown in figure 24b

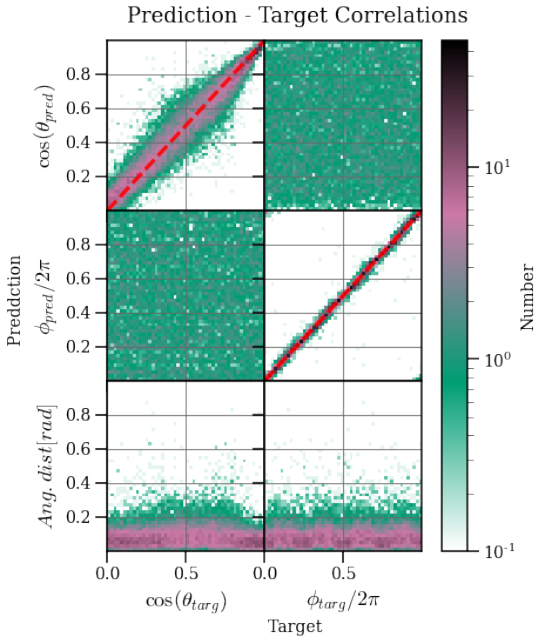
Figure 32: Analysis of the distribution of the angular distance between target and predictions.

As in chapter 9, the performances of a model are determined by the distribution of the angular distance between the target and predictions on the testing sample. It is also important to know where the model has difficulties to predict accurately the events. Those two analyses are shown respectively in figures 32b and 33a.
The fact that the distribution is broader than the one shown in figure 26b for M_{900} shows that the dataset is more complex. It should be noted that contrary to the model studied in section 8, this one doesn't show any special difficulties to predict events at the zenith (a hot pixel is to be noted). This is thought to be a result of a difference in the geometry. Since the two detectors used for the two studies are different, they have different sensitivities. Interestingly enough is the fact that the shape of the detector can also be guessed from figure 33a, because it is harder to accurately locate events coming from the edge of the detector.

Another way to characterize the model performances is by studying the correlations between the predicted θ and ϕ values and the angular distance. This is what figure 33b shows. Note that both the predictions and targets have been normalized to one to avoid binning effects. The $\pi/2$ periodicity can also be seen on the bottom right graph, which represents the angular distance as a function of ϕ.
Overall, the method used to predict the location of the incoming GRB seems to be good. The fact both the counts per bar and the energy histograms constitute the model's input helps to accurately predict the GRB location. This method will hence be applied to the real POLAR simulations.



(a) Polar histogram of the mean angular distance between target and predictions on the testing sample as a function of the incoming location.



(b) Correlations between the targets and the predictions on the testing sample. Quantities labelled by "t" are the targets whereas the "p" index refers to the predicted values. The red dashed lines represent the ideal case.

Figure 33: Localization analysis for the model that takes the energy into account.

10.2 Spectral fit

The targets for this study are obviously the three Band parameters. However, the values of these 3 are very different since β is often negative while E_0 can be as big as a few hundred (see figure 30). It is therefore a good idea to transform those values such that it is easier for the model to converge. More precisely, those target values must be in the range [0, 1], which motivates the following transformations:

$$\begin{aligned} \alpha_{target} &= -\alpha + 1 \\ \beta_{target} &= \beta + 3.1 \\ E_{0,target} &= \frac{E_0 - 100}{1000} \end{aligned} \tag{27}$$

With these transformations, all target values are in the desired range, which makes it possible to use the `ReLU` activation function (because it only outputs positive numbers). Additionally, since there are some physical constraints on the parameter ($\alpha > \beta$ by definition), it is crucial that the model learns them properly. Two options are available :

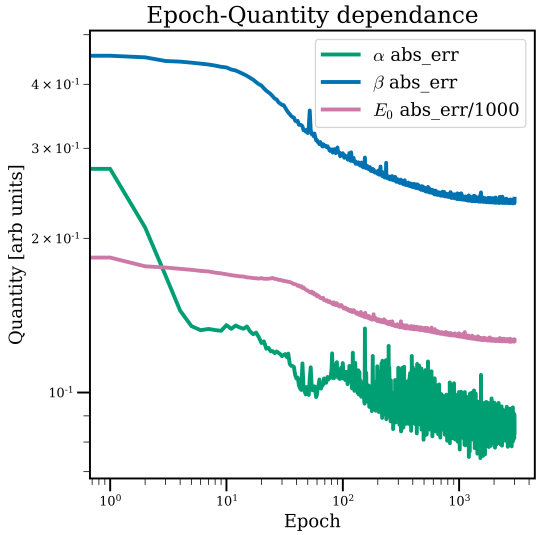
1) Each Band parameter has its own independent output and the model learns the constraints by itself.

2) The Band parameters are fitted together and a custom loss is defined such that it applies the mean squared error on each parameter but penalizes the model when it predicts a set (α, β, E_0) that has no physical meaning.

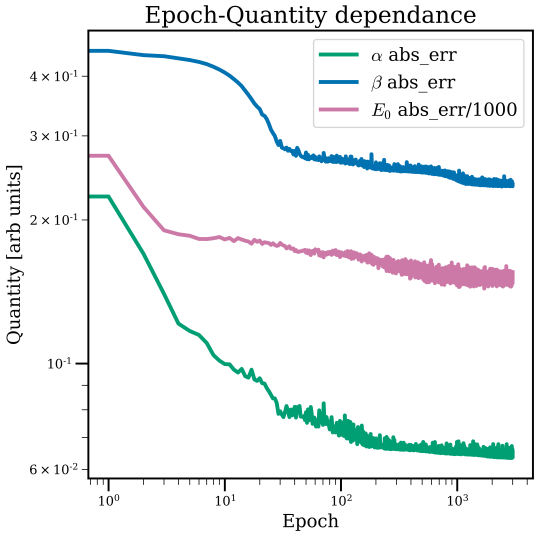
In the first case, the loss can simply be the mean squared error as it showed great performances for the localization analysis. Since the three parameters outputs are independent, let's call this model M_{sep}. For the second case, the loss is very similar to the mean squared error, but it adds an extra penalty of 10 times the difference between α and β (if $\beta > \alpha$). It is also a good idea to reduce the loss in some cases due to the intrinsic degeneracy of the Band function. Indeed, if the α and β values are very close to one another, the produced spectrum is very similar to a power spectrum, in which case E_0 has no meaning. It is therefore a good idea to weight the mean squared error on E_0 by the difference $\alpha - \beta$ (only if $\alpha > \beta$). The desired effect is that the closer the spectral indices are to each other, the less impact the error on E_0 has on the loss. This model will be denoted by M_{block} since it outputs a block containing the three Band parameters.

To compare the two methods, the two models were built and trained using the same data as for the localization (see 10.1). At each epoch, the mean absolute error is monitored on each spectral parameter. The results are shown in figure 34a and 34b (respectively M_{block} and M_{sep}).
The first thing to be noted is that the model M_{block} seems to show a smaller mean absolute error for all parameters. However, the mean error on the α parameter is noisier than for model M_{sep} and even shows a period of overfitting that coincides with the one where the mean error on E_0 starts to decrease. For both models, it should be noted that the β mean error starts to decrease only when the α mean error reaches a plateau phase.

Similar to the localization analysis, the model has to be evaluated on its predictions on the testing sample. As it was seen in chapter 10.1, the correlations between the target parameters are a good way to characterize it. Hence, those are shown in figure 35a and 35b. The red lines are the ideal cases where the model predicts the target. Since the ranges are different for each subplot, the bin content has been normalized such that they all have the same range.
The first element one can notice is that even though the evolution of the monitored quantities is relatively different for the two models, the two correlation analysis looks fairly similar (it is not even easy to spot the differences). Furthermore, the α parameter is the easiest to predict since it almost follows the red dashed line in both cases. This accuracy is due to the higher



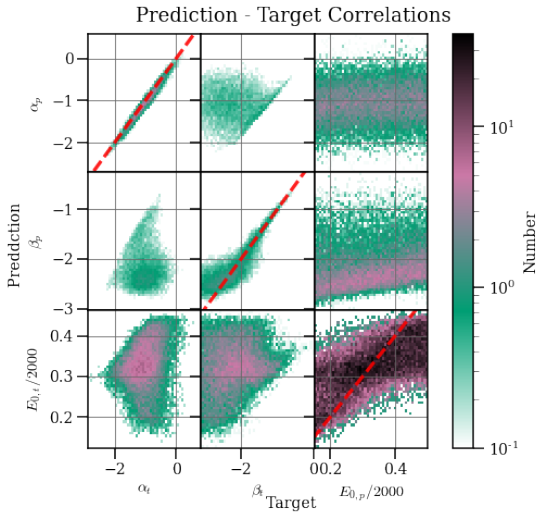
(a) Evolution of the mean absolute error between target and predictions on the validation sample on the three Band parameters for the model M_{block}. The loss applied here penalizes the model for non-physical predictions.



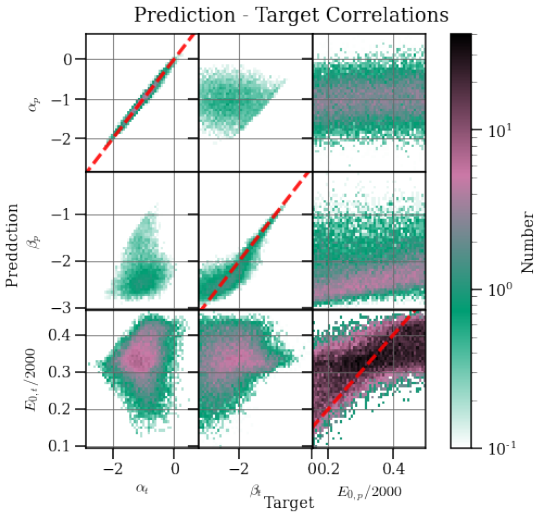
(b) Evolution of the mean absolute error between target and predictions on the validation sample on the three Band parameters for the model M_{sep}. The model has three outputs having the mean squared error as a loss.

Figure 34: Comparison of the evolution of the mean absolute error for the three Band parameters that the two different models are fitting.

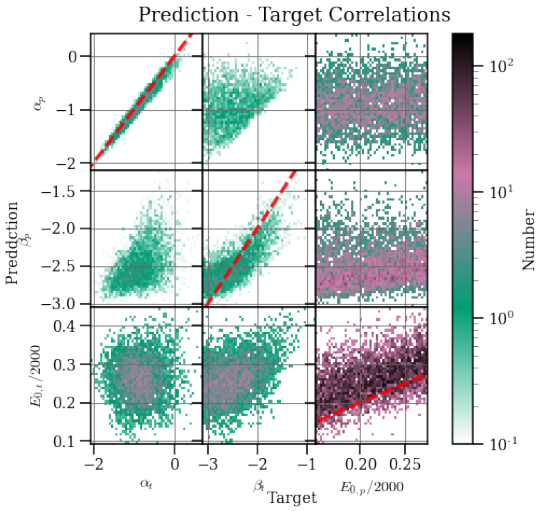
detector sensitivity of the detector in this range. On the other hand, the β parameter is harder to predict, especially for very low values. The reason could be that with a very low β parameter, the spectrum drops very quickly and so does the number of photons available for the analysis. The very important element to note is that both models always predict $\alpha > \beta$ as can be seen in both the top middle graph and the middle left one. Besides that, it is clear that the E_0 parameter is the hardest to predict. This difficulty is thought to be caused by the fact that E_0 has no meaning if α and β are very close because the resulting spectrum would essentially be a power-law spectrum. It furthermore makes no sense to predict an E_0 value that is far beyond the sensitivity region of the detector.
To verify what we see for E_0, a filter was applied to the testing sample, to select events where $|\alpha - \beta| \geq 1$ and $E_0 < 550\,keV$. The result is shown in figure 35c (using the model M_{sep}). There is no difference to be noted on the α parameter, and very little on β (very low values of β are still hard to predict). On the other hand, the tendency for the E_0 prediction seems to be better in this case, which confirms the hypothesis of the previous paragraph.
After a lot of tries and verification, it was decided that a good moment to save the model is the epoch at which the mean absolute error is at its lowest. All the models in this section and the following have the same criterion.



(a) Correlations between targets and predictions on the testing sample with the M_{block} model.



(b) Correlations between targets and predictions on the testing sample with the M_{sep} model.



(c) Correlations between targets and predictions on the testing sample done by the model M_{sep}. A filter was applied to take only the events with $|\alpha - \beta| \geq 1$ and $E_0 < 550\,keV$. The E_0 predictions seem to be better in this case, whereas the α and β predictions are similar.

Figure 35: Correlations between targets and predictions on the testing sample for the two models that were investigated. The bin content has been normalized such that they all have the same range

11 GRB detection

The first challenge one encounters when having a large field of view detector is to detect the transient burst. For this work these studies were performed chronologically after the localization studies provided good results, thereby motivating the choice to detect GRBs in the lightcurves using deep learning. This way, the analysis process includes both the detection and the subsequent analysis.

The task the models will be trained on is, given the 60 past seconds of counts and energy per module, to predict if a GRB is happening at time $t = 0$ or not.

11.1 Training data

The training data were already available from the GRB simulations since each one of them is essentially a database from background and GRB photons coming from a specific direction. The training data in this case consists of lightcurves of 60 seconds. 50% of the training sample has a GRB at $t = 0$, and the rest is composed of pure background (25%) and GRBs that are not happening at $t = 0$ (25%). This way, the model trains on various types of signals.
To build the lightcurve, the background rate is simulated by a third-order polynomial. On the other hand, as there exist many GRB shapes. Not only does the length of the GRB differ by many orders of magnitude, but also the number of pulses in a GRB and the time and relative height of these vary significantly. Therefore a lot of GRBs must be simulated. To do so, the function `norris` from the `cosmogrb` framework [9] was used and modified to create more realistic GRB shapes. This function describes the shape of the ideal, FRED (Fast Rising Exponential Decay) like, GRB with a few parameters: t_{start}, t_{rise}, K and t_{decay}:

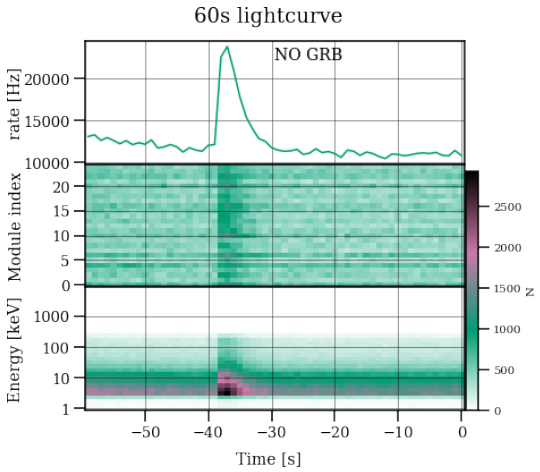
$$f(x,\, t) = \begin{cases} K \cdot \exp\left(2\sqrt{t_{rise}/t_{decay}}\right) \cdot \exp\left(-\frac{t_{rise}}{x - t_{start}} - \frac{x - t_{start}}{t_{decay}}\right) & \text{if} \quad x \geq t_{start} \\ 0 & \text{else} \end{cases} \tag{28}$$

As true GRBs can be composed of several sub-pulses, a simulated GRB is defined as a sum of N_p functions (one for each pulse). The distribution of pulse number was here chosen to follow a Poisson distribution with a mean of 1: $N_p \sim \mathcal{P}(1)$. The time parameters of the functions are adapted such that the full duration of the GRB roughly follows the log-normal distribution of T_{90} as described in 3.

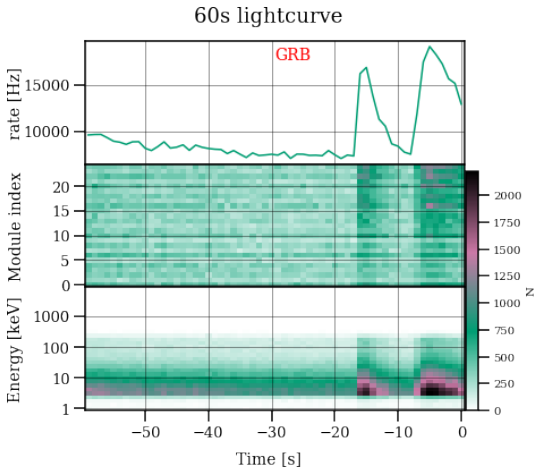
Since the model must learn when a GRB is happening, an event (60 seconds of lightcurve) is labelled as 1 if the GRB is happening at $t = 0$, while the event is labelled as 0 if the GRB happens anywhere else (or if the lightcurve only consists of background). Note that even if there was a GRB during the last 60 seconds, the model should not predict 1 unless the GRB is still happening.

For faster training, it was decided to use the counts and energy histogram grouped by the detector module (so reducing the number of channels by a factor of 25). As a result, the files containing the $131'582$ lightcurves are smaller and the training time is drastically reduced[18].

[18] Note that this work was done at the very end of the master's work. This choice was hence motivated by intuition gained during experimenting various models trained on similar datasets.



(a) Simulation of an event without any GRB at the end of the lightcurve this event is labelled as $'0'$.



(b) Simulation of an event with a big GRB at the end of the lightcurve this event is labelled as $'1'$.

Figure 36: Evolution of the monitored quantities as a function of the training epoch for FC NN and the $2D$ Conv.

11.2 Data normalization

Since a GRB can be detected at any time, any rate of background should be simulated. A background rate distribution that reflects the real one is better, and in this case, energy follows a power spectrum (which was verified by analysing a few days of real data.). Additionally, the model should be as sensitive in a low background rate period as in a high rate background period, which implies that the data have to be centred and normalized. To do so, a first-order polynomial was used to catch the general background time dependency. The slope and the intercept are inferred by taking the mean rate (see figure 37) in the first and last 10 seconds of the event (respectively m_{55} and m_5). The first-order approximation hence reads:

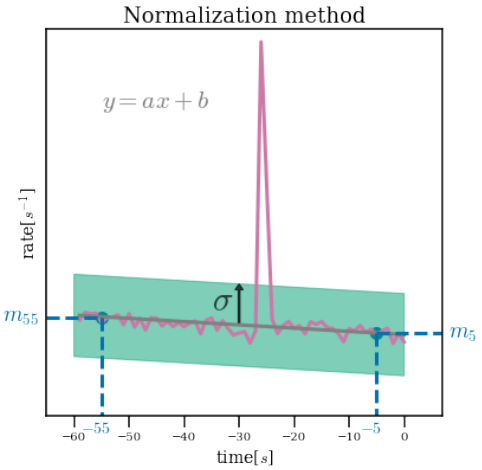


Figure 37: Scheme describing how the data are centred and normalized.

$$y(x,\, t) \approx ax + b \quad \text{where} \quad \begin{cases} a &= \frac{m_{55} - m_5}{55 - 5} \\ b &= m_{55} - 5a \end{cases} \tag{29}$$

The last step is to centre and normalize the signal (*ie.* subtracting the mean μ_c and dividing by the standard deviation σ_c):

$$c_{i,norm} = \frac{c_i - \mu_c}{\sigma_c}$$

Note that both the counts per module and the energies per module are normalized this way.

11.3 Analysis

For this task, different models will be used and the best of them will be selected. The quality criterion will be the area under the ROC (Receiver Operating Characteristic) described in detail below, which is a commonly used criterion in such analyses, and one often refers to it as "AUC" (Area Under the Curve).

ROC curve This curve allows to compare the results of a binary classifier. A standard way to do it is to set a threshold above which all events are classified as 1 and below as 0[19].
Then, given the predictions and true labels, it is possible to build a *confusion matrix* that compares the number of true positives (the model makes a right prediction on a 1 event), true negatives (the model makes a right prediction on a 0 event), false positive (the model predicts a 1 instead of a 0) and false negatives (the model predicts a 0 instead of a 1). However, as any threshold between 0 and 1 can be set, it can be confusing to distinguish which threshold is the best. The ROC curve makes it easy to compare all thresholds at once [35].
For a given threshold, two values are computed: the true positive rate and the false positive rate. Those values are defined by :

$$\begin{aligned} \text{True Positive Rate} &= \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \\ \text{False Positive Rate} &= \frac{\text{False Positive}}{\text{False Positive} + \text{True Negative}} \end{aligned} \tag{30}$$

Note that in mathematics, one could refer to those values as the *sensitivity* and $1-$*specificity*. Nevertheless, it gives crucial information on how the model performs and especially if it manages to predict events accurately (high true positive rate), without too many errors (low false positive rate). When performing this operation for a range of different threshold values and plotting the true positive rate as a function of the false positive rate, one obtains a ROC curve. This allows to easily identify which threshold offers the best-desired performances. Note that the choice highly depends on the task which, in the context of GRB detection, is that no GRB is missed [20].

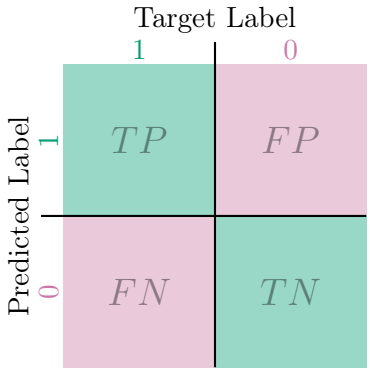


Figure 38: Confusion matrix to evaluate predictions. Elements in the main diagonal are events that are correctly predicted.

[19] Remember, in a binary classifier, the predicted value is a probability.
[20] This situation is similar to rare and dangerous diseases like Ebola. It is not a big deal if a predicted carrier case has not the disease but if one has it, it is critical to detect it.

11.3.1 Models

As explained earlier, different types of neural networks were investigated: Fully connected neural networks, Recurrent neural networks (RNN) and Long Short Term Memory (LSTM). The RNN and LSTM models are famous for their high capabilities in analysing time series. In all cases, the model architecture is the same, it is just the type of layer that changes. The models have two inputs: the counts per module and the energy histograms per module. The output is a single neuron with the `sigmoid` activation function which represents the confidence level that a prediction is a GRB. In all cases, the loss is set to be the binary crossentropy (see 6.0.1). The common architecture is shown in figure 39. The reason why the inputs are both the energies and the counts is that it is important that the model learns that a GRB leads typically to higher energy depositions, so it doesn't predict a GRB for low energetic events.

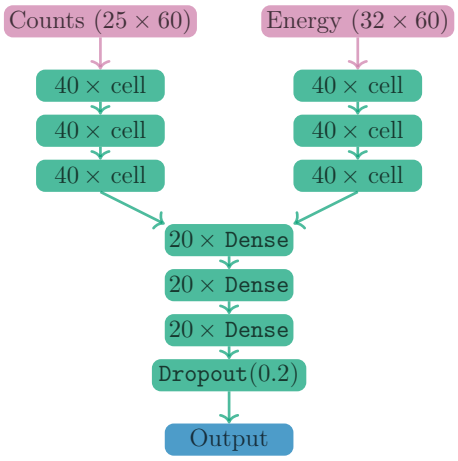


Figure 39: Common architecture for all the models that have been investigated. The "cell" refers to the type of network being tested *ie.* FC, RNN or LSTM.

11.3.2 Results

After some tests, it was clear that the models were converging at different speeds, so a *Checkpoint*[21] was used to assure storing the best validation accuracy. The evolution of the accuracy evaluated on the validation sample is shown in figure 40. It is easy to see from this figure that due to its higher accuracy on the validation sample, the model that performs the best is the LSTM one, which confirms the fact that those kinds of models show great efficiency for time series.

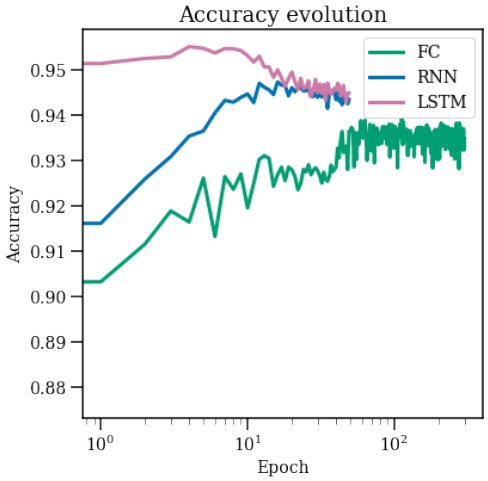


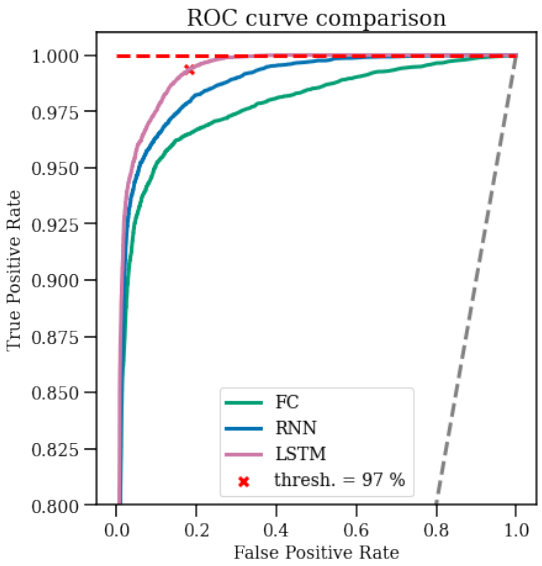
Figure 40: Evolution of the accuracy evaluated on the validation sample as a function of the training epoch. Some models are trained for too long, which leads to overfitting signs. However, the models have been saved at the epoch where the accuracy is the highest.

However, as mentioned above, the accuracy is not a sufficient criterion to fully characterize the models, it is important to build the ROC curves. Those are shown in figure 41a, and it is clear that the model that performs the best is the LSTM one due to the highest AUC (being 0.940). From this graph, it seems that to keep a true positive rate high while having a relatively high false positive rate, a threshold of ~ 0.97. Indeed, this way a good proportion of GRBs are predicted as so, and the proportion of undetected GRBs stays relatively low.

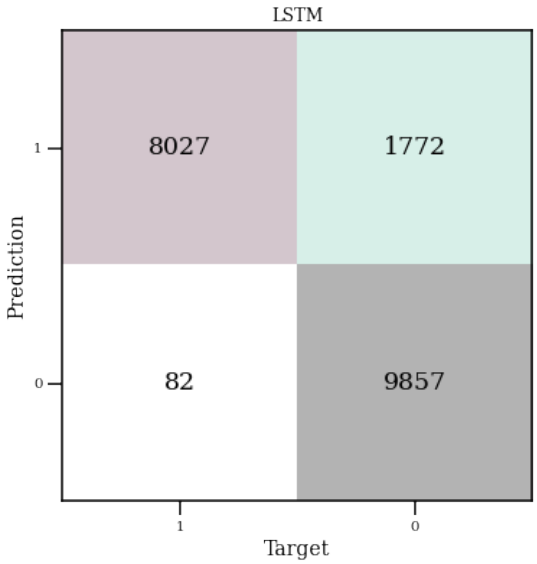
It is expected that once applied to real data, a lot of false positives will be predicted, because the real background contains events that haven't been simulated, like solar flares, particle-induced events (*ie.* when clouds of trapped charged particles from space hit the detector and are suddenly triggering bars.), and even unknown events. Also, the POLAR detector was turned off as it passed in the South Atlantic Anomaly (SAA) as the

[21] A useful `Tensorflow` class that allows saving the model when it performs the best.

rate increases dramatically, which will cause a lot of false positives. Therefore, some filters will be applied to reduce the number of uninteresting events. This complete analysis (detection, localization and spectral analysis) is performed in chapter 13.



(a) ROC curves for the three different models that have been investigated. The best one is the LSTM, having an AUC of 0.940 (and respectively 0.897 and 0.929 for the FC and RNN). The grey dashed line represents the case where the model randomly predicts 1 or 0 (a random classifier).



(b) Confusion matrix for the LSTM model, using a threshold of 0.97. As can be seen, very few GRB events were classified as non-GRB events.

Figure 41: Detailed comparison between the three investigated models, and the corresponding confusion matrix (for the LSTM model).

12 POLAR Analysis

In this section, the methods developed in sections 9 and 10 will be applied to data produced using the official POLAR simulations. As those datasets have much more complications than the self-made simulations (for example due to the effects of electronics), a verification of the methods will be applied to the POLAR simulation of the unrealistic case where all the GRBs have the same energy spectrum (but a varying SNR). The objective is to verify the methods, but also to further develop them in order to improve the results. More specifically, different model architectures will be compared in order to provide the best results, even though no real hyperparameter optimization will be performed.
After this preliminary verification, the full analysis will be performed and tested on real GRB data. The obtained predictions will be compared to the best-fit localization from POLAR to evaluate the possibility to deploy the algorithm onboard the CSS.

12.1 Data

The official POLAR simulation software was used to simulate GRBs with various signal-to-noise ratios, and different spectral parameters (for the complete analysis only). In order for the simulations to be as close as possible to the real data, the background is taken from POLAR flight data, which means that they include hot pixels (*ie.* bars that suddenly record a very high rate). The distributions of the 156′212 GRB parameters for the full analysis are shown in figure 42. It should be noted that in this case, the E_0 values are simulated up to $2000\,keV$. The distribution for the constant spectrum analysis is the same, except that the spectrum is fixed (there are 50′376 such simulations). Note that the distribution of GRBs in the sky is isotropic, which explains the θ distribution.

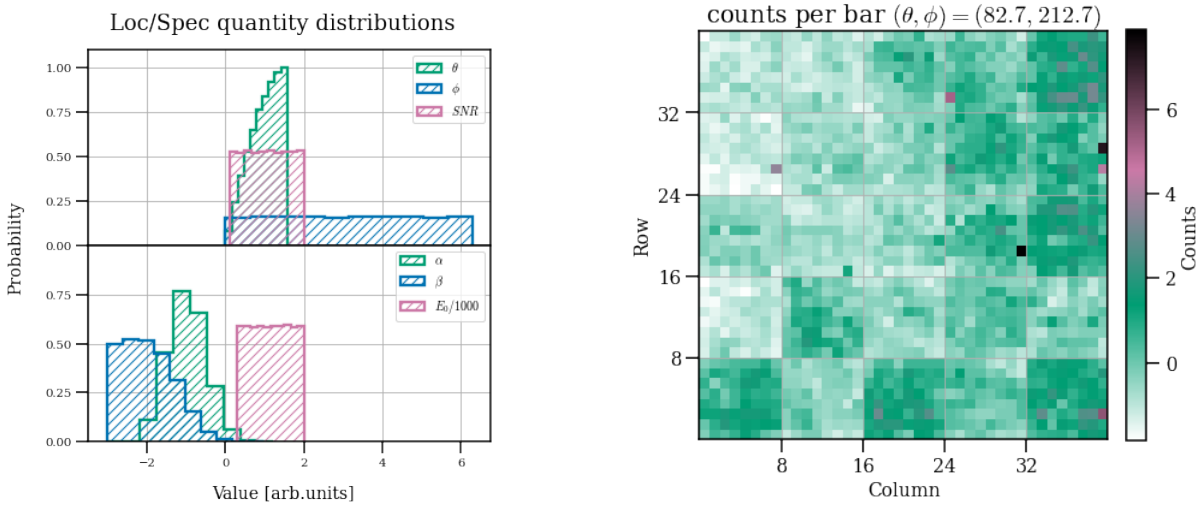


Figure 42: Left: Distribution of the different GRB properties of the POLAR simulations. The GRBs are distributed homogeneously on the sphere. Right: Typical hitmap from a POLAR simulation. The modules are clearly delimited and hot pixels are also easy to visualize.

The normalization technique as used in chapters 9 and 10 have shown great results, and will hence be used in this analysis as well. The training target will be θ and $[\cos(\phi), \sin(\phi)]$ for the localization model. The model that will predict the spectral parameters will have $(-\alpha + 1)$, $(\beta + 3.1)$ and $\frac{E_0 - 100}{1000}$ as targets (see chapter 10.2).

12.2 Constant spectrum Analysis

In this case, all the simulated GRBs have the same spectral parameters as GRB170114. This is a GRB which was observed by POLAR and has a high signal-to-noise ratio making it relatively easy to analyze. By keeping the spectrum constant, while the data are already far more complex than for the toy Monte Carlo detector, the complexity of the problem is greatly reduced, which makes it an ideal dataset to verify localization methods.
In order to ensure better results, two models were investigated: the fully connected neural network and the convolutional one.

12.2.1 Architecture choice

In pursuit of having the best results, it is crucial to determine which kind of model performs the best on this kind of dataset. Both the Fully connected neural networks and the $2D$ Convolutional neural networks have shown great results and are compared in this section (their architectures are shown in figures 58 and 59).
Since both architectures are intrinsically different, it is difficult to have points of comparison apart from the results, so in order to regularize the process a bit, the same architecture as in 19 is used at the end of both models. Beyond that, it is clear that in order to ensure the best results, proper hyperparameter optimization would be necessary. However, such a study will require significantly more time than that available during this project.
Both models were trained on the same training data and targets. The loss is in both cases the mean squared error and both models are trained with a batch size of 400 over 15′000 epochs. Such training sessions take several days on a GPU. As before, the mean absolute error and the angular distance between the targets and predictions on the validation sample are monitored and their evolution is shown in figure 44. In both cases, the model has been saved at the best validation angular distance[22]
As it has been seen in chapters 9 and 10.1, the mean angular distance can not be the only criterion to decide which model is the best, so it is important to compare their performances on the test data as shown on figure 43 that shows the distribution of the angular distance between target and predictions on the testing sample for the two investigated models. It is interesting to see that even though the fully connected neural network has many more trainable parameters than the convolutional one (respectively 4′164′191 and 224′095 trainable parameters.), the performances are not better. An important thing to notice is that the shape of the distribution and the statistical quantities of the distributions are comparable to the results obtained in the two previous chapters.

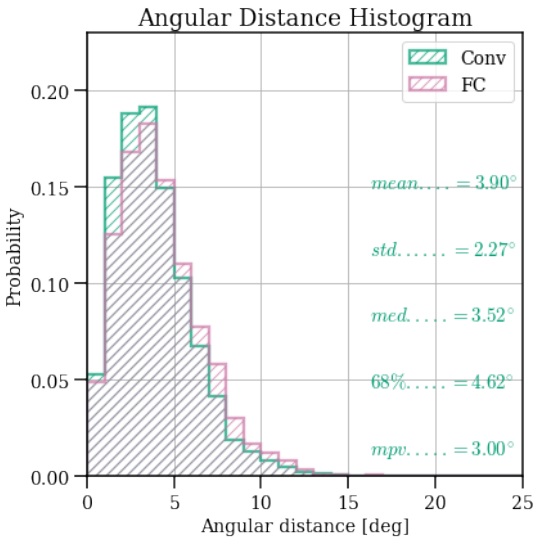
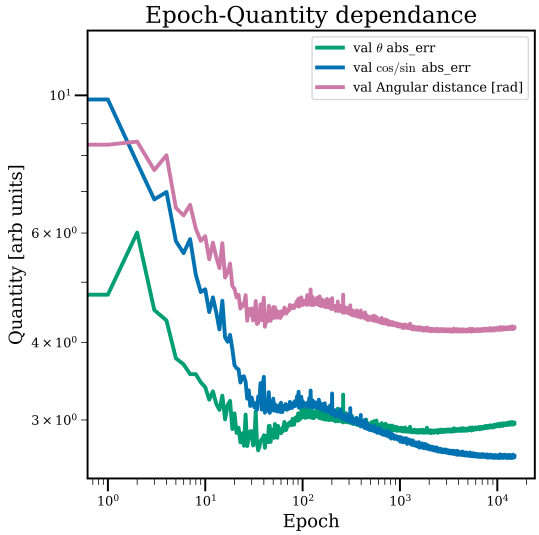


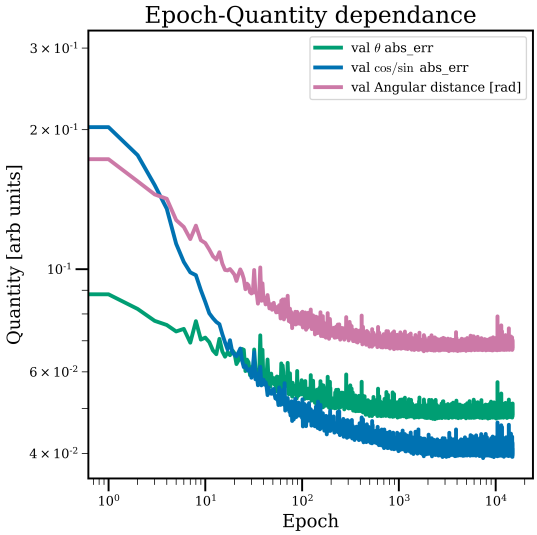
Figure 43: Distributions of the angular distance between target and predictions for the fully connected neural network and the convolutional one.

Previous analysis has shown that depending on the geometry of the detector, some incoming directions might be harder to predict accurately than others (*eg.* GRBs coming from the poles were hard to locate accurately in chapter 9 but not particularly in 10.1). In this context, figures 44c and 44d show the correlations between targets and predictions on the testing sample. It is evident that for both models, the pole is a region where it is hard to get an accurate localization.

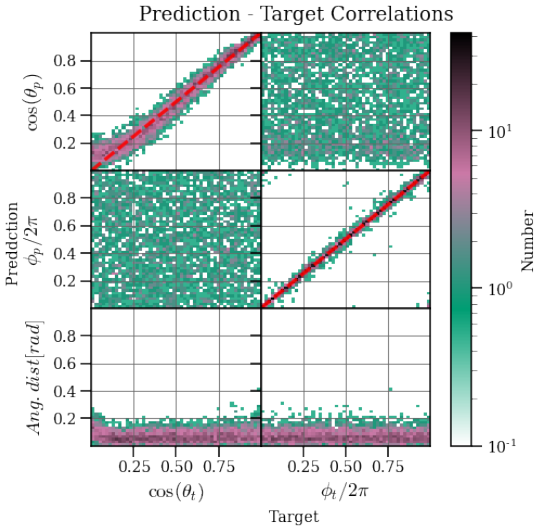
[22] This is obviously a naive calculation.



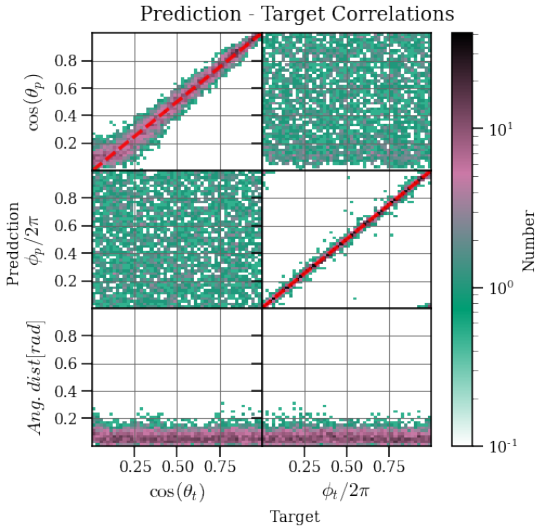
(a) Quantity evolution graph for the FC NN. It shows a short period of overfitting, after which the model didn't improve a lot.



(b) Quantity evolution graph for the $2D$Conv. It is much more stable than with the FC NN and the model barely improves after 3000 epochs.



(c) Correlations between targets and predictions on the testing sample for the fully connected network trained using the energies in the 1600 bars. The model has difficulties to predict events coming from the equator. On the other hand, the predicted ϕ values are close to the targets (the ideal case is marked with the red line).



(d) Correlations between targets and predictions on the testing sample for the convolutional network trained using the energies in the 1600 bars. Considerable errors are to be noted for events coming from the equator.

Figure 44: Top graphs : Evolution of the monitored quantities as a function of the training epoch for FC NN and the $2D$ Conv. Both models were saved at the epoch where the mean angular distance was the smallest. Bottom graphs: Correlations between targets and predictions on the testing sample for two models that were trained using the energy histogram per bar.

12.2.2 Reduced data

After testing different kinds and architectures of models, it appeared that the different models were taking considerable time to converge. For instance, the $2D$ convolutional neural network took over three days to finish, with roughly 5000 $epoch/day$). The reason is largely due to the considerable weight of the training data, and especially the energy, as there are 32 energy bins per bar. The file containing the energies of the 50376 simulations weighs $9.7GB$, whereas the file containing the counts and target values represents only $309MB$. The size of this dataset poses a challenge as it exceeds the available RAM capacity of many personal computers.

It was therefore investigated to reduce the data size by summing all the energies by module of 8×8 scintillator bar. This means that instead of having an array of 1600×32 energy bins per simulation, the data is reduced to 25×32 energy bins, which reduces the needed storage by a factor of 64. The models converged significantly faster since roughly 3 hours were needed for the models to converge. Additionally, grouping the energies helped the model to converge much faster as the fully connected neural network shows clear overfitting signs from epoch ~ 800, while the monitored quantities of the convolutional model don't seem to decrease after ~ 3000 epochs.

The angular distance distribution is shown in figures 45 and it is to be noted that there are no big differences between the distributions, even though the convolutional neural network is slightly better. The black dashed line represents the distribution obtained using the energies by bars, which is the best one.

As can be seen in figure 46c and 46d that show the correlations between target and predictions on the testing sample, both models are providing accurate predictions on the ϕ angle as it follows the red dashed line, representing the ideal case. Both models seem to have some difficulties to predict events at the equator since the angular distance is bigger for those events. On the other hand, the events on the zenith are rather well predicted for both models, even though the angular distance seems to increase a bit for those events. It is important to note that by grouping the data, the models converged much faster, but the interesting point is that the obtained angular distance distribution is essentially the same. The faster convergence of the models and lower number of parameters makes it possible to test a lot of different architectures both rapidly and at lower computational costs.

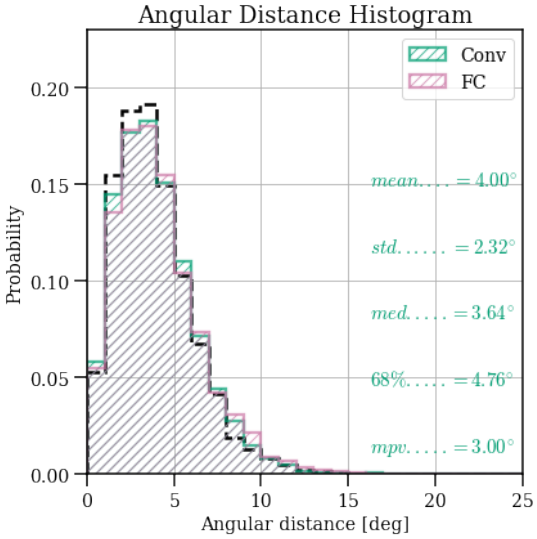
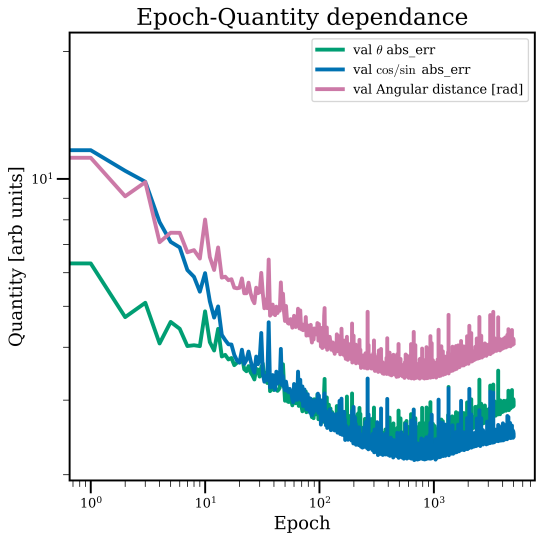
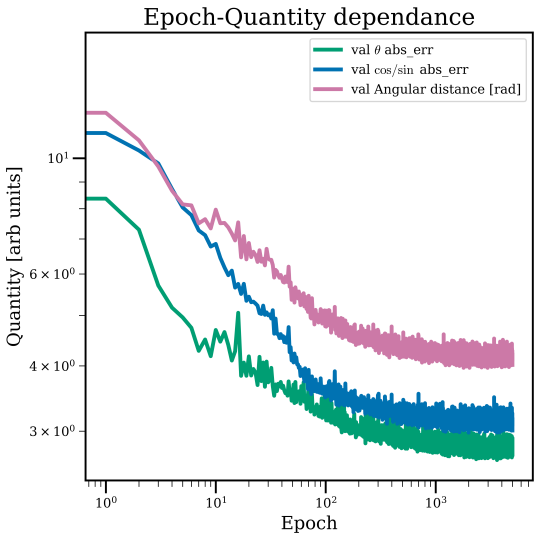


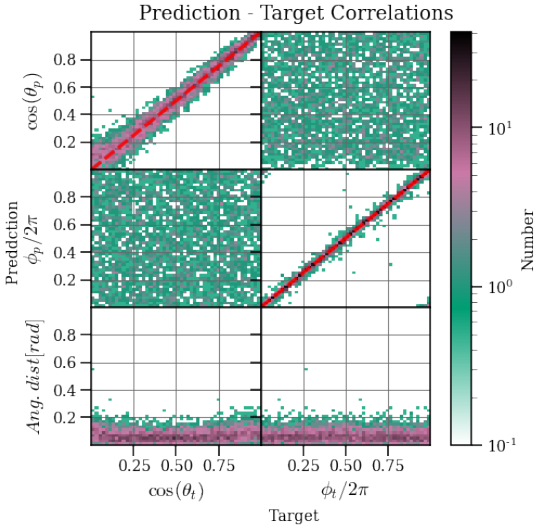
Figure 45: Angular distance between target and prediction on the test sample made by the convolutional model and fully connected one, both have been trained using the energy histograms grouped into modules. The statistical quantities refer to the convolutional one that seems to be the best based on those criteria. The black dashed line represents the distribution obtained using the convolutional network trained on the energies by bars (see figure 43)



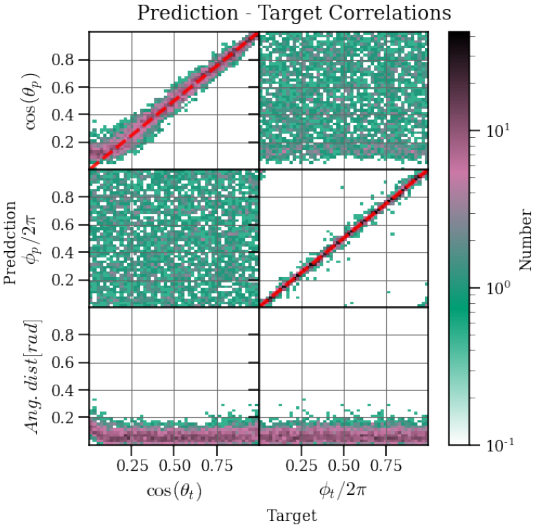
(a) Evolution of the monitored quantities as a function of the training epoch for the fully connected model, trained using the energies grouped by modules. Clear overfit signs are to be noted from epoch $\sim$ 800. The model was saved at the epoch where the "val_Angular distance" is the lowest.



(b) Evolution of the monitored quantities as a function of the training epoch for the convolutional neural network, trained using the energies grouped by modules. No overfit signs are to be noted.



(c) Correlations between target and predictions on the testing sample for the fully connected model. It shows great results, even though the angular distance increases for events at the equator (*ie.* $\theta = \pi/2$) and slightly at the zenith.



(d) Correlations between target and predictions on the testing sample for the fully connected model. Contrary to the FC model, the angular distance doesn't seem to be increasing for events at the zenith, but it does for events at the equator.

Figure 46: Top graphs: Evolution of the monitored quantities as a function of the training epochs for the two investigated models. Both are trained using the energies grouped by modules. Bottom graphs: Correlations between target and predictions on the testing sample for the two models trained using the energies grouped by modules.

12.3 Deeper model

As grouping the energy data by module helped to train models much faster without penalizing the performances, different architectures could be tested. The investigated architectures are thought to keep a relatively low number of trainable parameters. This can be done by using fewer neurons per layer, but more layers, making a deeper model.

In this chapter is shown a convolutional model (its architecture is shown in figure 60) that achieved better performances on the testing sample as can be seen in figure 48a that compares the three convolutional neural networks studied in this work. It is clear that this model is the best one since the statistical quantities of the angular distance distribution are the smallest. It is also clear from the bottom left graph of figure 48b that this model is better at predicting GRBs coming from high θ values (*ie.* $\cos(\theta) \sim 1$), even if the error for those events are the biggest. Consequently, this model will be used for the complete analysis.

As a final test, it is interesting to see the predictions of simulations of 1000 GRBs, all of them having the same properties as angular and spectral properties as GRB170114A. The predictions are shown in figure 47. Predictions are forming a blob centred on the best-fit location, indicating that the simulations are consistent with the best-fitted GRB parameter.

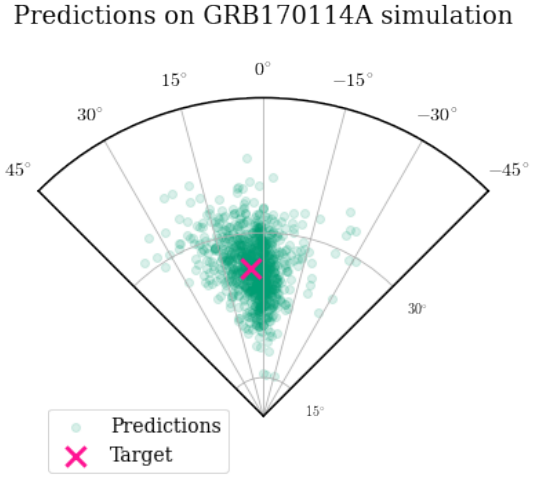
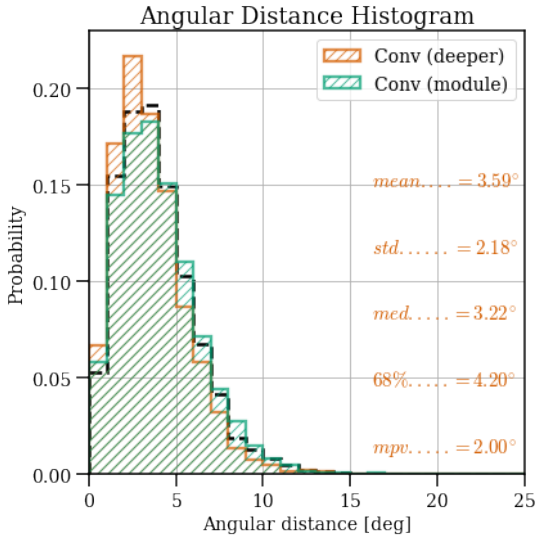
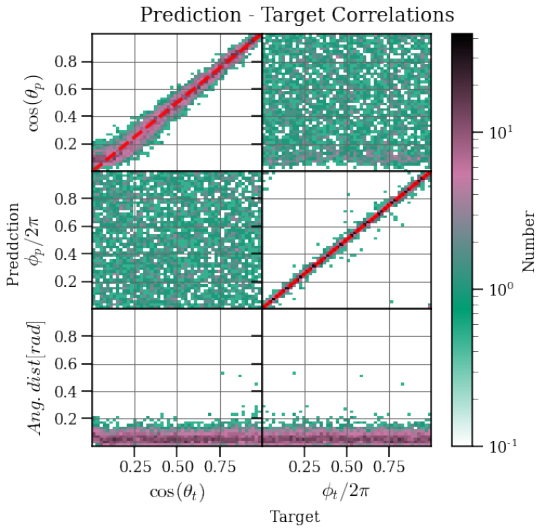


Figure 47: Predictions of the deeper convolutional neural network on 1000 simulations of GRB170114A. The produced blob is well-centred on the cross which is the best-fit location.



(a) Distribution of the angular distance between the target and predictions for the deeper convolutional neural network. The green histogram is the distribution corresponding to the model that has been trained using the energies grouped by module, and the black line is for the one trained using the energies per bar.



(b) Correlation between the target and predictions for the deeper convolutional neural network. The model is performing well, except for events that are coming from the equator. The ϕ predictions are good since very few outliers are to be reported.

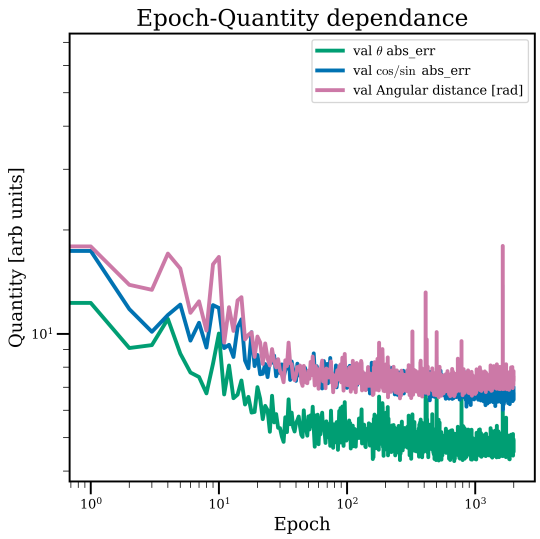
Figure 48: Detailed analysis of the deeper convolutional neural network.

12.4 Varying spectrum

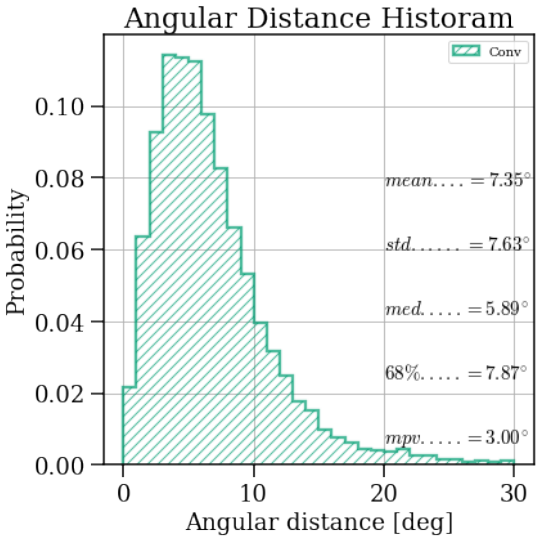
For this part, the dataset that is used is composed of 156′212 POLAR simulations, having the GRB parameter distributions shown in figure 42. The model that has been chosen to perform the localization analysis is the convolutional one because its angular distance distribution (see figure 48a) was the best one. For the spectral analysis, since the models tested in chapter 10.2 didn't show any big difference, both are investigated.

12.4.1 Localization

For this part, the model that will be used is the one that performed the best with the constant spectrum dataset. It was trained for 2000 epochs. The evolution of the monitored quantities is shown in figure 49a. Note that the model was saved at the epoch where the mean angular distance between targets and predictions is the lowest.



(a) Evolution of the monitored validation quantities over the training epochs for the localization model. There seem to be some overfitting signs from epoch ∼ 1000. The model has been saved at the epoch where the "val_Angular distance" is the smallest.



(b) Histogram of the distribution of the angular distance between target and predictions done by the localization model. The obtained distribution is the wider distribution ever obtained in this work which indicates that the dataset is complex to analyse accurately.

Figure 49: Analysis of the model evolution over the training epoch and of the resulting angular distance distribution.

The distribution of the angular distance is shown in figure 49b and it is clear that this dataset is harder to analyse because the mean of the distribution is at 7.35 degrees, which is the highest encountered in this work. The reason is obvious once the correlations are studied (see figure 50). Events at the equator are quite badly predicted since the angular distance is $\sim 0.2\,rad \approx 11.5°$. Events at the poles also seem to be hard to predict since the bottom left graph clearly shows a correlation between high $\cos(\theta)$ values and the angular distance. The error is comparable to the one near the equator, but more outliers are to be reported. The error seems to be independent of ϕ, whose values are overall well predicted.
In order to verify the method, it is interesting to see how the model predicts a simulation of GRB170114A (real data will be analysed in chapter 51). The scatter plot on the left shows the predictions on 1000 simulations, all of them being generated using the same band parameters (the ones from the best fit). Predictions are forming a blob that is close to the best-fit location, but it is not centred on it. The result is the same for the scatter plot on the right which shows

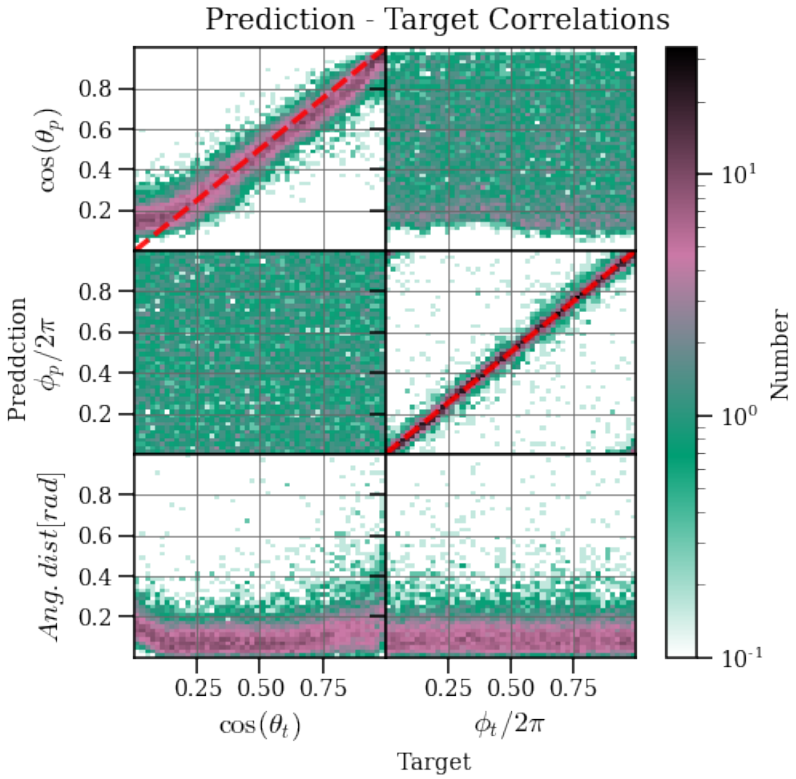


Figure 50: Correlation between target and values as predicted by the localization model trained on the complete dataset. The red lines are showing the ideal cases. Big errors have to be reported for events at the zenith and near the equator.

predictions on 5000 simulations for which the Band parameters were slightly varying from the est fitted ones. This could indicate that the best-fit location is not perfect, but as the model have considerable difficulties in accurately predicting the θ values, this hypothesis is very pretentious.

The same analysis as in chapter 8 can be performed in order to know if the training sample should be increased to get better results. To do so, 8 models were trained using an increasing fraction of the maximal training sample size. Each model is saved at the epoch where the angular distance between the target and predictions is the lowest. Finally, all the models are evaluated on the testing sample size and the statistical quantities of the angular distance distributions are computed. The result is shown in figure 52 which shows both the angular distance distributions and the evolution of their statistical quantities. It is evident that the actual training sample size is not optimal, because all statistical quantities are almost decreasing exponentially (it forms a line in log space). This result could have been expected. If one wants to reach an accuracy of the order of magnitude of one degree, it means that for each square degree in the sky ($\sim 40'000$), several combinations of band parameters are needed. Let's assume $\mathcal{O}(10)$ different values of α for which $\mathcal{O}(10)$ values of β are to be simulated, and this for $\mathcal{O}(10)$ different E_0 values. The resulting number of simulations to be performed is :

$$N_{sim} \sim 40'000 \times 10^3 = 4 \cdot 10^7$$

which is much more than what we have in this work. Note however that in this case, the SNR doesn't change, if several SNRs are simulated, N_{sim} is bigger.

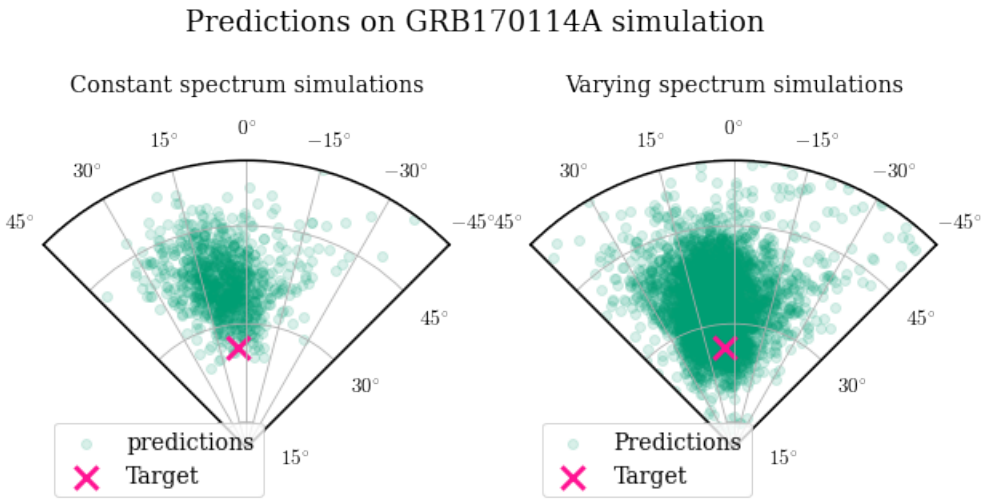


Figure 51: Predictions on GRB170114A simulations. The left plot shows predictions on simulations where all the GRBs have the same parameters (defined by the best fit), whereas, on the right, the parameters slightly vary. The resulting blobs are in both cases not perfectly centred on the best-fit location.

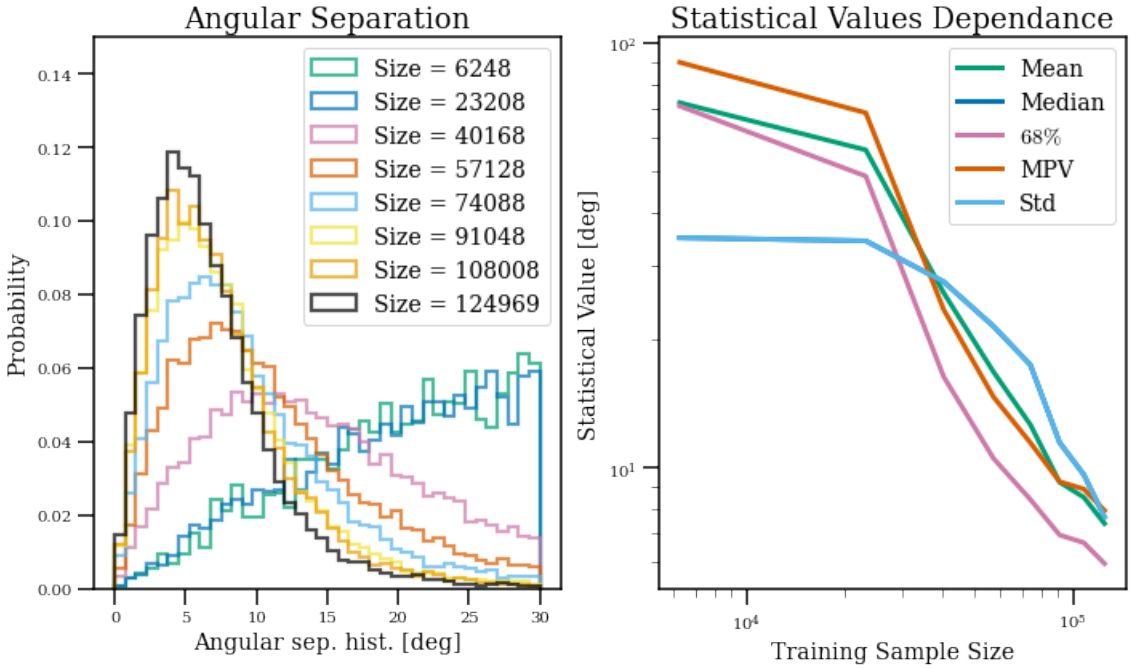


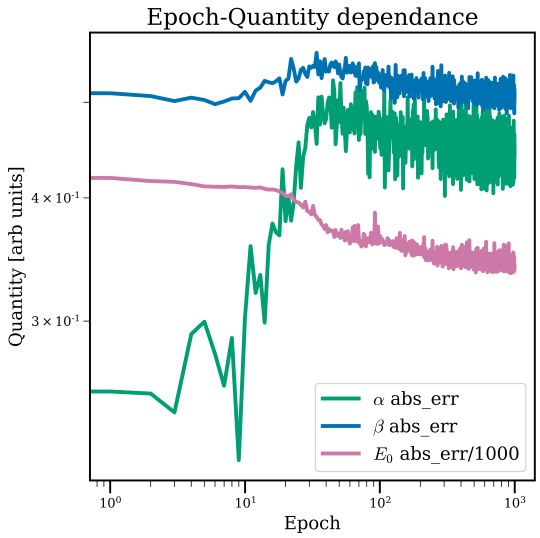
Figure 52: Evolution of the statistical quantities of the distribution of the angular distance between target and predictions on the testing sample for different models trained on an increasing training sample size.

12.4.2 Spectral Inference

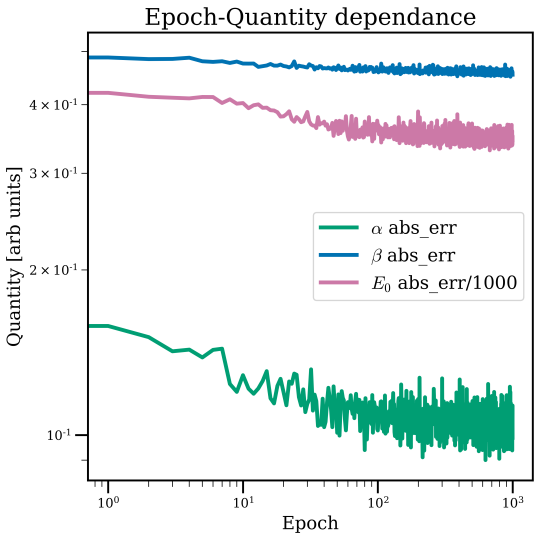
The two models presented in section 10.2 had similar results, so it is relevant to train both of them on the POLAR simulation. As a reminder, the output of the first model (which we will again call M_{bloc}) is a vector with three components and the loss is similar to the mean squared error but adds a penalty if the model predicts $\alpha < \beta$ and reduces the weight of the E_0 prediction by $\alpha - \beta$ (if it is positive) to take into account the fact that if $\alpha \approx \beta$, the power spectrum is recovered and E_0 has no meaning. The second model (which we call M_{sep}) has three independent outputs and the loss is the mean squared error. In both cases, the input of the models is a 32 bin histogram, obtained by summing the 1600 histograms (one for each bar).

The two models were trained using the POLAR simulations and as it can be seen on figure 53a and 53a, the evolution of the spectral parameters is very different from what was obtained in chapter 10.2. Indeed, for the model M_{block}, there are clear signs of overfitting after epoch 9. During this phase, the E_0 MAE also increases, while the MAE on the β parameter decreases. The three curves finally decrease but a lot of noise is to be noted on the α evolution curve. On the other hand, for the model M_{sep} the three evolution curves are decreasing, even though the E_0 MAE is not evolving a lot. The α MAE is also noisy but less than with M_{block}. Even though the scales of the two graphs are different, it is still clear that the model M_{sep} has lower MAE values.
In order to better compare the models, the correlation between the spectral parameters has to be studied in detail, as it was done and explained in 10.2. Those are shown on figure 53c and 53d (for M_{block} and M_{sep} respectively.)
Taking into account the strange epoch evolution of model M_{block}, it is not surprising that the α parameter is badly predicted. Even if the model was saved after only 9 epochs, it managed to learn that α should be greater than β as it can be seen in the top middle and middle left subplots of figure 53c. It is also to be noted that the β parameter is rather well predicted only for $\beta \geq -1.5$ similar to what was seen in the toy MC analysis. Concerning the E_0 parameter, it seems to be rather well predicted below $1000\,keV$ which makes sense because the sensitivity of POLAR is between $50\,keV$ and $500\,keV$.
The α predictions of model M_{sep} on the other hand are good since they follow the red line (which is the ideal case). On the other hand, the β parameter is not well predicted, especially for low values. Note however that all the β parameters are predicted smaller than α, which is good. Concerning E_0, the comment for M_{sep} is also valid here.

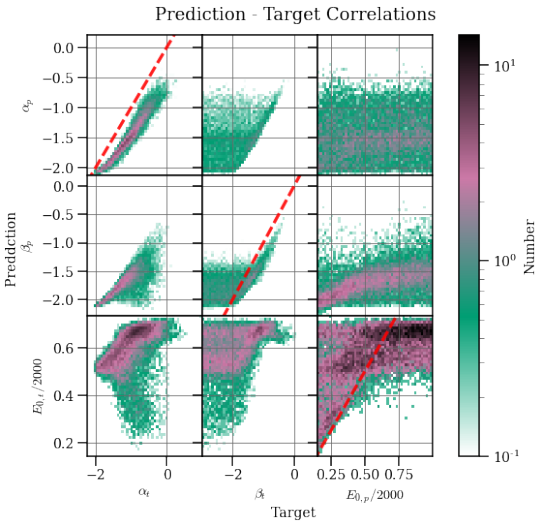
Overall, and even though the β parameter is not well predicted, the model M_{sep} is the best because it manages to predict α quite accurately. The reason why β is hard to predict is thought to be the same as in chapter 10.2, *ie.* when β is very small, the number of high energetic photon is low, which leads to imprecise prediction. This effect is more severe with the POLAR data than on the toy MC probably because the toy MC has only 10% of energy measurement uncertainty (where POLAR has $\sim 40\%$ [23]) and all the energies are measured, while on POLAR, energies are measured by an ADC (analogue to digital converter) and very high energies are binned in the overflow bin, which is not analysed here. The fact that the E_0 is difficult to predict is also due to the intrinsic degeneracy of the Band function. Indeed, when $\alpha \approx \beta$, the spectrum is essentially described by a power-law spectrum, in which E_0 has no meaning.



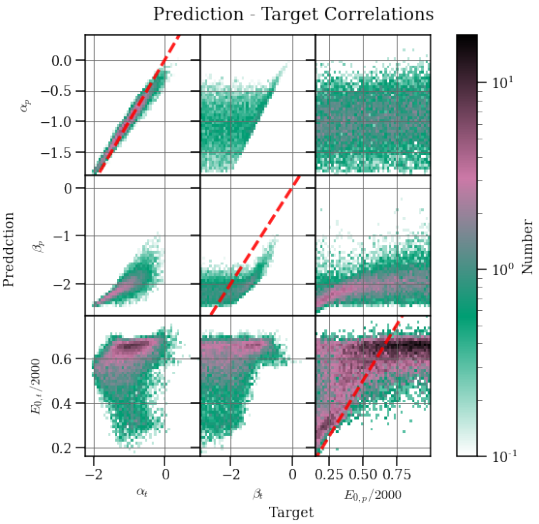
(a) Evolution of the mean absolute error (MAE) between target and predictions on the validation sample for the model M_{block}. It is evident that the evolution is not optimal as the α parameter starts to overfit while the other evolution curves are essentially stable. It is only after the overfit period that the MAE of the other parameters starts to decrease. A lot of noise is to be reported.



(b) Evolution of the mean absolute error (MAE) between target and predictions on the validation sample for the model M_{sep}. The evolution of the E_0 parameter is very subtle. On the other hand, the α and β parameters don't seem to overfit. A considerable noise is to be noted on the α evolution curve.



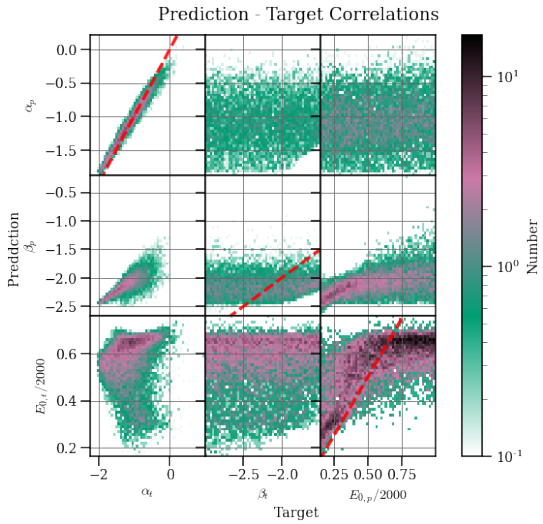
(c) Correlation between target and predictions of model M_{bloc}. This model is clearly not performing well as the α parameter is badly predicted. Nevertheless, the model always predicts $\alpha > \beta$.



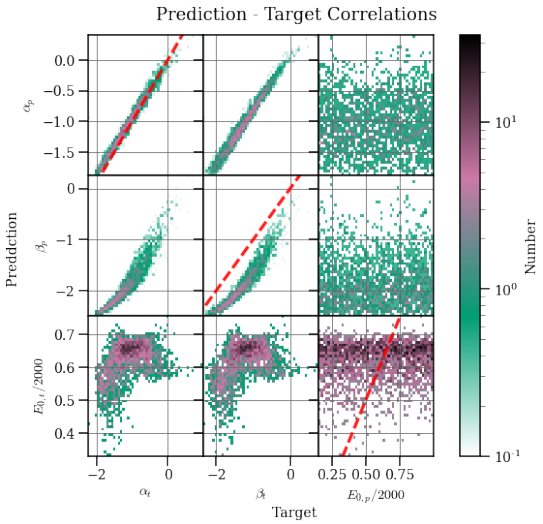
(d) Correlation between target and predictions of model M_{sep}. This model predicts the α parameter quite accurately, whereas the β and E_0 parameters are harder to predict. Note however that the model always predicts $\alpha > \beta$

Figure 53: Top graphs: Evolution of the mean absolute error between targets and prediction. The models have been saved at the smaller mean absolute error on the α prediction. Bottom graphs: Correlation between the targets and predicted Band parameters for the two spectral models. The red lines represent the ideal case. The bin content has been normalized such that they all have the same range.

To verify the above suspicions, two filters have been applied to the correlations of model M_{sep} and the results are shown in figures 54a and 54b. The first filter is meant to take events where $\beta \leq -1.5$. Those events have well-predicted α values, bad β predictions and E_0 seems to be relatively well predicted below $1000\,keV$, probably because a lower β implies fewer photons but makes it easier to predict E_0 which is characterized by the knee of the Band function. The second filter selects the events where $|\alpha - \beta| < 0.3$. This way, and as it can be seen in figure 54b, the α predictions are good. The β predictions are not very good because there is an offset between the ideal case (the red line to what the model should tend) and the actual distribution. The cause of this offset is unknown but could be caused by the fact that the training sample contains all the possible cases, including the degenerated ones, which could create a bias. Using this second filter makes the predictions on E_0 very bad, which confirms the hypothesis of the degeneracy.



(a) Correlation between target and predictions of model M_{sep}, using a filter that only selects events $\beta \leq -1.5$. It shows that the bad predictions of the β parameter are probably caused by the low number of photons induced by the steep power spectrum tail.



(b) Correlation between target and predictions of model M_{sep}, using a filter that only selects events $|\alpha - \beta \leq 0$.. It shows that the bad predictions of the β parameter are probably caused by the low number of photons induced by the steep power spectrum tail.

Figure 54: Correlation between target and predictions of model M_{sep} using different filters meant to verify the hypothesis causing the bad predictions on both β and E_0.

Overall, the M_{sep} model performs well but is limited by the intrinsic degenerate cases that the Band function allows and its performances are thus reduced. The predictions on the energy values above $1000\,keV$ are thought to be caused by the lower sensitivity of the detector at those energies. The complete analysis of real data (detection, localization and spectral analysis) will be performed in section 13.

Rapidity Since the aim of the project is to make *rapid* predictions, it is good to provide some results. The two testing samples were composed of $15'621$ events and the required time to perform predictions was $1.53s$ for the localization and $1.2s$ for the spectral analysis (model M_{sep}). The computation time for a single event was also computed and is $\sim 83ms$. Note that the predictions could be faster by using tools such as `TensorRT`, which are dedicated tools (the speed could be improved up to $36\times$ according to Nvidia[23]).

[23]Source: https://developer.nvidia.com/tensorrt

13 Complete analysis and discussion

To finish this work, it is interesting to perform a complete analysis, which means detecting GRBs with the LSTM model developed in section 11 and then analysing their location using the model from section 12.4.1.
For this analysis, real POLAR flight data are used. Data are converted to form a time series of counts per bar and energies per module. The data can be reshaped or grouped by modules to be provided to the different models. A typical POLAR lightcurve is shown in figure 55 and has been chosen because it contains two GRBs. This particular lightcurve represents almost a day ($\sim$ 23hours).

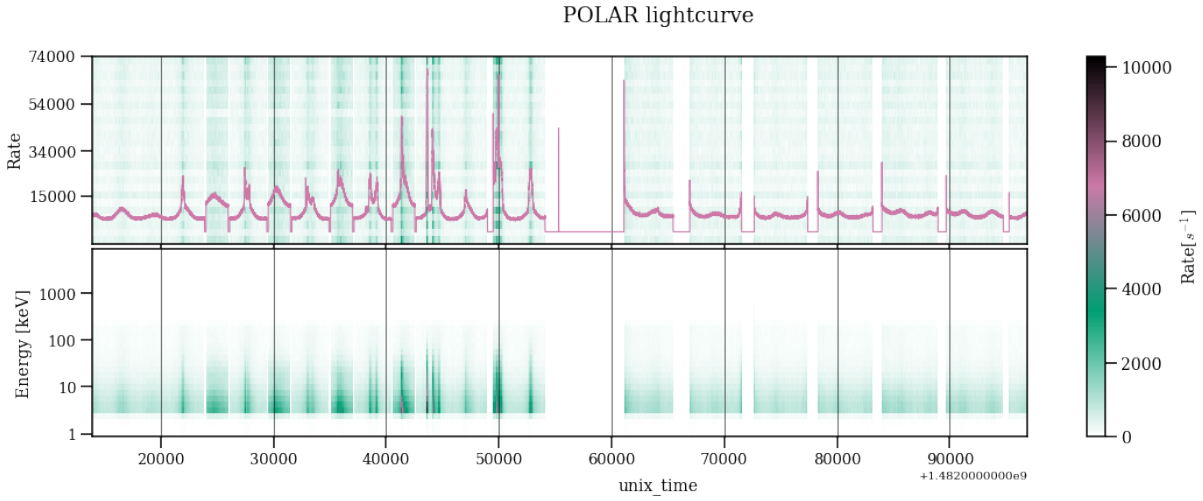


Figure 55: Example of a typical POLAR lightcurve. The moment when the POLAR detector is over the SAA is easy to see due to the cutoff. Two known GRBs are in this lightcurve. The background colour indicates the rate per module on the top graph and the rate per energy bin on the bottom one.

13.1 Detection

The first step is to detect the signals to analyse using the LSTM model. To do so, the lightcurves are separated into 60s lightcurves whose counts need to be grouped by module and normalized in order for the model to make predictions on it. In the lightcurve shown in figure 55, the model was able to detect 376 events that have a confidence level above 97%. As it was explained in chapter 11, this high number is expected because the real background is quite different from the simulated one as it contains other events than just GRBs. Therefore, filters need to be applied. After some tests, it turned out that the following criteria were good filters :

- Select events only if the rate during the two previous minutes was greater than 100 (in order to avoid events near the South Atlantic anomaly where the instrument is switched off).
- Select events only if there is an event before having a confidence level of at least 95%. Note that by applying this filter, very short signals are penalized.

Of course, those filters can be improved, but they gave good results on the few lightcurves that were tested, by filtering events that had no noticeable event and by allowing interesting events. Using those two filters, the number of triggers on the above lightcurve was reduced to 97. Note that some events are triggered tens of times, for the simple reason that they last longer. Overall, 21 different events were recorded.

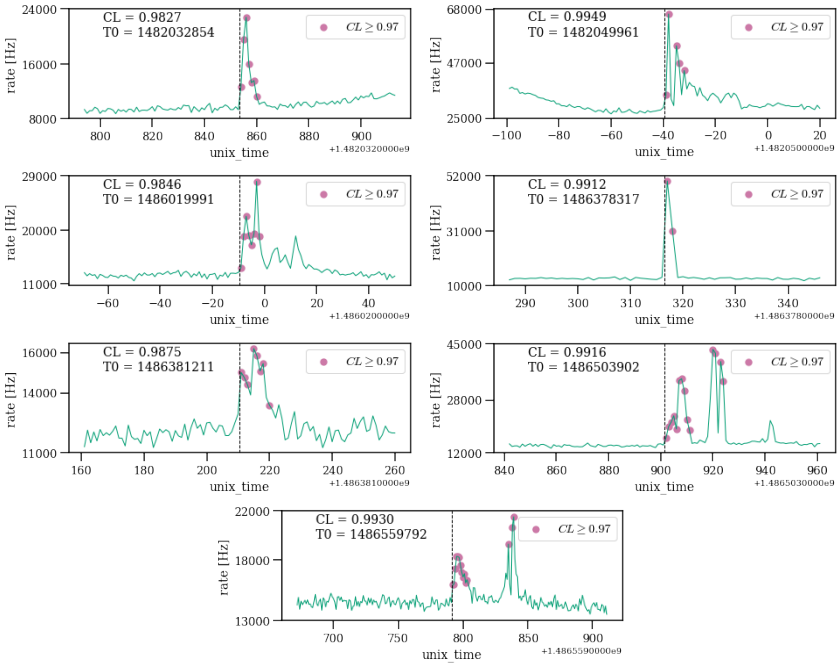


Figure 56: Exhaustive list of the detected GRBs in the few analysed days. All known GRBs were detected. From left to right and top to bottom: GRB161218A, GRB161218B, GRB170202B, GRB170206A, GRB170206C, GRB170207A, GRB17020C. Events, where the predicted probability is ≥ 0.97, are shown in magenta.

The analysis of the above light curve and others was successful because all the known GRBs were correctly detected, and all the corresponding lightcurves are shown in figure 56.
However, other signals were detected, such as the one characterized by three consecutive peaks in figure 57. Note however that the model only classifies the first peak as being a GRB. Interestingly enough, this signal has the same shape as a GRB but no GRBs were reported at this time. This however does not mean that this is not a real GRB, this has to be confirmed with follow-up studies. Other examples are the ones where the rate increases from a few thousand to tens of thousands in less than two minutes (the lower one in figure 57). After an internal discussion[24] and a short analysis, it was clear that this signal was not caused by induced particles. Indeed, the typical characteristic of such an event is that a row (or a column) of bars is triggered, which was not observed. Additionally, the spectrum of this event was too energetic to be only due to the background or solar flares. It is to be noted that long signals are only detected at their beginning, which is certainly due to the fact that by training the model on $60s$ lightcurves, a clear bias is introduced towards the detection of GRBs having a duration of tens of seconds or less.

13.2 Localization and spectral analysis

Once the signals are detected, they can be located, by using the deeper model of chapter 12.4.1. To do so, the counts per bar and the energy histograms need to be integrated over the full GRB duration, and normalized (*ie.* all values should be divided by the maximal count value of the training sample). However, by doing so, it is clear that the normalization that was used was not optimal, because if the analysed GRB is long, the maximal count value can be bigger than the one in the training sample. As a result, the predictions are completely wrong (one of the θ values was a few tens of thousands, which makes no sense). To get around this problem, the counts

[24] Merlin and me

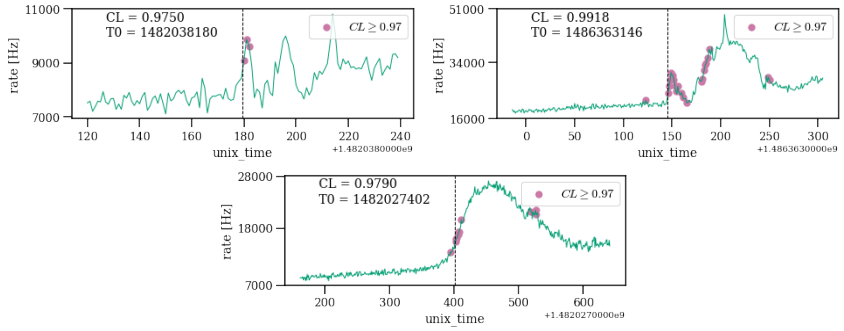


Figure 57: Three examples of GRB candidates detected during the lightcurves analysis. Other ones could easily be found by applying the model to the whole mission duration.

will be divided by the maximal value of the true GRB, which guarantees that all the values are between 0 and 1. Obviously, this is not optimal. For the spectral analysis, this method was also used, but it always provided the same results, which means that the model is not predicting anything interesting. This analysis must hence be improved in order to be tested on real data.
In order to be constant, the counts will always be integrated from $T_{90}/2$ seconds before the burst to $T_{90}/2$ seconds after the end of the burst, where T_{90} taken from the GCN[25] reports.
In table 1 we show the predicted location and spectral parameters for different GRBs (all of them have been detected by the LSTM model).

GRB	θ_p/ϕ_p	θ_t/ϕ_t	Ang.Dist.
GRB161218A	43.2/356.8	24.3/356.6	18.9
GRB161218B	79.1/209.8	77.8/252.2	41.5
GRB170202B	90.1/185.5	-	-
GRB170206A	68.1/174.0	19.5/148.7	50.8
GRB170206C	85.0/195.46	-	-
GRB170207A	85.8/334.5	70.6/357.8	27.33
GRB170208C	107.3/271.6	93.2/310.0	40.2

Table 1: Table of predicted location for the detected GRBs. All values are in degree

The number of detected GRBs is too low to perform any statistical analysis, but it is still possible to see that overall, the predictions have large errors. This is partly due to the fact that the model was trained with shorter GRBs. Generally, however, these first results look promising and prompt to continue such studies in the POLAR-2 collaboration.

[25] url: https://gcn.gsfc.nasa.gov/

14 Discussion

14.1 GRB detection

This analysis was quite successful since for all the analyzed lightcurves that contained known GRBs, the model was able to detect them. However, a large number of events that looked like GRBs were reported. Since they haven't been yet properly analyzed, it is difficult to make a clear conclusion. If they turned out to be GRBs that were not detected up to now, that means that the model is better than the current methods, which is a pretentious assumption. On the other hand, if those events are caused by the background, it means that the filters have to be improved in order to have a reasonable number of triggers per day. This is very important since the aim of this work is to build a reliable alert system, and hence, sending false positive alerts should be avoided. In this context, an additional model that classifies an event in different classes (*eg.* GRB, Background, solar flare, particle-induced events, etc...) is a good idea. This however requires a very good knowledge of which events constitutes the background as they need to be simulated to provide an exhaustive training sample. It would also be interesting to see if the model that was trained on using bins of $1s$ can make accurate predictions on lightcurves having bins of smaller time intervals.
Overall, this analysis showed great results, which were verified by discovering interesting events that could be GRB candidates as they looked like GRB signals, but no alert was reported by any other instrument in the past.

14.2 GRB localization

The analysis of the complete dataset showed that this task is not easy and needs to be improved if one wants to analyse GRBs directly on the CSS.
The reason is that real GRBs are more complex than the simulation since they can be very long. In order to properly analyse them, they should be normalized independently, and the length of the GRB should not be an issue. A good idea would be to centre and normalize the counts per bar (*i.e.* to each count per bar, subtract the mean and divide by the standard deviation of the hitmap). This way, the localization analysis is based on the relative counts per bar and not on the absolute count per bar (dividing the counts by the maximal value only changes the scale).
The few real GRBs that were analysed were quite badly predicted, mainly due to a big error on the θ angle. As it was explained in 3.2, the hitmap can be similar for a soft GRB coming from a low θ value (close to the zenith) as a harder GRB coming from a larger θ value. Consequently, all the models used in this work (and trained on POLAR data) were having difficulties to analyse events coming from the equator. It doesn't seem to be an artefact due to the convolution since this effect was also present by using fully connected neural networks. This could also be induced mathematically, but it is less probable since Additionally, the effective area of POLAR is strongly reduced for $\cos(\theta) < 0.4$ [30], which can also explain why GRBs coming from $\theta \approx \pi/2$ are harder to predict accurately. Finally, it is clear from figure 52 that the training sample size must be increased in order to have better results. Rough calculation showed that a training sample size of $\mathcal{O}(10^7)$ would be appreciable.
Overall, the localization analysis showed interesting results on the real POLAR data, it is clear that more work is needed to achieve a sub-degree level of accuracy. Indeed, a better normalization method has to be defined and more importantly, a proper hyperparameter optimization has to be done in order to ensure better results. Note that more models could be tested by reducing the data size as the results are similar while reducing the training time.

14.3 GRB Spectral analysis

This analysis is by far the most unsuccessful one since it was not possible to test it on real data, which again motivates the search for a clever normalization method. Nevertheless, using the POLAR simulations, the best model (M_{sep}) showed that the degenerates cases that the Band function allows (*eg.* when $\alpha \approx \beta$, the function is essentially a power spectrum, in which case E_0 has no meaning.) are very hard to analyze. This was proven by using filters that showed that if β is very small, the number of high energetic photons being detected is low (due to the steepness of the power spectrum), which leads to a poor prediction on β, but a better prediction on E_0 since it is characterized by the knee of the power spectrum. This effect was more important on the POLAR data than on the toy MC simulations because POLAR has an energy measurement uncertainty of $\sim 40\%$ (10% for the toy MC) and high energy measurement are placed in the overflow bin (which is not the case for the toy MC detector). A second filter, selecting GRBs for which $|\alpha - \beta| < 0.3$ showed that for those events, the β prediction was better, even though they were not ideal since a clear offset between the obtained distribution and the ideal case was reported. In this case, the E_0 predictions are poor. This analysis leads to the conclusion that the Band function does not always describe a GRB power-law spectrum in a way that is easy to analyze with deep learning.
Even though a proper analysis hasn't been performed, it is highly probable that to achieve better results, more simulations need to be done.
Overall, this method showed that analyzing GRB spectra using a Band function is a difficult task. Additionally, a proper normalization method would certainly improve the results and definitely allow testing the model on real data, which was not possible in this work.

14.4 HAGRID in space ?

It is clear that in its current version, the analysis method is not well suited for a space application due to the results not being satisfactory or enough to justify its installation on the CSS.
Nevertheless, the method is promising since with an insufficient amount of training data, the models were able to predict relatively the location of the simulation. The method needs to be improved to analyze GRB location and spectral parameters reliably.
There are very good elements that justify further investigation to use deep learning for the localization and spectral analysis of POLAR-2. Since this detector will be much more sensitive than POLAR, it should be able to achieve better results with a similar training sample size as the one used in this work. Additionally, deep learning is by nature well suited for alerts applications since once the model is trained, predictions can be done in a few milliseconds and at low computational costs, which is much shorter than the χ^2 method that is currently being used (which takes several seconds).
Note however that part of the method could be implemented in the POLAR-2 mission relatively easily. Indeed, the GRB detection method showed great results and it is both reliable and rapid. Note, however, that to ensure better results, the background should be simulated in a more conscientious way than what was done in this work. Another idea could be to take real background lightcurves and artificially add GRBs.
Overall, besides the GRB detection model that could be implemented relatively easily, the localization, and spectral models should be improved.

15 Conclusion

The goal of this work was to investigate the feasibility of rapidly and accurately inferring the location and spectral parameters of an incoming GRB.

Using simulations, proper analysis techniques and a strong intuition could be developed. Those methods were applied to POLAR simulations that provide a training sample of $156'212$ GRBs, all of them having different incoming directions and spectrums, characterized by the band function. The localization analysis showed good results on the simulations as the mean angular distance between targets and predictions on the testing sample was $7.35°$. Different model architectures were investigated and showed that convolutional neural networks are showing good potential to accurately analyse GRB incoming direction as they seem to be more accurate than the fully connected neural networks for some specific angles, like the zenith ($\theta = 0$). It was also shown that reducing the data sizes by grouping the data into modules of 40×40 bars can help to make better predictions and to reduce the training time, which could allow more models to be investigated.
The test performed on real GRB data showed that some methods used in this work were not optimal as they are rate-dependent. All the GRBs on which predictions were made were located up to $\sim 50°$ off the best-fitted location. To increase the accuracy, it is thought that a training sample size containing at least $10 - 100$ times more GRBs should help.

The spectral analysis showed that the Band function is not always the best way to describe a GRB spectrum. The reason is that due to the degenerates cases of the band function (*ie.* when $\alpha \approx \beta$), the model doesn't manage to make predictions on the energy, because this quantity has no meaning in this case. It was also reported that for GRBs having very low β values, the predictions on β were very poor due to lower statistics, but also to the fact that high energy measurements are placed in the overflow bin, which is not analysed.
The model has not been able to be tested on the real data due to the normalization method that was not optimal.

The part of this work that showed the best result is the one that was originally not planned: GRB detection. Using POLAR simulation and rather simple lightcurve simulations, which were used by an LSTM (Long Short Term Memory) neural network. This model was tested on real POLAR lightcurves, which showed great results since all known GRBs were detected by the model. Additionally, numerous GRB candidates were detected indicating that the model performs well.

Overall, to rapidly analyse GRBs on the CSS (China Space Station), deep learning models are thought to be an option with a lot of potentials as this method allows to make predictions in a few milliseconds. Detecting GRBs onboard the CSS is a task that would not require much improvement from the current state. On the other hand, accurately locating an incoming GRB and reconstructing its spectral properties would require more work.

References

[1] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] B. P. Abbott et al. Gravitational Waves and Gamma-Rays from a Binary Neutron Star Merger: GW170817 and GRB 170817A. *The Astrophysical Journal*, 848(2):L13, Oct. 2017. ISSN 2041-8213. doi: 10.3847/2041-8213/aa920c. URL https://iopscience.iop.org/article/10.3847/2041-8213/aa920c.

[3] L. Audubon. Le Machine Learning: définition, histoire et cas d'usage, 2022. URL https://www.sicara.fr/fr/parlons-data/machine-learning.

[4] D. Band and other. BATSE Observations of Gamma-Ray Burst Spectra. I. Spectral Diversity. *The Astrophysical Journal*, 413, Aug. 1993. ISSN 0004-637X. doi: 10.1086/172995. URL https://ui.adsabs.harvard.edu/abs/1993ApJ...413..281B. ADS Bibcode: 1993ApJ...413..281B.

[5] D. Band et al. BATSE Observations of Gamma-Ray Burst Spectra. I. Spectral Diversity. *The Astrophysical Journal*, 413, Aug. 1993. ISSN 0004-637X. doi: 10.1086/172995. URL https://ui.adsabs.harvard.edu/abs/1993ApJ...413..281B. ADS Bibcode: 1993ApJ...413..281B.

[6] J. Brownlee. *Master Machine Learning Algorithms*. Machine Learning Mastery, 2016. URL http://MachineLearningMastery.com.

[7] J. Brownlee. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, July 2017. URL https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[8] J. Brownlee. A Gentle Introduction to Batch Normalization for Deep Neural Networks, Jan. 2019. URL https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/.

[9] J. M. Burgess. grburgess/cosmogrb: A cosmological GRB light curve simulator. Because, why not?, 2022. URL https://github.com/grburgess/cosmogrb/tree/master.

[10] J. M. Burgess, H.-F. Yu, J. Greiner, and D. J. Mortlock. Awakening the BALROG (BAyesian Location Reconstruction Of GRBs): A new paradigm in spectral and location analysis of gamma ray bursts, Oct. 2016. URL http://arxiv.org/abs/1610.07385. arXiv:1610.07385 [astro-ph].

[11] E. Costa et al. Discovery of an X-ray afterglow associated with the γ-ray burst of 28 February 1997. *Nature*, 387(6635):783–785, June 1997. ISSN 1476-4687. doi: 10.1038/42885. URL https://www.nature.com/articles/42885. Number: 6635 Publisher: Nature Publishing Group.

[12] J. E. Davis. The Formal Underpinnings of The Response Functions Used in X-Ray Spectral Analysis. *The Astrophysical Journal*, 548(2):1010–1019, Feb. 2001. ISSN 0004-637X, 1538-4357. doi: 10.1086/319002. URL https://iopscience.iop.org/article/10.1086/319002.

[13] F. Fleuret. *The Little Book of Deep Learning*. 2023. ISBN 978-1-4476-7861-8.

[14] G.Cavallo and M.J.Rees. A qualitative study of cosmic fireballs and gamma -ray bursts. *Monthly Notices of the Royal Astronimical Society*, 183, 1978. doi: 10.1093/mnras/183.3.359. URL https://ui.adsabs.harvard.edu/abs/1978MNRAS.183..359C.

[15] GIMP Documentation. Convolution Matrix, 2010. URL https://docs.gimp.org/2.6/en/plug-in-convmatrix.html.

[16] D. Godoy. Understanding binary cross-entropy / log loss: a visual explanation, July 2022. URL https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.

[17] J. Greenwood. The correct and incorrect generation of a cosine distribution of scattered particles for Monte-Carlo modelling of vacuum systems. *Vacuum*, 67(2):217–222, Sept. 2002. ISSN 0042-207X. doi: 10.1016/S0042-207X(02)00173-2. URL https://www.sciencedirect.com/science/article/pii/S0042207X02001732.

[18] D. H. Hartmann. Afterglows from the largest explosions in the universe. *Proceedings of the National Academy of Sciences*, 96(9):4752–4755, Apr. 1999. doi: 10.1073/pnas.96.9.4752. URL https://www.pnas.org/doi/full/10.1073/pnas.96.9.4752. Publisher: Proceedings of the National Academy of Sciences.

[19] Healpix documentation. HEALPix - Features, 2019. URL https://healpix.sourceforge.io/.

[20] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[21] John David Jackson. *Classical Electrodynamics, 2nd Edition*. John Wiley & Sons, Inc., Oct. 1975. URL http://archive.org/details/ClassicalElectrodynamics2nd.

[22] O. Klein and Y. Nishina. Über die Streuung von Strahlung durch freie Elektronen nach der neuen relativistischen Quantendynamik von Dirac. *Zeitschrift für Physik*, 52(11):853–868, Nov. 1929. ISSN 0044-3328. doi: 10.1007/BF01366453. URL https://doi.org/10.1007/BF01366453.

[23] M. Kole. POLAR: Gamma-Ray Burst Polarimetry onboard the Chinese Spacelab. In *Proceedings of The 34th International Cosmic Ray Conference — PoS(ICRC2015)*, page 999, The Hague, The Netherlands, Aug. 2016. Sissa Medialab. doi: 10.22323/1.236.0999. URL https://pos.sissa.it/236/999.

[24] M. Kole and POLAR-2 collaboration. POLAR-2: The First Large Scale Gamma-ray Polarimeter. In *Proceedings of 36th International Cosmic Ray Conference — PoS(ICRC2019)*, page 572, Madison, WI, U.S.A., July 2019. Sissa Medialab. doi: 10.22323/1.358.0572. URL https://pos.sissa.it/358/572.

[25] M. Kole, N. De Angelis, et al. The POLAR Gamma-Ray Burst Polarization Catalog. *Astronomy & Astrophysics*, 644:A124, Dec. 2020. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/202037915. URL http://arxiv.org/abs/2009.04871. arXiv:2009.04871 [astro-ph].

[26] O. Luongo and M. Muccino. A Roadmap to Gamma-Ray Bursts: New Developments and Applications to Cosmology. *Galaxies*, 9(4):77, Dec. 2021. ISSN 2075-4434. doi: 10.3390/galaxies9040077. URL https://www.mdpi.com/2075-4434/9/4/77. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[27] P. Mészáros. The Fireball Model of Gamma-Ray Bursts. *Progress of Theoretical Physics Supplement*, 143:33–49, 2001. ISSN 0375-9687. doi: 10.1143/PTPS.143.33. URL https://academic.oup.com/ptps/article-lookup/doi/10.1143/PTPS.143.33.

[28] C. Olah. Understanding LSTM Networks – colah's blog, 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[29] B. Paczynski and J. E. Rhoads. Radio Transients from Gamma-Ray Bursters. *The Astrophysical Journal*, 418, Nov. 1993. ISSN 0004-637X, 1538-4357. doi: 10.1086/187102. URL http://arxiv.org/abs/astro-ph/9307024. arXiv:astro-ph/9307024.

[30] N. Produit et al. POLAR, a compact detector for gamma-ray bursts photon polarization measurements. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 550(3):616–625, Sept. 2005. ISSN 0168-9002. doi: 10.1016/j.nima.2005.05.066. URL https://www.sciencedirect.com/science/article/pii/S0168900205012891.

[31] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, New York, NY, 2004. ISBN 978-1-4419-1939-7. doi: 10.1007/978-1-4757-4145-2. URL http://link.springer.com/10.1007/978-1-4757-4145-2.

[32] G. Santin. Normalisation modelling sources. 2007. URL http://geant4.in2p3.fr/2007/prog/GiovanniSantin/GSantin_Geant4_Paris07_Normalisation_v07.pdf.

[33] G. Schilling. *Flash! the hunt for the biggest explosions in the universe*. Cambridge University Press, Cambridge, UK ; New York, NY, 2002. ISBN 978-0-521-80053-2.

[34] C. Simon. Generating uniformly distributed numbers on a sphere, Feb. 2015. URL http://CorySimon.github.io/articles/uniformdistn-on-sphere/.

[35] J. Starmer. The StatQuest Illustrated Guide to Machine Learning!!! 2022.

[36] Z. Tang, J. D. Kanu, K. Hogan, and D. Manocha. Regression and Classification for Direction-of-Arrival Estimation with Convolutional Recurrent Neural Networks. In *Interspeech 2019*, pages 654–658, Sept. 2019. doi: 10.21437/Interspeech.2019-1111. URL http://arxiv.org/abs/1904.08452. arXiv:1904.08452 [cs, eess].

[37] TensorFlow documentation. tf.keras.layers.Dropout | TensorFlow v2.12.0, 2023. URL https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout.

[38] TensorFlow documentation. tf.keras.layers.BatchNormalization | TensorFlow v2.12.0, 2023. URL https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization.

[39] TensorFlow documentation. tf.keras.layers.Conv2D | TensorFlow v2.12.0, 2023. URL https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D.

[40] TensorFlow documentation. Recurrent Neural Networks (RNN) with Keras | TensorFlow Core, 2023. URL https://www.tensorflow.org/guide/keras/rnn.

[41] Y.-H. Wang et al. Localization of Gamma-ray Bursts using the Compton polarimeter POLAR. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 988:164866, Feb. 2021. ISSN 0168-9002. doi: 10.1016/j.nima.2020.164866. URL https://www.sciencedirect.com/science/article/pii/S0168900220312638.

[42] William R. Leo. *Techniques for Nuclear and Particle Physics Experiments: A How-to Approach*. Springer-Verlag, 2nd revisited edition, 1994. ISBN 0-387-57280-5.

[43] B. Zhang. Open questions in GRB physics. *Comptes Rendus Physique*, 12(3):206–225, Apr. 2011. ISSN 1631-0705. doi: 10.1016/j.crhy.2011.03.004. URL https://www.sciencedirect.com/science/article/pii/S1631070511000703.

[44] B. Zhang. *The Physics of Gamma-Ray Bursts*. Cambridge University Press, 2018. doi: 10.1017/9781139226530.

[45] S.-N. Zhang, M. Kole, et al. Detailed polarization measurements of the prompt emission of five gamma-ray bursts. *Nature Astronomy*, 3(3):258–264, Mar. 2019. ISSN 2397-3366. doi: 10.1038/s41550-018-0664-0. URL https://www.nature.com/articles/s41550-018-0664-0. Number: 3 Publisher: Nature Publishing Group.

16 Appendix

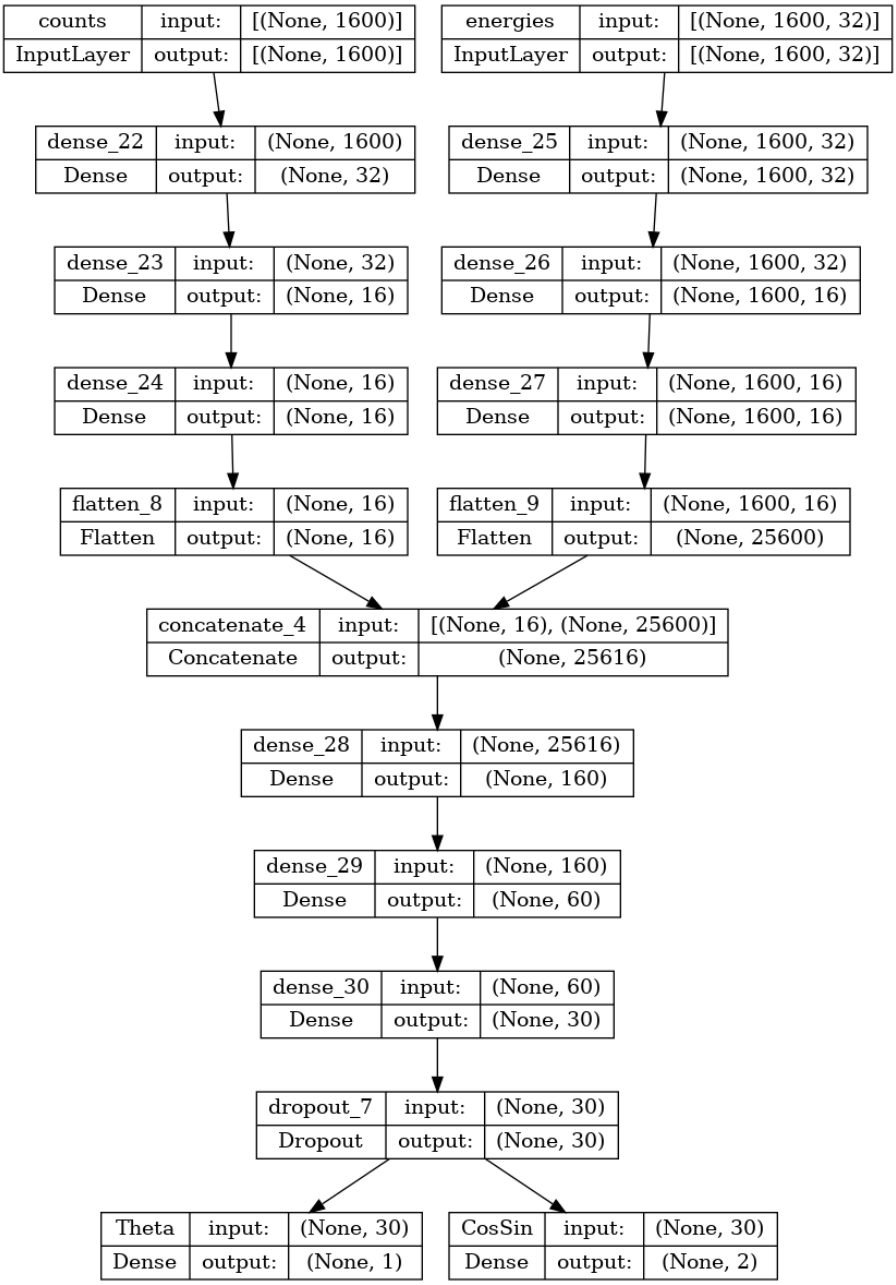


Figure 58: Architecture of the fully connected neural network used on POLAR simulation. This model was trained using the energy histogram by bars which implies that the number of trainable parameters is considerable.

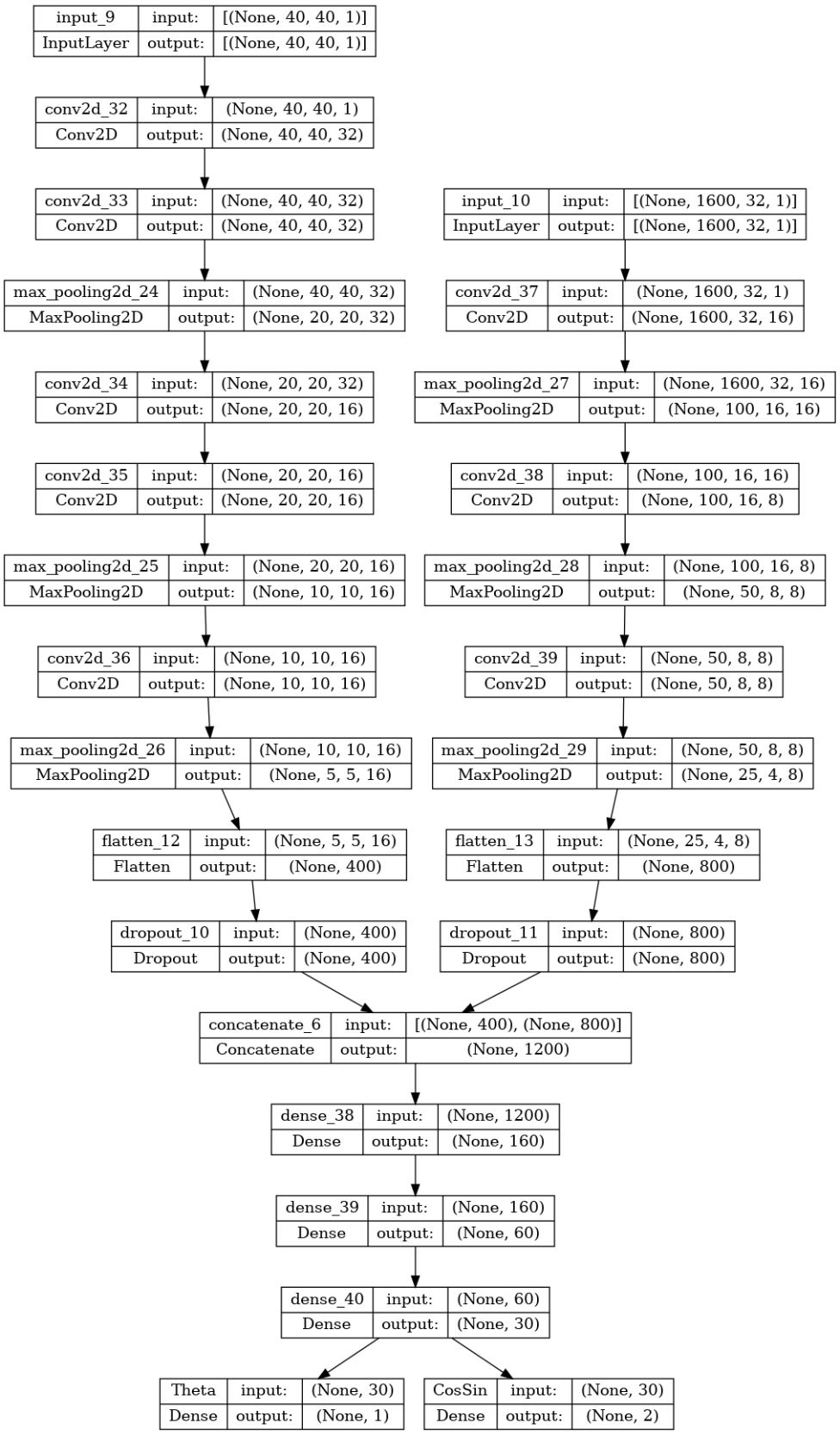


Figure 59: Architecture of the convolutional neural network used on POLAR simulation. This model was trained using the energy histogram by bar.

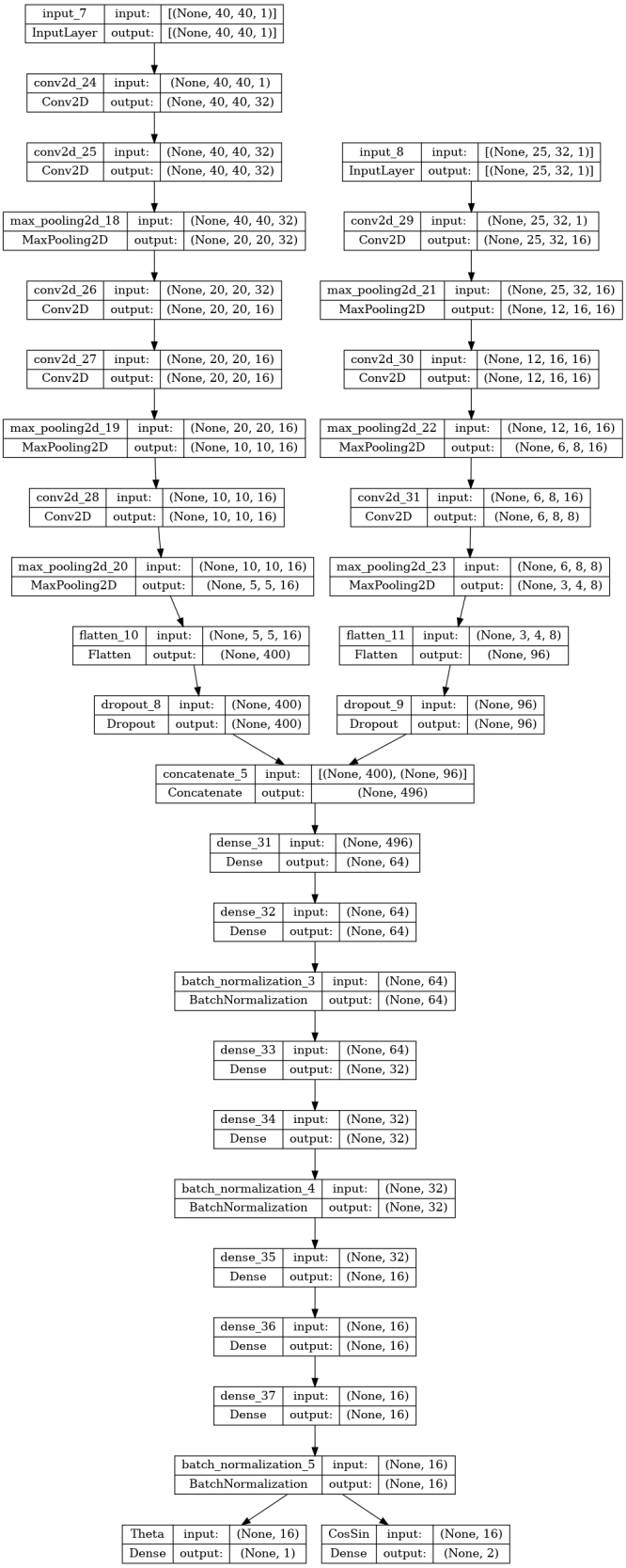


Figure 60: Architecture of the deepest convolutional neural network used on POLAR simulation. This model was trained using the energy histogram by module, which accelerated the training process.