Cherenkov Image Analysis with Deep Multi-Task Learning from Single-Telescope Data

Mikaël Jacquemont

UNIVERSITÉ SAVOIE MONT BLANC

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ SAVOIE MONT BLANC

Spécialité : **STIC Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Mikaël JACQUEMONT

Thèse dirigée par **Patrick LAMBERT,**
codirigée par **Gilles MAURIN**
et coencadrée par **Alexandre BENOIT et Thomas VUILLAUME**

préparée au sein des **Laboratoires LISTIC et LAPP**
dans **l'École Doctorale SISEO**

Cherenkov Image Analysis with Deep Multi-Task Learning from Single-Telescope Data

Thèse soutenue publiquement le **26/11/2020**,
devant le jury composé de :

M. Mathieu JACOBE DE NAUROIS
Directeur de recherche CNRS, Ecole Polytechnique, Président

M. Abelardo MORALEJO OLAIZOLA
Chercheur, Institut de Fisica d'Altes Energies - Espagne, Rapporteur

M. Georges QUENOT
Directeur de recherche CNRS, Université Grenoble Alpes, Rapporteur

Mme Jenny BENOIS-PINEAU
Professeure, Université de Bordeaux, Examinatrice

M. Christian WOLF
Maître de conférences HDR, INSA de Lyon, Invité

M. Patrick LAMBERT
Professeur, Université Savoie Mont Blanc, Directeur de thèse

M. Gilles MAURIN
Maître de conférences HDR, Université Savoie Mont Blanc, Co-Directeur

M. Alexandre BENOIT
Maître de conférences HDR, Université Savoie Mont Blanc, Co-Encadrant

M. Thomas VUILLAUME
Chercheur CNRS, Université Savoie Mont Blanc, Co-Encadrant

"A Bernadette, Michel, et bien sûr Wei"

Remerciements

Je n'imaginais certainement pas il y a 7 ans, lorsque technicien en métrologie j'ai repris des études d'ingénieur, que cette aventure me conduirait à soutenir une thèse, qui plus est à la frontière entre l'apprentissage profond et l'astrophysique ! Le stage effectué en quatrième année d'école d'ingénieur à la Tokyo Metropolitan Université y est sûrement pour quelque chose, aussi je voudrais remercier le Professeur Kubota, le docteur Botzheim et surtout mon ami le docteur Jinseok Woo de m'avoir donné le goût de la recherche et des réseaux de neurones artificiels.

Si je mesure aujourd'hui avec fierté le chemin parcouru, je n'oublie pas que la recherche est avant tout un travail d'équipe. Je remercie en premier lieu les membres du jury qui ont accepté d'évaluer mon travail. Les circonstances particulières ne nous ont pas permis de nous rencontrer physiquement, j'espère que ce n'est que partie remise. Je souhaite également exprimer ma profonde gratitude à mes encadrants : Patrick, Gilles, Alexandre et Thomas. On peut dire que j'ai été bien encadré ! Ils m'ont guidé tout au long de cette aventure hors du commun qu'est la thèse, et j'ai eu plaisir à travailler avec eux.

Je remercie également les collègues du LAPP et du LISTIC pour tous les échanges enrichissants que nous avons eus. Merci à Jean-Claude et Patrick d'avoir permis au monde entier (et surtout au jury) de voir ma prestation de défense ! Merci à l'équipe de MUST, car faire de l'apprentissage profond sans moyen de calcul est impossible. Merci aux équipes administratives des deux laboratoires qui m'ont grandement facilité les nombreux déplacements effectués durant ces trois années. Merci à la Fondation de l'Université Savoie Mont Blanc, et en particulier à Cécile, sa Présidente, pour son soutien. Merci enfin à Giovanni sans qui cette thèse n'aurait été possible.

J'ai également une pensée pour mes amis et ma famille. À mon ami de toujours Rod, à Isa, Mamat, Sofie, aux vieux de la vieille Jérôme, les Seb et Viviane, aux vétérans. À mes frères et soeur Mag, Medhy et Max et bien sûr Graz et Maud. À mes parents Bernadette et Michel. Je n'ai malheureusement pas pu partager avec vous cet accomplissement qu'est la soutenance. Et si certains d'entre vous ont pu suivre la retransmission, je ne doute pas que mes nombreuses tentatives au cours de ces trois ans de vous expliquer mon travail ont fini par payer (n'est-ce pas Rod ?) !

Pour finir, j'ai une pensée toute particulière pour ma chère Wei, qui m'a soutenu, encouragé et supporté sans faille durant ces sept années. Cette thèse est un peu aussi la tienne.

We gratefully acknowledge financial support from the agencies and organizations listed here: www.cta-observatory.org/consortium_acknowledgment. This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 653477, and from the Fondation Université Savoie Mont Blanc. This work has been done thanks to the facilities offered by the Univ. Savoie Mont Blanc - CNRS/IN2P3 MUST computing center and HPC resources from GENCI-IDRIS (Grant 2020-AD011011577) and computing and data processing ressources from the CNRS/IN2P3 Computing Center (Lyon - France). We gratefully acknowledge the support of the NVIDIA Corporation with the donation of one NVIDIA P6000 GPU for this research.

Contents

List of Figures

List of Tables

List of acronyms

- CTA: Cherenkov Telescope Array
- CUDA: Compute Unified Device Architecture
- GEMM: GEneral Matrix to Matrix Multiplication
- GPU: Graphics Processing Unit
- H.E.S.S.: High Energy Stereoscopic System
- IACT: Imaging Atmospheric Cherenkov Telescopes
- LST: Large Size Telescope
- LST1: Large Size Telescope 1
- LSTM: Long Short Time Memory
- MAGIC: Major Atmospheric Gamma-ray Imaging Cherenkov Telescopes
- MSE: Mean Square Error
- ReLU: Rectified Linear Unit
- RF: Random Forest
- SGD: mini-batch Stochastic Gradient Descent
- VERITAS: Very Energetic Radiation Imaging Telescope Array System
- WIMPs: Weakly Interacting Massive Particles

1 Introduction

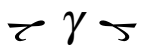
Cosmic rays have been discovered one hundred years ago by V. Hess through the ionization of the Earth atmosphere. They are particles accelerated to tremendous energies by non-thermal processes. Cosmic rays are a proxy to better understand the physics ruling these exotic mechanisms that occur in violent astrophysical phenomena. By studying them, astrophysicists try to solve some of the most interesting mysteries, such as what process can produce such energetic particles, how stars are born or can the fundamental laws of physics be violated. Unfortunately, cosmic rays, that are charged particles, are deviated along their journey to us by interstellar and extragalactic magnetic fields. This makes the identification of their source difficult. However, a fraction of the cosmic rays produced by astronomical objects interact in the close environment of their source with ambient matter, magnetic or radiation fields, and thus produce high-energy photons (from X to gamma). We can observe the gamma rays produced that way to trace the cosmic rays and identify their source. This field of research is called *gamma-ray astronomy*.

Ground-based gamma-ray astronomy is relatively young. It achieved its first success in 1989 with the detection of the Crab Nebula. Very high-energy gamma rays are detected with arrays of Cherenkov telescopes. They capture with optical cameras the Cherenkov light emitted by the shower of secondary particles produced by the relativistic photon entering the Earth atmosphere. The analysis of the data consists first in separating the gamma events from the background noise, *i.e.*, the cosmic rays that are also detected by the telescopes. It consists then in reconstructing the arrival direction and the energy of the primary gamma photon.

Although several Cherenkov arrays have been built since 1989, the Cherenkov Telescope Array (CTA) is the next generation observatory for the study of the high-energy universe. Composed of two sites for more than hundred telescopes when completed, CTA will dramatically improve the sensitivity compared to current arrays, such as the High Energy Stereoscopic System (H.E.S.S.). As a counterpart, it will generate a tremendous amount of data each year, making the state-of-the-art analysis methods complicated to use due to timing constraints. The first telescope of the northern site, the Large-Sized Telescope 1 (LST1), has been inaugurated in fall 2018, and the first exploitable data were made available during the writing of this thesis. The LST1 has been designed to opti-

mize its sensitivity in the lower part of the very high-energy gamma-ray spectrum. It is particularly relevant to study extragalactic and transient phenomena. The LST1 is also very important as it will allow the astrophysicists to probe their analysis tools. Indeed, as ground truth is impossible to obtain for real data, analysis models are developed with simulated data.

Besides, deep learning has achieved great successes in the computer vision field over the past decade. Inspired by the biological brain in their composition and their way of learning, neural networks have become the leading approach to solve many problems, such as object detection or semantic segmentation. These tasks are crucial for real-life applications such as autonomous vehicles or medical imaging. Simultaneously, the interpretability and explainability of the models have been improved to provide the transparency needed by these fields and many others. It is still an active field of research that shows its applicability in a large variety of domains.



The GammaLearn project, founded by the European H2020 project Asterics and the Fondation de l'Université Savoie Mont Blanc, aims to explore deep learning techniques for gamma astronomy. GammaLearn includes the Laboratoire d'Annecy de Physique des Particules (LAPP) and the Computer Science, Systems, Information and Knowledge Processing Laboratory (LISTIC). It is involved in the CTA consortium that comprises more than 1500 scientists and engineers from 31 countries.

This thesis, prepared in the framework of GammaLearn, proposes a novel deep learning approach for the analysis of LST1 data. In particular, it introduces a new deep multi-task learning architecture that outperforms a widespread method on simulated data for gamma event reconstruction from IACT data in the context of single-telescope analysis. This architecture is also used to analyze the first exploitable real data produced by the LST1. In addition, this thesis presents an original method to apply the convolution to any kind of pixel organization without preprocessing the data, in particular to the hexagonal grid of the LST camera. Finally, this thesis proposes a software framework to ease the deep learning procedure with CTA data, ensuring the traceability and reproducibility of the experiments.

It is worth noticing that deep learning algorithms are generally costly during their training phase, and then require specific computing resources. The experiments presented in this thesis have been carried out with the help of high-performance computing centers: MUST[1], CC-IN2P3[2] and Jean Zay[3].

[1] https://lapp.in2p3.fr/spip.php?article103
[2] https://cc.in2p3.fr/en/qui-sommes-nous/le-cc-in2p3/
[3] http://www.idris.fr/eng/jean-zay/cpu/jean-zay-cpu-hw-eng.html

1.1 Gamma astronomy

Gamma-ray astronomy is the astronomical observation of the most energetic photons (above 100 keV) of the universe, produced by non-thermal processes. They mainly result from the interaction of accelerated hadrons or leptons (mainly electrons[4]), denoted cosmic rays, with ambient matter or magnetic or radiation fields.

Hadrons are charged particles made of quarks, such as protons, light and heavy atomic nuclei, anti-protons, neutrons, etc. In the following I only consider protons, as they represent ∼90% of the hadronic cosmic rays [Mewaldt, 1994].

In this context, this thesis focuses on very high-energy gamma rays, with energies > 20 GeV, that will be detected by CTA.

1.1.1 Science cases

As they travel in straight lines, on the contrary to charged cosmic rays, gamma-ray photons allow for an accurate determination of their origin. They are thus a pertinent indirect probe of cosmic rays for different science cases.

Astrophysical Motivations. One of the major objectives of gamma astronomy is to study the sources of cosmic rays and the exotic mechanisms by which they are accelerated to relativistic energies. In addition, the study of their propagation in the interstellar environment is crucial to understand the role of accelerated particles in star formation and galaxy evolution.

Cosmic rays are accelerated in violent phenomena, such as supernova remnants, gamma-ray bursts, active galactic nuclei, pulsar wind nebulae or binary systems. The three main acceleration mechanisms are the diffusive shock acceleration (first order Fermi acceleration) in shock waves (as in supernova remnants), stochastic processes in turbulent environments (second order Fermi acceleration) and magnetic reconnection[5]. The observation of the gamma rays produced in these extreme environments can help understand their composition. For instance, in supernova remnants, the flux of gamma rays produced by protons is proportional to the density of accelerated protons times the density of the ambient matter they interact with. The flux of electronic gamma rays directly relates to the density of electrons and ambient photons (cosmic microwave background, infrared, optical or X-ray) [Hinton and Hofmann, 2009]. Determining the proportion of gamma rays produced by accelerated protons or electrons in a defined source is an active field of research.

The extragalactic background light reflects the history of the universe and is crucial to understand the processes giving birth to stars and the evolution of galaxies. Galactic[6] and other foreground sources generally prevent a direct measurement of this light. When they pass though the extragalactic background light, gamma rays can interact with its photons, annihilating into pairs of electron-positron. This process attenuates the gamma-ray spectrum of the extragalactic sources observed above a critical energy depending on

[4] In this thesis, 'electrons' stands for 'electrons and positrons'. Other leptons are the muon and the tau and corresponding antiparticle and neutrino.

[5] Two magnetic fluxes of opposite polarity encounter each other.

[6] Related to the Milky Way galaxy

their redshift[7]. This attenuation can be measured to place constraints on the population of active galaxies in the early universe [Funk, 2015].

Exploring New Physics. The observation of extraterrestrial gamma rays is also relevant to explore the frontiers of physics. In the standard cosmological model, the mass density of the universe is dominated by dark matter. One of the theories elaborated states that dark matter is composed of weakly interacting massive particles (WIMPs) that produce standard model particles when they self-annihilate, including gamma rays in the final state. The J-factor [Evans et al., 2016] describes the dark matter distribution in a particular astrophysical system, and helps to determine the expected strength of the gamma-ray signal emitted by this system seen from Earth. The study in the gamma energy band of astrophysical objects that we think are dominated by dark matter could help confirm or deny the theory. The analysis of 10 years of data from the observation of the active galactic center by the H.E.S.S. observatory (see Section 1.2 for details on H.E.S.S.) has already placed constraints on two self-annihilation channels predicted by the WIMP theory [Abdallah et al., 2016]. The observation of five dwarf galaxies that are believed to be dark matter dominated also placed constraints on a third self-annihilation channel [Abdalla et al., 2018].

Besides, the Lorentz invariance is a set of frameworks that support fundamental physics, such as the constancy of the propagation speed of photons. Quantum gravity models suggest that it may be violated such that the speed depends on the photon energy [Funk, 2015]. To search for this variation, gamma-ray bursts and active galactic nucleus flares are good candidates as they are distant objects with well-located peak and short emission in a wide energy range.

1.1.2 Gamma-Ray Production Mechanisms

Very high-energy gamma rays are produced by accelerated electrons and protons (cosmic rays) interacting with ambient matter or electromagnetic fields. They are also possibly produced by dark matter annihilation. In this thesis, we are interested in the detection of high-energy gamma rays and the reconstruction of their associated physical quantities. However, the sole observation of a gamma ray does not allow determining its production mechanism.

The following provides an overview of the gamma-ray emission processes and sources.

1.1.2.1 Gamma-Ray Emission

Electronic Emission. Accelerated electrons mainly produce very high-energy gamma rays through the inverse Compton process. Relativistic electrons up-scatter low-energy ambient photons (cosmic microwave background, infrared or optical photons) to gamma rays, losing a fraction of their energy. When passing through a ionized interstellar medium, relativistic electrons can also produce gamma rays via bremsstrahlung (*i.e.*, deceleration

[7]The redshift is an increase in the wavelength of the light emitted by astronomical objects. It is caused by the Doppler effect due to the object moving away from the observer, the expansion of the universe or strong gravitational fields.

radiation). They interact with the coulomb field of the plasma nuclei and are decelerated, emitting a photon.

Hadronic Emission. Very high-energy gamma rays are also produced through the interaction of relativistic protons with interstellar material. This collision generates pions. Neutral pions, in particular, are unstable particles, and decay with a lifetime of 8.4×10^{-17}s into 2 gamma rays (with a probability of 89%).

1.1.2.2 Some Gamma-Ray Sources

Galactic Sources

Supernova Remnants. A dying massive star collapses, then explodes in a supernova in a very short time (some ms). The shell of this supernova remnant, bounded by an expanding shock wave, is a potential source of gamma radiation due to proton collisions.

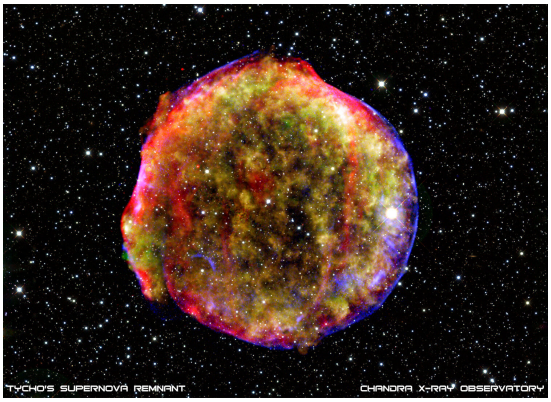


Figure 1.1 – Tycho's supernova remnant. Credit: Chandra X-ray Observatory.

Pulsars and Pulsar Wind Nebulae. A pulsar is a rotating neutron star created by a supernova explosion that emits electromagnetic radiation out of its magnetic poles. Its magnetic field axis is misaligned with its rotating axis making its radiation appear pulsed to us. A pulsar wind nebula is a nebula powered by the winds generated by its central pulsar. It is found inside the shell of a supernova remnant.

Figure 1.2 – Crab Nebula Pulsar, the first very high-energy gamma-ray source detected. Credits: J. Hester (ASU), CXC, HST, NRAO, NSF, NASA.

Compact Object Binary Systems. A compact binary system consists of two compact stellar remnants (neutron stars, black holes or white dwarfs) orbiting around a common center of gravity. Particles are accelerated in jets produced by accretion of matter.

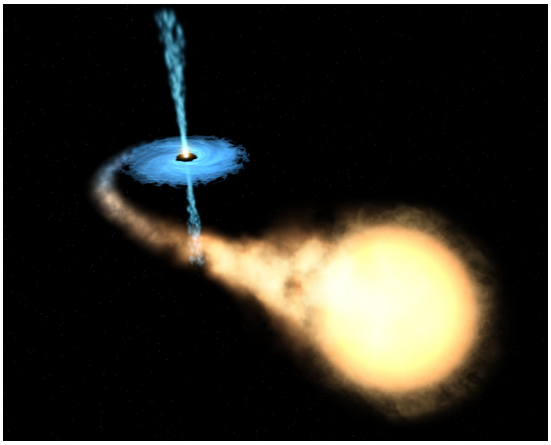


Figure 1.3 – Artistic impression of a binary system. Credits: F. Mirabel, ESA, NASA.

Extragalactic Sources

Gamma-Ray Bursts. Gamma-ray bursts are extremely energetic explosions that last up to a few hours. Most of detected gamma-ray bursts are extragalactic. The nature of their source is an active field of research.

Figure 1.4 – Illustration of a gamma-ray burst. Credits: NASA, ESA and M. Kornmesser.

Active Galactic Nuclei. An active galactic nucleus is a compact region at the center of a galaxy. A supermassive black hole accretes matter and powers jets of accelerated particles.

Figure 1.5 – Composite image of Centaurus A, revealing the lobes and jets emanating from the active galaxy's central black hole. Credits: ESO/WFI (Optical); MPIfR/ESO/APEX/A.Weiss et al. (Submillimetre); NASA/CXC/CfA/R.Kraft et al. (X-ray). Modified.

1.1.3 Gamma-Ray Detection

Taking into account all cosmic particle types, including gamma rays, the energy spectrum of detected very high-energy cosmic rays can be described by a power law $\frac{dN}{dE} \sim E^{-\Gamma}$ with N the number of cosmic rays detected, E the energy and Γ the spectral index. As illustrated in Figure 1.6, $\Gamma = 2.7$ above the *knee* situated at ~ 1 PeV, and $\Gamma = 3.1$ below. The number of detectable particles decreases dramatically with their energy, from $1\,cm^{-2}\,s^{-1}$ at ~ 100 GeV to $1\,m^{-2}\,yr^{-1}$ at ~ 3 PeV. Besides, most of these cosmic rays are protons. The ratio of the gamma rays, the particles we are interested in, is typically lower than $1/1000$. Moreover, during their propagation, very high-energy gamma rays can be absorbed by pair production on ambient matter or photons. This absorption is particularly important when observing extragalactic sources.

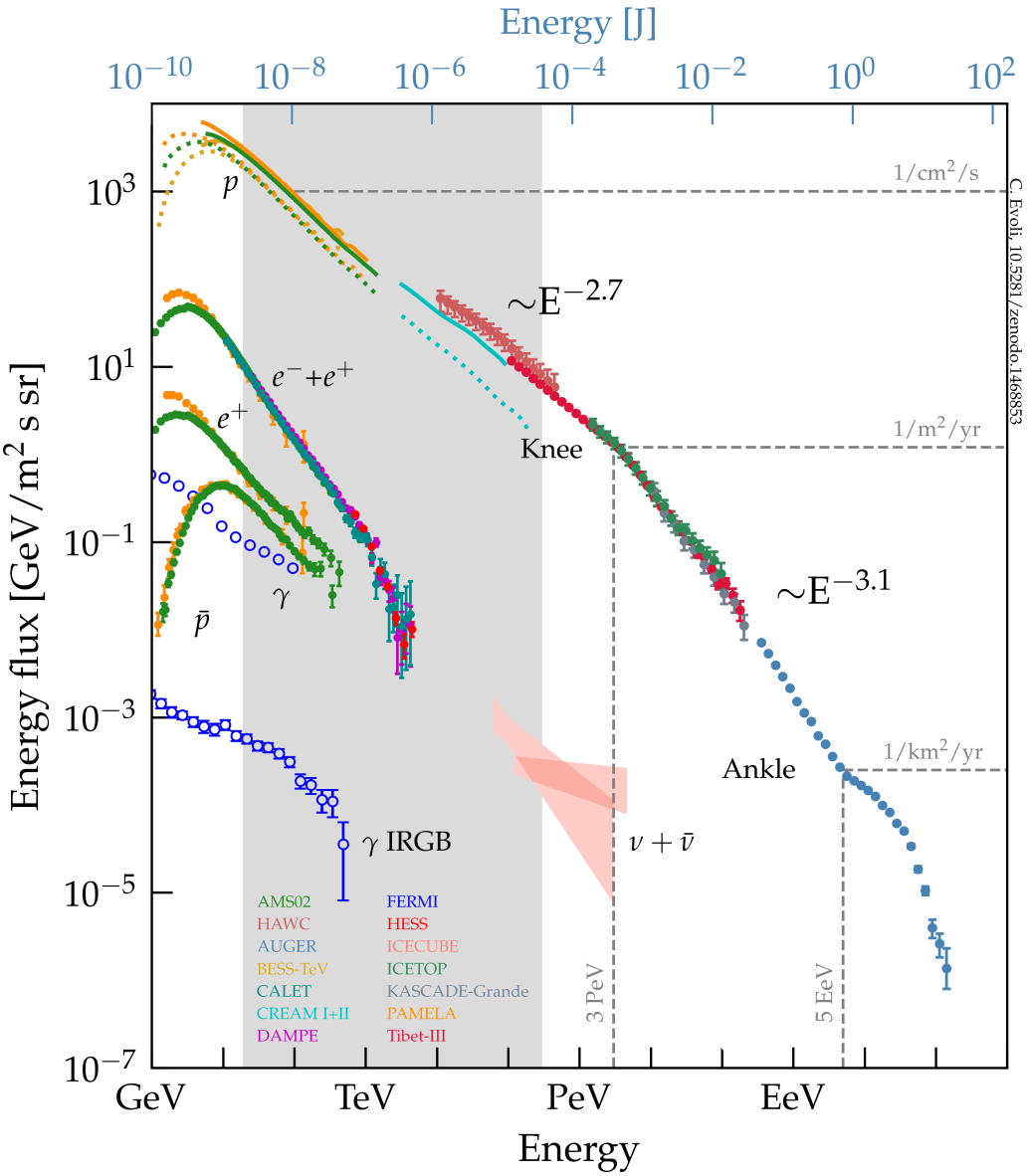


Figure 1.6 – The cosmic-ray all-particle energy spectrum, including gamma rays, measured by several instruments. In this thesis, we focus on the energy ranging from 20 GeV to 300 TeV (in gray) corresponding to the energy range of CTA. Source [Evoli, 2018], modified.

As the Earth atmosphere absorbs the gamma rays, they can only be directly detected by space-based instruments. On another hand, ground-based instruments detect the secondary products of the interaction between the gamma-ray particles and the dense matter of the atmosphere.

Space-Based Instruments. Their sensor relies on the principle of pair creation. When a gamma ray interacts with matter, it produces an electron-positron pair. For the Fermi-LAT, the main gamma-ray observation satellite, the detector consists of a tracker to measure the tracks of the electron-positron pair, a calorimeter to determine the pair energy and a anti-coincidence detector to suppress the background generated by charged particles. However, the gamma rays cannot be focused. The photon collection area is at most the surface of the detector ($\sim 0.65\,m^2$ for the effective collection area of the Fermi-LAT). While it is sufficient to detect low to high-energy gamma rays, other techniques are needed for very high-energy ones. For example, the Crab Nebula, a bright source, emits $\approx 10^{-7}$ photons $m^{-2}\,s^{-1}$ at ~ 1 TeV [Hillas et al., 1998]. At energies above ~ 10 GeV, collection areas of the order of $10^5\,m^2$ are needed.

Ground-Based Instruments. Ground-based instruments offer collection areas of the order needed to detect very high-energy gamma rays. They rely on the detection of secondary particles produced by the gamma ray interacting with the atmosphere.

Imaging Atmospheric Cherenkov Telescopes (IACTs) detect the Cherenkov light emitted by the secondary particles traveling through the atmosphere. Section 1.2 presents in details how IACTs work. The energy threshold of current IACTs is typically ~ 50 GeV. Above some tens of TeV, their detection area becomes too small compared to the flux of detectable gamma rays.

On another hand, water Cherenkov detectors, such as the former MILAGRO[8] and the High-Altitude Water Cherenkov Gamma-Ray Observatory[9] (HAWC), detect the secondary particles on the ground. They measure the Cherenkov light emitted by the secondary particles when they pass through water detectors. As water Cherenkov detectors need the secondary particles to reach the ground, their energy threshold is higher than the one of IACTs (~ 100 GeV). However, they have a wider field of view, and on the contrary to IACTs that require dark nights, water Cherenkov detectors can operate continuously.

In the following of this thesis, I focus on IACTs that are the components of CTA.

1.2 Imaging Atmospheric Cherenkov Telescopes

As the atmosphere is opaque to gamma rays, IACTs rely on an indirect observation method. When a particle of very high energy enters the atmosphere, it interacts with its dense matter producing a particle shower, a cascade of secondary particles with lesser and lesser energy. This shower emits a light in the visible to ultraviolet spectrum, the Cherenkov radiation, that is recorded by the telescope's camera.

[8] https://web.archive.org/web/20121129084657/http://www.lanl.gov/milagro/index.shtml
[9] https://www.hawc-observatory.org/

1.2.1 Particle showers

The shower development depends on the particle type. Gamma photons and cosmic electrons, which interact via the electromagnetic force, produce electromagnetic showers and protons, which interact via the nuclear force, produce hadronic showers.

Electromagnetic Shower Production. When a gamma photon enters the atmosphere, it interacts with the magnetic field of nuclei or with electrons, and produces a pair of electrons. These electrons produce then gamma photons via bremsstrahlung, losing energy. In the simple model of Heitler [Bethe and Heitler, 1934] shown in Figure 1.7, we assume that any gamma photon has produced a pair of electrons, each having half the energy of the photon, after traveling a radiation length X_0[10] in the atmosphere. In the same way, we assume that any electron has produced a gamma photon after a radiation length X_0. The development of this shower of particles stops after X_{max} when their energy reaches the critic energy E_c[11], and

$$X_{max} = X_0 \ln\left(\frac{E_0}{E_c}\right). \tag{1.1}$$

The number of electrons produced is then $N_{max} = E_0/E_c$, with E_0 the energy of the primary particle.

The lateral development of the electromagnetic showers comes from the diffusion of the electrons in the coulomb field of the nuclei. Due to the small diffusion angle of the electrons, electromagnetic showers are very collimated. And thanks to the large number of secondary particles, the showers have a symmetry of revolution around their incident direction.

Noteworthy, electronic cosmic rays also produce electromagnetic showers via the same mechanism. However, in this thesis we are only interested by electromagnetic showers produced by gamma rays.

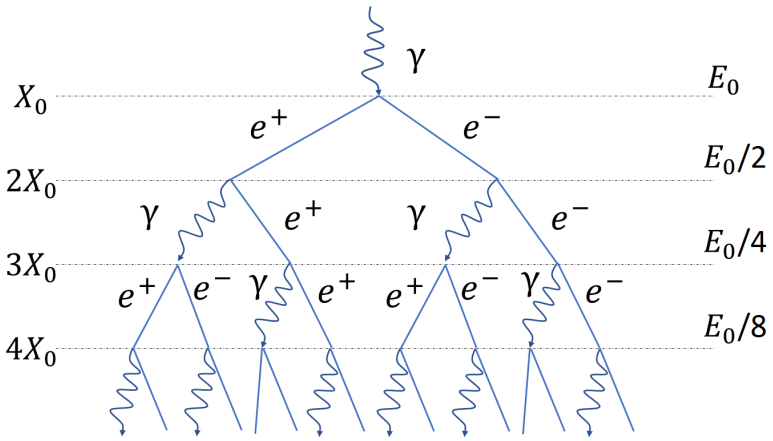


Figure 1.7 – Heitler model of electromagnetic shower. The primary gamma photon produces a pair of electrons after traveling a radiation length of X_0. The electrons lose energy via bremsstrahlung, producing after another X_0 a gamma photon that in turn decomposes into a pair of electrons, and so on so forth until all the particles generated in the shower reach the critic energy.

[10] In dry air, $X_0 = 36.7\,g\,cm^{-2}$.
[11] In the air, $E_c = 84.2MeV$.

Hadronic Shower Production. A proton entering the atmosphere also produces a shower, but a slightly different one. It interacts with oxygen and nitrogen nuclei. During its primary interaction, it produces mesons (pions and kaons), neutrons and secondary protons, which in turn interact with the nuclei, as shown in Figure 1.8. The π^0 (a third of the created pions) have a very short lifetime and disintegrate almost instantaneously in two gamma photons. Thus, electromagnetic sub-showers develop inside the hadronic one. The charged pions decay into muons.

Showers generated by protons are often larger than electromagnetic ones, and have a more random lateral development, as shown in Figure 1.9. In this thesis, we consider showers generated by protons as background noise.

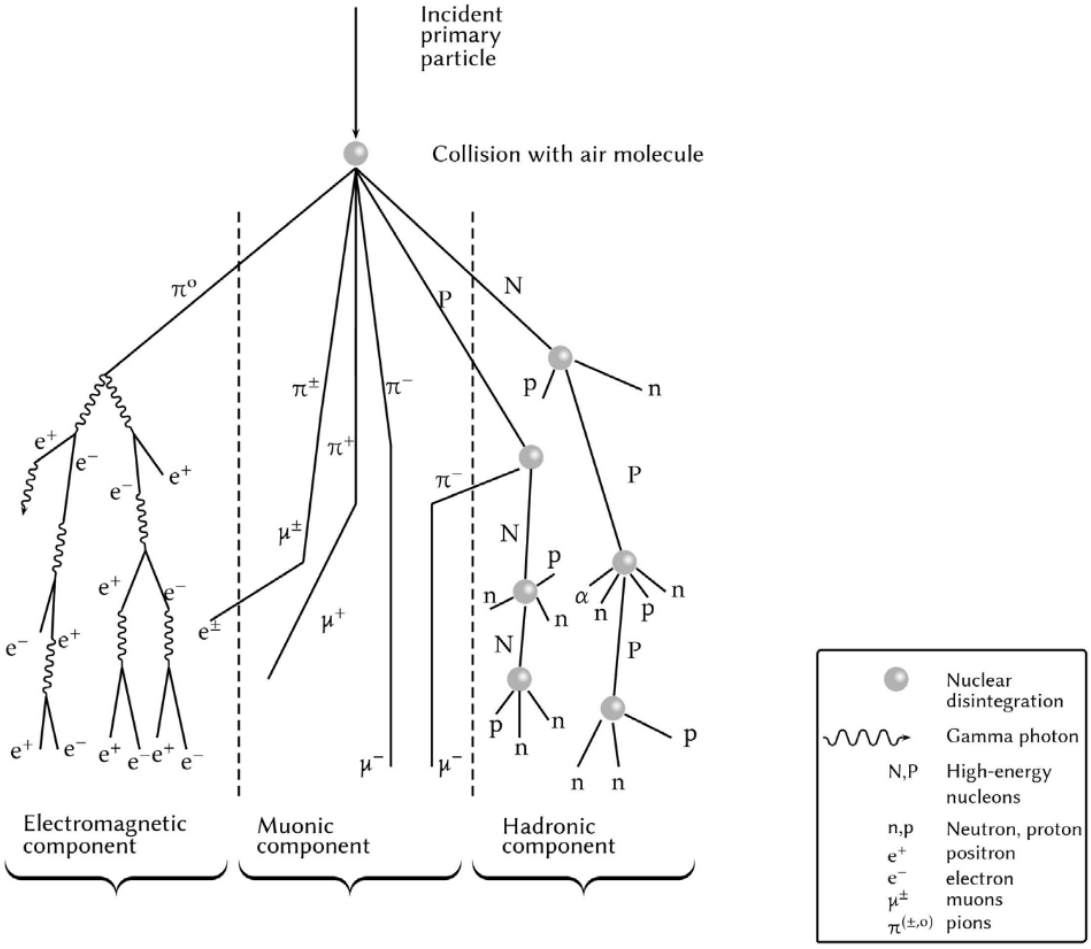


Figure 1.8 – Hadronic shower development. Source: [Barrantes et al., 2018]. The development of the showers initiated by protons entering the atmosphere is more complex than the ones produced by gamma rays. When a proton collides with a molecule of the atmosphere, it decomposes into secondary particles, and pions emerge. A third of these are neutral pions that decay instantaneously into two gamma rays, generating electromagnetic sub-showers. The charged pions decay to muons. The cascade continues until the secondary particles reach their critic energy.

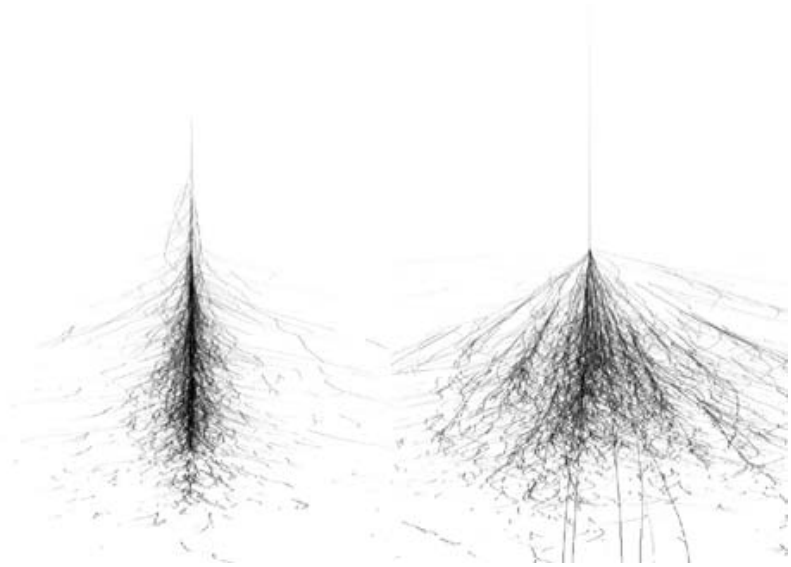


Figure 1.9 – Comparison of an electromagnetic (left) and a hadronic (right) shower. Source: [Völk and Bernlöhr, 2009]. The gamma shower is more collimated and roughly symmetric about the direction of the primary photon. On the other hand, the shower generated by a proton is irregular and contains electromagnetic sub-showers.

1.2.2 Cherenkov Radiation

When a charged particle passes through a dielectric medium at a speed v greater than the velocity of light in that medium of refractive index n, it emits an electromagnetic radiation, also known as Cherenkov radiation. As the source of this radiation has a greater speed than the radiation itself, it produces a conic shock front, at visible to ultraviolet wavelengths. The angle of the light rays with respect to the particle propagation direction is:

$$cos(\theta) = \frac{c}{vn} \tag{1.2}$$

where c is the speed of light in vacuum.

For a shower generated by a 1 TeV gamma ray, the X_{max} corresponds to an altitude of 10 km, and the front of the Cherenkov light is about 120 m wide (corresponding to $\theta \sim 0.9°$) around the shower axis at an altitude of 2000 m [Funk, 2015]. At this altitude, ~ 100 photons m^{-2} reach the ground, dropping to $\sim 10 m^{-2}$ at sea level.

The particles produced in the shower emit Cherenkov radiation until a threshold energy determined by $v > c$. The threshold energy is then given by:

$$E_{th} \approx \frac{m_0 c^2}{\sqrt{2\delta}} \tag{1.3}$$

with m_0 the rest mass of the particle and $\delta = n - 1$. In the case of the aforementioned gamma ray, $E_{th} \sim 23$ MeV for the electrons produced in the shower.

The Cherenkov signal of a particle shower seen by the telescope is very brief (some *ns*).

1.2.3 IACT Principle.

As illustrated in Figure 1.10, IACTs observe the Cherenkov radiation emitted by the shower. They are composed of a large parabolic or spherical mirror and a high-sensitivity camera. The mirror collects and focuses the light on a camera, usually made of photo-multipliers. The latter transforms the Cherenkov photons into photoelectrons to form an image. The gamma shower then appears as an ellipsoid. As the number of Cherenkov photons is about proportional to the primary gamma-ray energy, the effective area of the mirror, the sensitivity of the camera and the altitude of the telescope (see Section 1.2.2) determine its threshold energy [Völk and Bernlöhr, 2009].

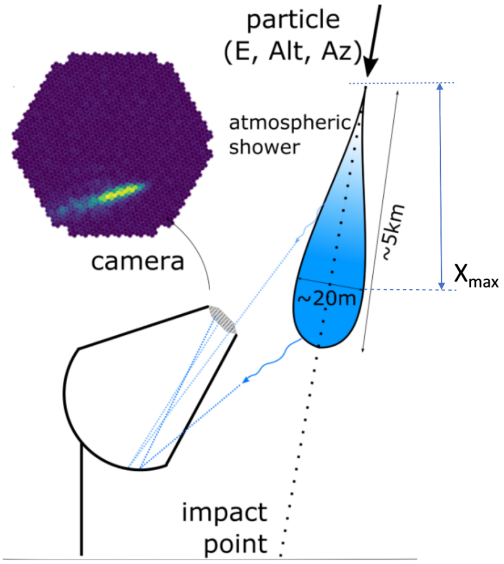
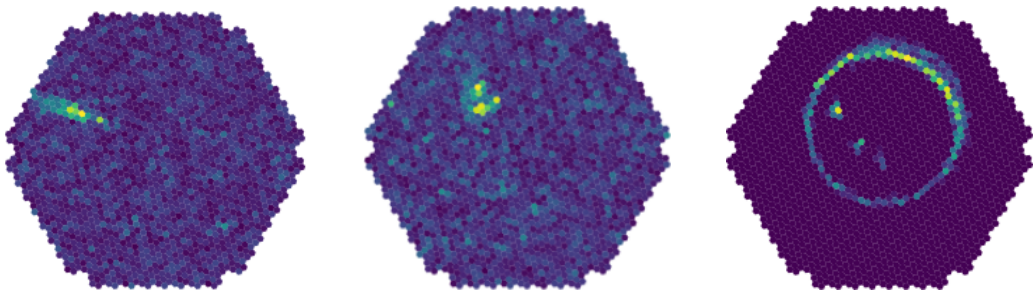


Figure 1.10 – Imaging Atmospheric Cherenkov Telescope. Credit: T. Vuillaume.

The shower produced by a proton also emits a Cherenkov light that is detected by IACTs. Depending on the energy of the primary particle, gamma photons and protons can produce very similar images in the telescope camera. Besides, muons, as secondary particles of protons, produce a characteristic ring in the image. Figure 1.11 shows a high-energy gamma image in the LST1 camera, a low-energy proton and a proton shower that includes a muon ring.



(a) Gamma event with an energy of 0.5 TeV.

(b) Proton event with an energy of 27 GeV.

(c) Proton event composed of several sub-showers and a muon ring.

Figure 1.11 – Comparison of the images in the LST camera produced by a high-energy gamma, a low-energy proton and a proton shower containing muons.

1.3 The Cherenkov Telescope Array

Since the first IACT, the Whipple observatory constructed in 1968 that detected the first very high-energy source [Weekes et al., 1989], many others have been built, mainly as arrays of telescopes to make the most of the stereoscopic techniques. Stereoscopic observations are useful for background suppression, in particular to reject showers generated by protons that contain muons detected by individual telescopes. They are also important to improve the angular resolution, *i.e.*, to determine the origin of the primary gamma ray.

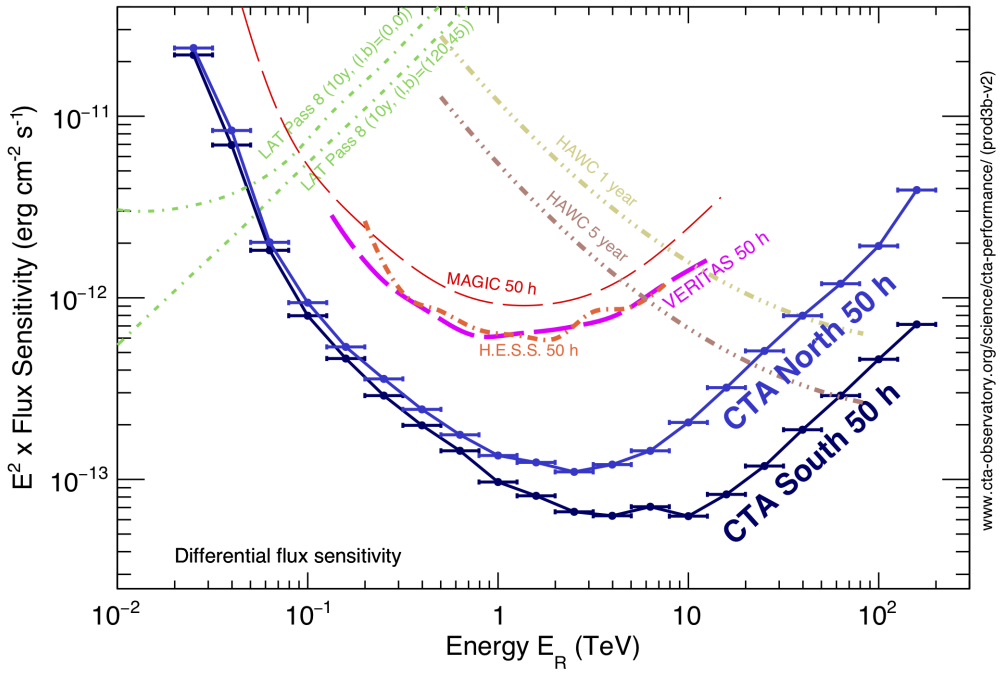
The High Energy Stereoscopic System (H.E.S.S.[12]) is located in Namibia. It is composed of four smaller ones (with mirror area of $107\,m^2$) and one large telescope with a mirror area of $\sim 600\,m^2$ added during a second construction phase. The large telescope allowed lowering the energy threshold from $\sim 100\,\text{GeV}$ originally to $\sim 30\,\text{GeV}$. Since the inauguration of its Phase I (the four smaller telescopes) in 2004, H.E.S.S. has detected 94 gamma-ray sources. As a side note, the gamma astronomy group at LAPP is also involved in the H.E.S.S. collaboration.

The Major Atmospheric Gamma-ray Imaging Cherenkov Telescopes (MAGIC[13]) is located in the Canary island of La Palma. It is a two telescope observatory designed to detect gamma rays below 100 GeV, down to $\sim 30\,\text{GeV}$, to find new classes of sources, in particular powerful extragalactic sources and transient phenomena. In 2008, MAGIC detected very high-energy gamma rays from a quasar, the most distant source observed at these energies [Albert et al., 2008a].

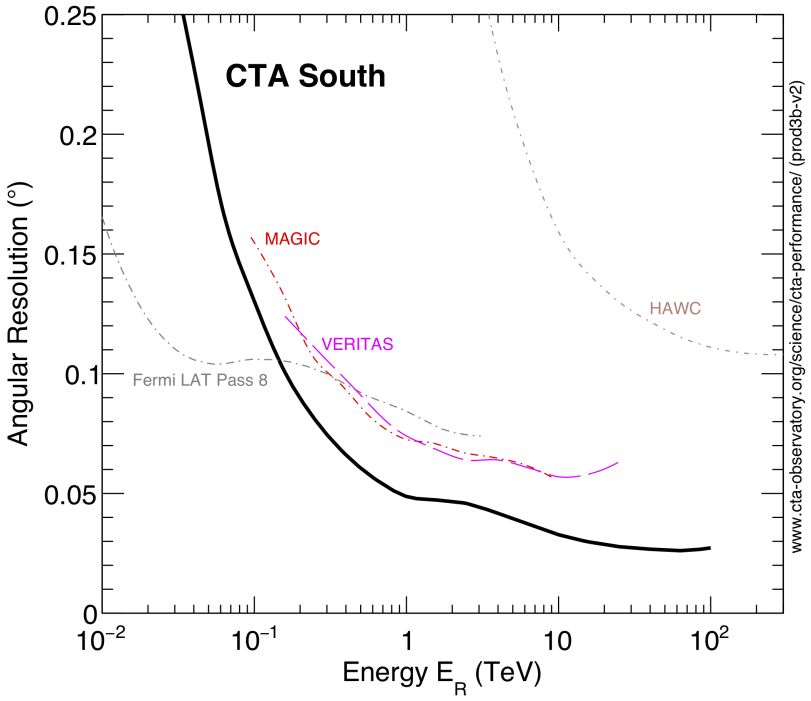
The Very Energetic Radiation Imaging Telescope Array System (VERITAS[14]) is similar to H.E.S.S. Phase I. It is also composed of four telescopes with mirror area of $106\,m^2$. It has comparable sensitivity and performance.

Built on the technology of current detectors, the Cherenkov Telescope Array (CTA)[15] is the next generation of IACTs. As illustrated in Figure 1.12, it will improve sensitivity by a factor of 10 compared to current ground-based observatories while also increasing accuracy in gamma-ray detection, and extending the spectral (energy) range up to 300 TeV. To achieve these improvements, CTA will be composed of 118 telescopes with very high-speed and high-sensitivity cameras (telescope readout event rate in kHz range$[0.6, 10]$) distributed on two sites, one in the north hemisphere and the other in the south one. CTA is in construction, and is expected to be completed in 2025. When in full operation, it is expected to detect ~ 12 gamma events and $\sim 13k$ proton events per second [Paz Arribas, 2017]. It will then produce a tremendous amount of **210 PB** of raw data per year to be analyzed in real time and then reduced and compressed to 3 PB before archiving. Moreover, thanks to an improving knowledge of the telescopes and thus better analysis algorithms, all the data already acquired will be reprocessed every year.

[12] http://www.mpi-hd.mpg.de/hfm/HESS/pages/about/telescopes
[13] https://magic.mpp.mpg.de
[14] http://veritas.sao.arizona.edu
[15] https://www.cta-observatory.org/



(a)



(b)

Figure 1.12 – CTA estimated differential sensitivity (a) and angular resolution (b) compared to other instruments. Lower is better. Source: www.cta-observatory.org.

1.3.1 The Array

CTA will be composed of telescopes of three different sizes corresponding to different sensitivities: the Large-Sized Telescopes (LSTs), the Medium-Sized Telescopes (MST) and the Small-Sized Telescopes (SST).

Large-Sized Telescope (LST). The LST has the largest mirror ($\sim 400\,m^2$) and the smallest field of view (4.3°). It has been designed to detect gamma rays with an energy between 20 GeV and 3 TeV, which is especially interesting for the study of transient phenomena such as gamma-ray bursts recently observed for the first time by IACTs [Abdalla et al., 2019b, Acciari et al., 2019]. Noteworthy, its camera, the LSTCam [Ambrosi et al., 2013b], is made of 1855 photomultipliers tubes organized in a hexagonal grid, resulting in hexagonal pixels. Besides, the LSTCam has a roughly hexagonal shape.

Medium-Sized Telescope (MST). The MST has a smaller mirror ($\sim 90\,m^2$) but a larger field of view ($\sim 8°$). It is designed to detect mid-energy particles in the range [80 GeV, 50 TeV] with an optimal sensitivity between 150 GeV and 5 TeV. It is worth noticing that the MST will be equipped with two different cameras (NectarCam [Glicenstein et al., 2013] and FlashCam [Pühlhofer et al., 2012]), both with hexagonal pixels (as the LSTCam), respectively 1855 and 1764.

Small-Sized Telescope (SST). The SST is the smallest telescope with the largest field of view ($\sim 10°$). It is designed to improve the sensitivity of CTA for the highest-energy particles in the range [5, 300] TeV, which come from our own galaxy. Its camera, the ASRTICam, is made of silicon photomultipliers producing square-pixel images (2368), but with an unconventional shape.

Figure 1.13 illustrates the images produced by the different cameras of CTA with random data drawn from a normal distribution.

Array Layouts. The telescopes will be spread between two sites in the northern and the southern hemispheres to improve the observation of extragalactic and galactic sources. This way, CTA will virtually be able to cover the entire sky, as illustrated in Figure 1.14.

The northern site of CTA is located in the Canary island of La Palma, at $2,200\,m$ in altitude, on the same site as MAGIC. It will be composed of 4 LSTs and 15 MSTs to cover an energy range from 20 GeV to 20 TeV, as the northern hemisphere allows for a better observation of extragalactic sources. Figure 1.15 illustrates the future CTA site in La Palma that is in construction. However, the Large-Sized Telescope 1 (LST1 [Ambrosi et al., 2013a]), the first on-site prototype, is already built, and started to produce exploitable data during the writing of this thesis.

The southern site of CTA is located in Paranal, in Chile. As the southern hemisphere is more suitable to observe our galaxy and the highest-energy gamma rays detected from Earth come from it, in addition to 4 LSTs and 25 MSTs, the Paranal site will also consists of 70 SSTs. It will be able to detect particles between 20 GeV and 300 TeV.

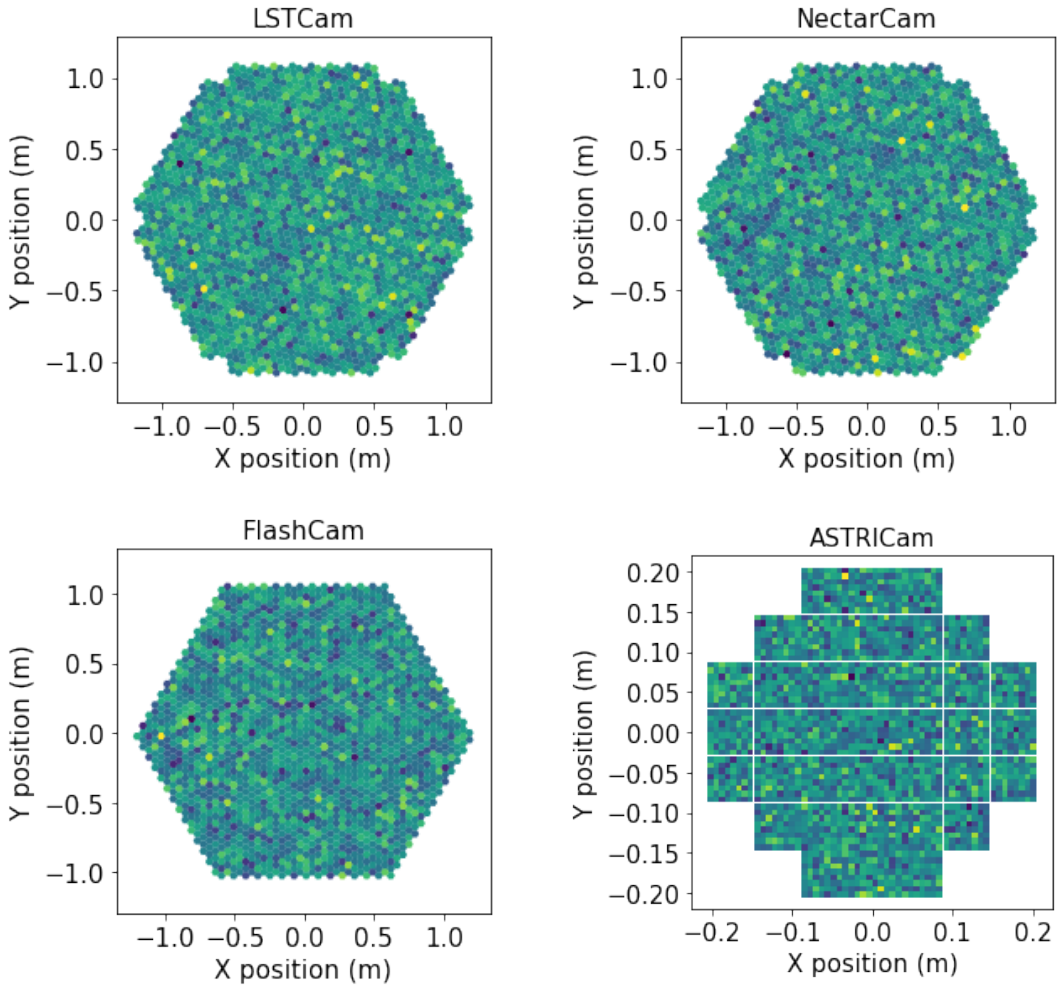


Figure 1.13 – Images produced by the different cameras used in CTA with random data drawn from a normal distribution. The LSTCam, NectarCam and FlashCam produce images with hexagonal pixels.

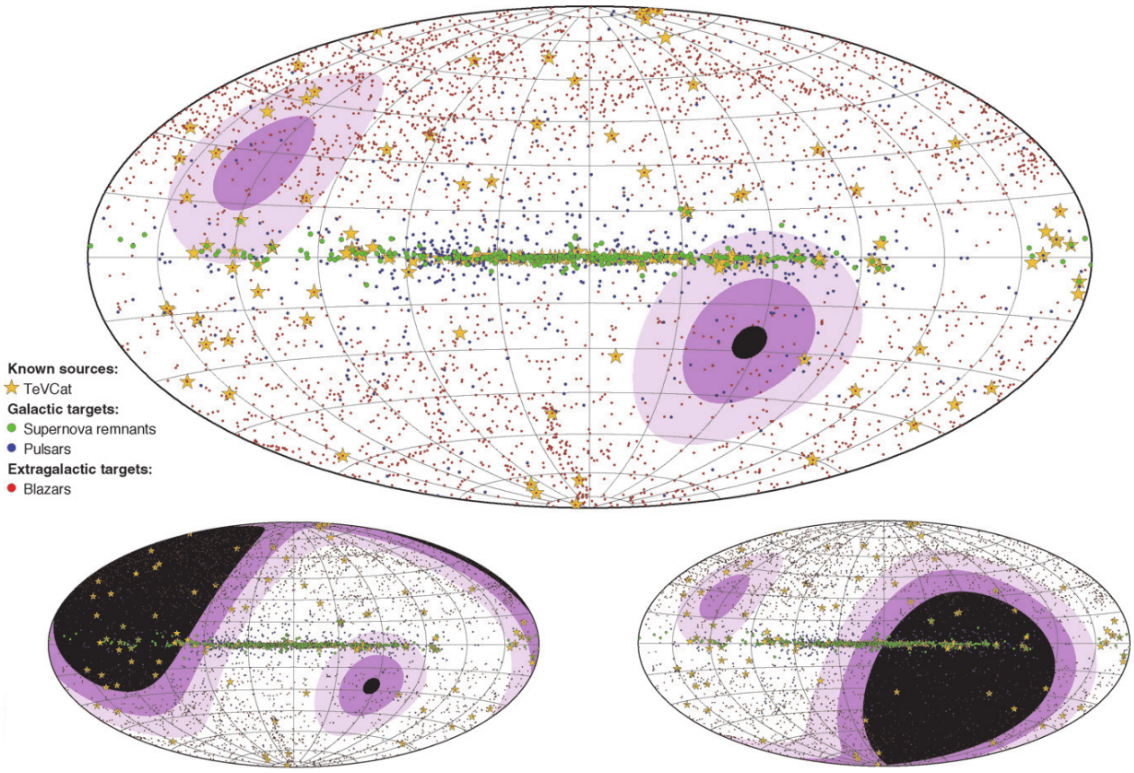


Figure 1.14 – CTA whole array sky coverage compared to the southern (*left*) and the northern (*right*) sites alone. The sky is shown in galactic coordinates, with the galactic plane along the equator. The red dots indicate extragalactic very high-energy gamma-ray sources, while the green and red dots indicate the galactic ones. The color scale indicates the minimum zenith angle under which a target is visible, from 0°–30° (white) to 30°–45°, 45°–60° and > 60° (black). The white areas are the optimal ones. As 60° is the practical limit, the black areas are not covered. Source [Hofmann, 2017], modified.

Figure 1.15 – CTA, La Palma site. Credit: Gabriel Pérez Diaz, IAC

1.3.2 Data Acquisition and Calibration

The cameras of CTA are composed of photomultipliers and readout systems. They convert the Cherenkov photons into photoelectrons via high-speed analogic-to-digital converters (ADCs) and a calibration procedure. In the case of the LST, the waveform signal from the photomultipliers is divided into a high gain channel and a low gain channel, in order to extend the dynamic range. Both channels are temporarily stored in a buffer, and a dedicated mechanism triggers the generation of the output of the camera as a sequence of images (40 for the LSTCam) based on the strength of the received signal.

For each photomultiplier, the calibration converts the ADC counts of both high-gain and low-gain channels into pixel intensity (in photoelectron), also called pixel charge. It takes into account the pedestal[16] and the gain of the channels, as well as the flatfield coefficient that represents the difference in optical and quantum efficiency between pixels. For each pixel, one channel is selected based on a threshold value to produce the final pixel value. Eventually, the gain channel will be selected at the camera level. The data produced by the telescopes consist then in sequence of calibrated images that last some ns.

At the array level, the default working mode of each site is stereoscopy. An array-wise trigger mechanism controls the generation of data. At least two telescopes should trigger for an event in order to store the output of every triggered telescope. Noteworthy, as the LST1 will be the only telescope of the northern site until mid 2022, a temporary mono-trigger mechanism has been set up.

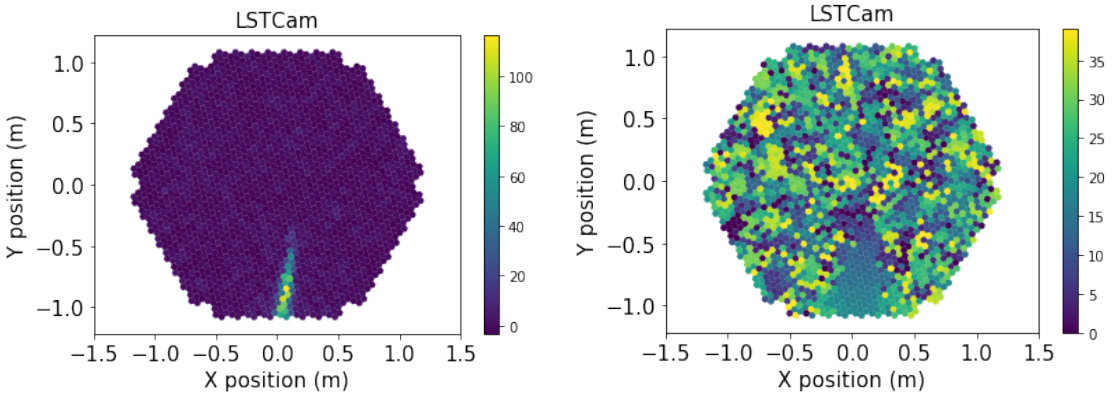
1.3.3 Image Integration and Temporal Information

The data produced by CTA are sequences of images. To reduce the volume of data to process, a common way consists in integrating the sequences as single images. Three integration methods are mainly used in operating IACT arrays. In the *local peak integrator*, for each pixel, a temporal window is centered on the maximum value of the sequence (the peak). The intensities included in this temporal window are summed to produce the integrated value of the pixel. On another hand, the *global peak integrator* takes into account the maximum intensity across all the pixels to position the integration window. A more elegant method, the *neighbor peak integrator*, takes into account the peak of the neighbors of each pixel to determine the position of the integration window. This results in following the shower development and minimizing the readout noise. However, this method needs a very good temporal calibration of the camera. Currently, the data acquired by the LST1 are integrated with the local peak method. As the telescope is still in the commissioning phase, the temporal calibration of the pixels has to be improved. The local peak method ensures a robust integration of the data with respect to the time information. It is important to note that the models developed in this thesis are sensitive to the data distribution, and so to the integration method applied. Consequently, updates of this method may involve model retraining.

Furthermore, the temporal information that is lost in the integrated image is crucial for the analysis, especially in the single-telescope mode. Indeed, showers produced by protons take longer to develop than electromagnetic ones, and can thus be distinguished

[16] ADC count when the sensor does not receive any target photon. It corresponds to the pixel charge distribution from night sky background light and the electronic noise of the sensor.

from the latter. Also, the gradient of the pixel peaks gives an insight on the shower development, and so on the particle direction. To keep the temporal information, the peak position per pixel in the original sequence can be added as an additional channel to the data, as illustrated in Figure 1.16.



(a) Integrated image. The value of the pixels is in number of photoelectrons.

(b) Temporal information. The value of the pixels represents the temporal position of the intensity peak in the sequence produced by the camera.

Figure 1.16 – Integrated image of a gamma event.

1.3.4 Simulated Data

In a general manner, the ground truth is impossible to obtain for real data in gamma-ray astronomy. Fortunately, the knowledge of the physics of the phenomenon and of the telescopes allows simulating high-quality events with a Monte Carlo shower simulator and a telescope simulator [Bernlöhr, 2008].

Given the information about the primary cosmic particle to simulate and the atmosphere, the CORSIKA [Heck et al., 1998] software generates the particle air shower through Monte Carlo calculations. The *CERENKOV* option enables the simulation of the Cherenkov emission for the showers. With the option *IACT*, CORSIKA outputs telescope independent data in a format readable by sim_telarray [Bernlöhr, 2008]. The sim_telarray package generates the telescope response, if it detects the shower, based on its configuration (position, pointing direction, optical and electronic properties).

As the Monte Carlo simulation of the shower is costly, to save computation time, a shower is reused several times to generate as many events. We slide it over the array of telescopes, changing its virtual impact point, to produce different telescope responses with sim_telarray.

Different types of particle are simulated, including proton, electron, diffuse gamma and point-like gamma, for given arrival directions. Modifying the arrival direction changes the quantity of atmosphere the shower goes through, and thus changes its size and shape. Diffuse gamma events correspond to extended sources emitting in various directions while point-like gamma events correspond to sources emitting in a unique direction. Although point-like sources do not exist in real life, very distant objects (as extragalactic ones) appear as points because of telescope optical properties, as for any optical system.

A typical dataset for a particle type is composed of several hundreds of thousand

events[17]. The analysis models are prepared with these simulations.

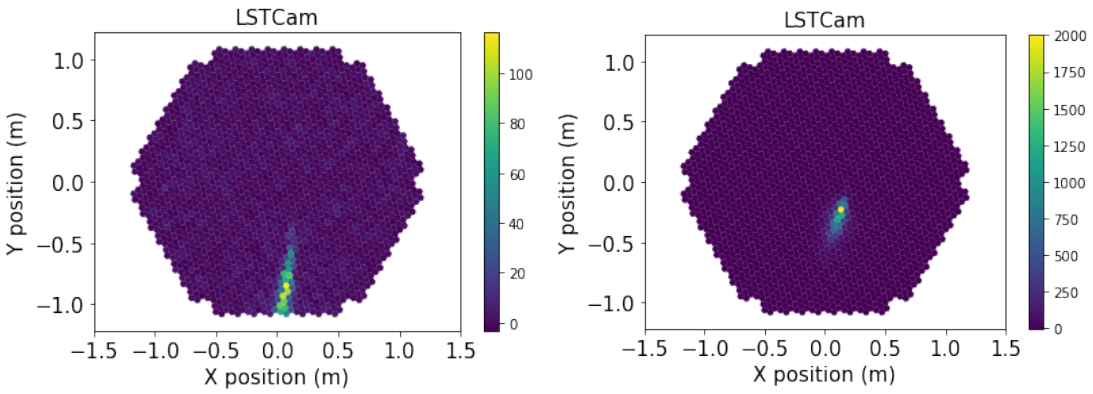
1.4 IACT data analysis

1.4.1 Gamma Event Reconstruction

The purpose of the image analysis is to separate the gamma rays from the cosmic ray background and to estimate the energy and direction of the primary gamma ray. The background rejection is complex because cosmic rays (*i.e.*, protons and electrons) can generate very similar images and the signal-to-noise ratio is typically lower than $1/1000$. The analysis method is then driven by the gamma detection in a high background noise and the regression of its parameters in big data context (3 PB per year for CTA).

It is worth noticing that the direction is reconstructed as the altitude and the azimuth of the gamma ray. Besides, although not required for higher-level analysis, some other parameters can be useful to help with the analysis, such as the virtual impact point of the primary particle on the ground.

The atmospheric detection of gamma rays introduces degeneracies into the parameters to reconstruct. Indeed, there is a strong interdependence between the energy, the arrival direction, the virtual impact point on the ground of the particle and the image produced in the camera (*i.e.*, pixel intensity, shower shape and position). Figure 1.17 shows two gamma rays with the energy and the same arrival direction but a different virtual impact point seen by the LST1. This interdependence make the IACT data analysis multi-task by nature.



(a) Virtual impact point (-28 m, -297 m)

(b) Virtual impact point (-6 m, -79 m)

Figure 1.17 – Comparison of two gamma events with the same energy (2 TeV) and direction (altitude: 70°, azimuth: 180°) and a different impact point. The colorbars represent the intensity of the pixels in photoelectrons.

Furthermore, Figure 1.17 also illustrates that the Cherenkov images are mainly composed of noise. The useful information is contained in a rather small number of pixels compared to the image size.

[17]Events detected by the telescopes. The number of showers simulated can be several hundred times higher, depending on the telescope array considered.

Several approaches have been considered in the past to perform this analysis. First, a series of selection filters (cuts) is generally applied to the data to discard the events that are too difficult to reconstruct. Then, the most common approach relies, after a data preprocessing step, on a geometrical characterization of the ellipsoid produced by a gamma ray combined with multivariate analysis methods. However, this approach does not take into account the multi-task nature of the reconstruction. On another hand, state-of-the-art methods, named Template analysis and detailed in Section 1.4.3.2, are based on a pixel level comparison relying on likelihood between a bank of image templates and the recorded images.

Besides, as already said, in gamma-ray astronomy, it is impossible to obtain the ground truth from real data. Analysis methods then rely on high-quality simulated data (see Section 1.3.4 for details) to produce their model. This also allows for the preparation of the analysis toolchains (ctapipe[18] for the whole CTA and cta-lstchain [19] for the LST1) during the construction of the array.

1.4.2 Data Selection and Preparation

As the first step of the analysis, standard analysis methods described in Section 1.4.3.1 and 1.4.3.2 may require a data selection to discard "bad quality" events or a data preparation to remove the noise in the images. In this section, I present three standard operations in IACT data analysis.

Tail-Cut Cleaning. The purpose of the cleaning operation is to remove the noise of the images to keep only the pixels containing the shower signal, as illustrated in Figure1.18. Additionally, it acts as a selection cut (filter) on the shower size, as images that do not pass the cleaning, *i.e.*, no pixel survives the operation, are discarded.

The tail-cut cleaning consists in a two-threshold procedure. The *picture threshold* defines the minimum intensity of the retained pixels, the picture pixels. The *boundary threshold* is generally lower. It defines the minimum intensity to retain a pixel that is the neighbor of a picture pixel. A third parameter defines the *minimum picture pixel neighbors* for a pixel to survive the cleaning operation.

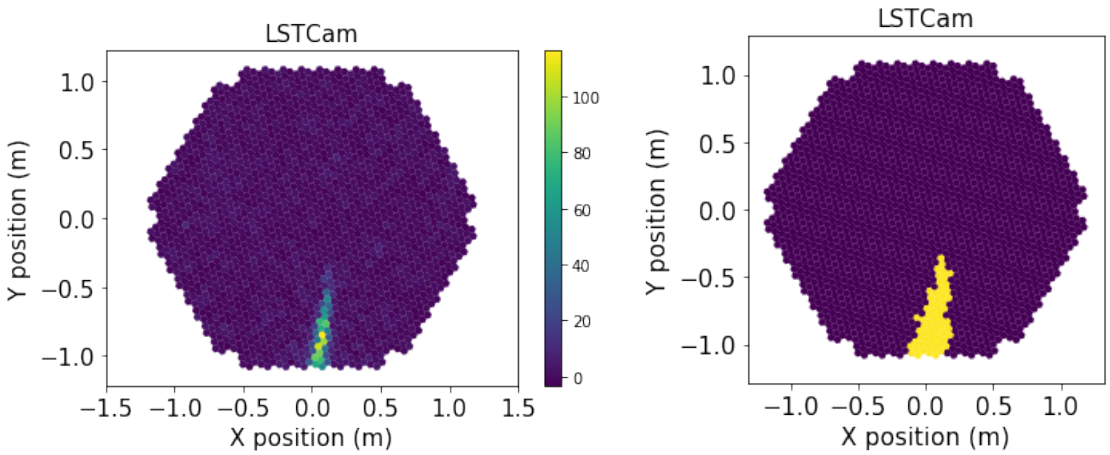
Both thresholds are defined by the expert depending on the data properties, such the level of night sky background or the electronic noise of the telescope hardware.

Intensity Cut. The intensity selection cut consists in discarding images whose total intensity, *i.e.*, the sum of all the pixel intensities, is lower than a defined threshold. In standard analysis frameworks, this selection is generally done after a cleaning operation.

The threshold is defined by the expert depending on the analysis use case.

Leakage Cut. The leakage corresponds to the fraction of the ellipsoid located at the border (one or two pixel width) of the image. It is computed after a cleaning operation and can be measured in number of signal pixels or in intensity. Filtering based on a leakage threshold allows discarding truncated showers.

[18] https://github.com/cta-observatory/ctapipe
[19] https://github.com/cta-observatory/cta-lstchain



(a) Input image.

(b) Pixel mask resulting from the cleaning operation.

Figure 1.18 – Cleaning example.

1.4.3 State-of-the-Art Reconstruction Methods

1.4.3.1 Hillas + RF

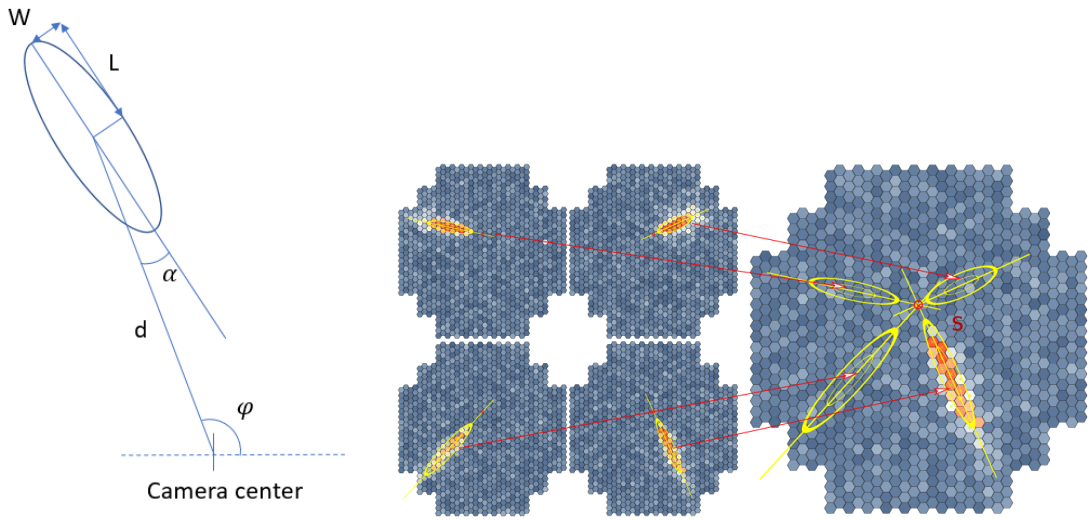
Geometrical Reconstruction. Developed by A. M. Hillas [Hillas, 1985], the geometrical part of the method assumes that, to a good approximation, the image produced by gamma rays has an elliptical shape [De Naurois, 2006]. After a cleaning operation, it characterizes the ellipsoid by its moments up to second order, illustrated in Figure 1.19a, namely:

- the total image intensity,
- the position of the centroid (given by its distance d to the center of the camera and the azimuthal angle φ),
- the semi-minor axis W and the semi-major axis L,
- the angle α between the major axis and the axis defined by the centroid and the center of the camera.

Using several telescopes, the direction of the shower can be determined geometrically. As shown in Figure 1.19b, all the images of the same event are stacked and the intersection of the major-axis of the ellipses shows the position of the source. In the same way, the intersection of these major axis in the site space indicates the impact point of the shower, that is the virtual impact position of the primary particle.

Improving the Sensitivity with Machine Learning. To improve the sensitivity of the analysis, the ellipsoid parameters have been combined with multivariate analysis methods, such as neural networks, boosted decision trees or random forests (RF) [Bock et al., 2004, Albert et al., 2008b, Ohm et al., 2009, Fiasson et al., 2010].

The toolchain designed to analyze the LST1 data is referred in the rest of this thesis to as *Hillas + RF*. It consists in extracting the Hillas parameters and other relevant features from the data (integrated image and peak position) followed by inferring target particle



(a) Characterization of the Hillas parameters.

(b) Geometrical reconstruction of the particle direction with four telescopes. Source [Völk and Bernlöhr, 2009].

Figure 1.19 – Geometrical analysis method.

parameters with random forests, one parameter at a time. It relies on the open-source library cta-lstchain [20] and the scikit-learn library [Pedregosa et al., 2011]. Noteworthy, the direction and the energy of the primary particle are reconstructed first. Then, the inferred energy is used as an additional parameter to perform the gamma/proton separation (classification).

Pros and Cons. The Hillas + RF method is robust and relatively fast, as we will see in Section 4.3.1. However, it lacks sensitivity at low energies ($< 80\,\text{GeV}$), where the ellipsoids produced by faint showers may be harder to characterize.

1.4.3.2 Template-Based Analysis

Template-based analysis of IACT data relies on a per pixel comparison, via a likelihood function, between a bank of templates and the actual images on the cameras. In Model Analysis [de Naurois and Rolland, 2009], the bank is obtained by generating predictions from a semi-analytical model of the shower produced by the primary particle. In ImPACT [Parsons and Hinton, 2014], the images templates are generated with CORSIKA [Heck et al., 1998] and Sim_telarray [Bernlöhr, 2008]. For a set of defined parameters, a large number of simulations are run. The average image is stored as a finely binned histogram representing the response of a 'perfect' camera, then oversampled with the camera pixel size. All the events thereby generated are binned in a number of bins of expected depth of the maximum development of the shower (X_{max}).

Both methods use the same likelihood function proposed in [de Naurois and Rolland, 2009]:

$$P(s|\mu,\sigma_p,\sigma_\gamma) = \sum_n \frac{\mu^n e^{-\mu}}{n!\sqrt[2]{2\pi(\sigma_p^2+n\,\sigma_\gamma^2)}} \exp\left(-\frac{(s-n)^2}{2(\sigma_p^2+n\,\sigma_\gamma^2)}\right) \tag{1.4}$$

[20] https://github.com/cta-observatory/cta-lstchain

where s is the actual image, μ the template image, n a photo-electron, σ_p the width of the pedestal and σ_γ the width of the single photo-electron distribution.

Pros and Cons. Template-based analyses are the state-of-the-art methods for gamma event reconstruction from IACT data. On the contrary to Hillas + RF method, they do not need a cleaning operation to extract the signal. However, for the moment, their computational cost is very high and their inference time is too long [Parsons et al., 2016] to comply with CTA trigger rate. Moreover, each telescope of the array needs a huge database of templates. They may be inoperable for CTA data analysis due to the number of telescopes and the huge amount of data generated every year.

1.4.4 Performance Evaluation

To evaluate the performance of the different analysis methods compared in this thesis, I will use standard classification indicators for the gamma/proton separation task. On the other hand, I will use standard gamma-ray astronomy ones for the energy and direction reconstruction tasks and the overall performance of the pair telescope-analysis method.

It is important to note that the standard procedure in IACT data analysis consists in building the analysis models with gamma diffuse events, so as to reconstruct events coming from any directions within the field of view, and proton events, and to evaluate them on gamma point-like events.

1.4.4.1 Angular Reconstruction

The performance of the direction reconstruction is evaluated with the *angular resolution* of the model. Similarly to the energy resolution, the angular resolution or spatial resolution represents, per energy bin, the interval that contains 68% of the distribution of the angular separation between predictions and true directions. The angular separation is computed as follows:

$$\theta = \arccos(\cos(alt_{reco})\cos(alt_{true})\cos(az_{reco} - az_{true}) + \sin(alt_{reco})\sin(alt_{true})) \quad (1.5)$$

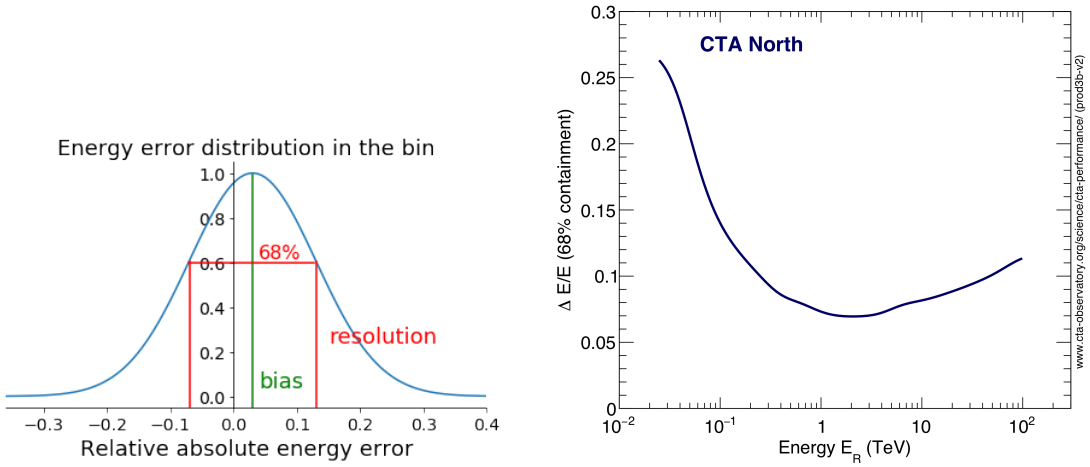
where *alt* stands for altitude, *az* for azimuth and *reco* for reconstructed. The altitude and azimuth biases represent the mean prediction error per bin of respectively the altitude and the azimuth. For all indicators, lower is better.

1.4.4.2 Energy Reconstruction

The performance of the energy reconstruction is evaluated with the *energy resolution* of the model. The energy resolution or spectral resolution represents, per energy bin, the interval from 0 which contains 68% of the distribution of the relative prediction error $\frac{\Delta E}{E}$. Lower is better. The *energy bias* represent the mean relative prediction error per bin. Figure 1.20 shows the resolution computation for a particular bin and, as an example, the expected energy resolution of CTA north (whole site).

1.4.4.3 Gamma/Proton Separation

For the gamma/proton separation task, we consider gamma as the positive class in the description of the following metrics.



(a) Energy resolution for a particular energy bin

(b) CTA north expected performance

Figure 1.20 – Computation of the energy resolution.

Area under the Curve (AUC). The AUC is the area under the receiver operating characteristic (ROC) curve. The ROC curve represents the evolution of the performance of a binary classier as its discrimination threshold varies. It is plotted as the true positive rate against the false positive rate at various thresholds. The AUC then is the probability that the model ranks[21] a random positive example higher than a negative one.

Precision and Recall. The precision is the ratio gamma events to events classified as gamma. It measures the ability of the model to reject the background noise. The recall is the ratio of gamma events classified as gamma to the total amount of gamma events in the test set. It represents the ability of the model to detect gamma events.

F1 Score. The F1 score is a combination of the precision and the recall defined as

$$F_1 = 2\,\frac{Pr \cdot Re}{Pr + Re}. \tag{1.6}$$

1.4.4.4 Overall Performance

Sensitivity. The *sensitivity* curve represents, per energy bin, the gamma-ray flux that an observed point-like source should emit to allow a detection with a significance (defined below) of 5 σ[22] for a 50 hours observation. The sensitivity is an overall evaluation of the performance of the telescope and the analysis together. It takes into account the efficiency of the telescope and the performance of the analysis, *i.e.*, all the other indicators. Lower is better.

Significance. The *significance* describes the probability that a detected signal belongs to a gamma-ray source rather than the background noise. It is computed by comparing

[21] The output of the model varies in range [0, 1].

[22] Standard deviation of a normal distribution. 5 σ means that the signal detected has one chance in 3.5 millions to belong to the background noise.

the signal detected coming from a region containing a source (ON-region) and the signal detected from a region that does not contain any source (OFF-region). The definition of the significance given in [Li and Ma, 1983] is widely used in IACT data analysis:

$$S = \sqrt{2}\left(N_{ON}\ln\left(\frac{(1+\alpha)}{\alpha}\frac{N_{ON}}{N_{ON}+N_{Off}}\right)+N_{Off}\ln\left((1+\alpha)\frac{N_{Off}}{N_{ON}+N_{Off}}\right)\right)^{\frac{1}{2}} \quad (1.7)$$

where N_{ON} is the number of events detected when pointing to the source, N_{Off} the number of events detected from background noise, and α represents the ratio of ON and OFF-region areas and exposure time.

1.5 LST4 Monotrigger Dataset

The dataset used for the experiments presented in this thesis is referenced as the LST4 monotrigger Production (from 2019/04/15), the large-scale Monte Carlo production generated by the LST collaboration for the LST1 commissioning. For the moment, these data are available exclusively for the CTA collaboration. The specificities of this dataset compared to the current prod3b [Cumani et al., 2017] are twofold. First, it only contains the data of the four LSTs of the Northern site of CTA. Next, the array-level trigger mechanism authorizes to keep the events detected by a single telescope. The LST4 monotrigger dataset is then composed of events of four different types (diffuse gammas and point like gammas, protons and electrons) simulated for a particular telescope pointing direction (70° in altitude and 180° in azimuth) that trigger at least one LST.

1.5.1 Data Statistics

The dataset contains $977k$ diffuse-gamma and $663k$ proton events, among them respectively $485k$ and $282k$ detected by the LST1. The energy, image intensity, altitude, azimuth, core x and core y distributions of diffuse gamma and proton events detected by the LST1 are represented in Figure 1.21.

Gamma and proton events have different simulated energy distributions, both following a power law with a spectral index of -2. Indeed, high-energy events generate larger showers and take thus more time to be simulated. Observing a power law allows saving computing resources. Moreover, the spectrum presented in Figure 1.6 shows that it is coherent with the distribution of particles detectable. However, the spectral index does not have to be exactly 2.7 as each source has a different emission spectrum. Besides, the energy range of simulated protons is different from the one of the simulated gammas, respectively $[0.01;100]$ TeV and $[0.005;50]$ TeV. As the shower generation process is different between gammas and protons, the latter produce less intense images for a given energy of the primary particle. Therefore, the telescope detects, in average, events of higher energy for the protons compared to the gammas. This simulation configuration leads to an imbalanced dataset in terms of the number of events per energy and per particle type.

The altitude, azimuth, core x[23] and core y[24] of detected particles are roughly normally

[23] X position of the virtual impact point of the particle in the array coordinate system.
[24] Y position of the virtual impact point of the particle in the array coordinate system.

distributed, respectively around the telescope pointing direction and the array center (the red line in the figure). Noteworthy, the distribution of the proton event parameter is larger. This is expected as the proton set contains more events of higher intensity. Showers distant to the telescope[25] or diverging from the telescope pointing direction are more likely to be detected if their shower have a higher intensity.

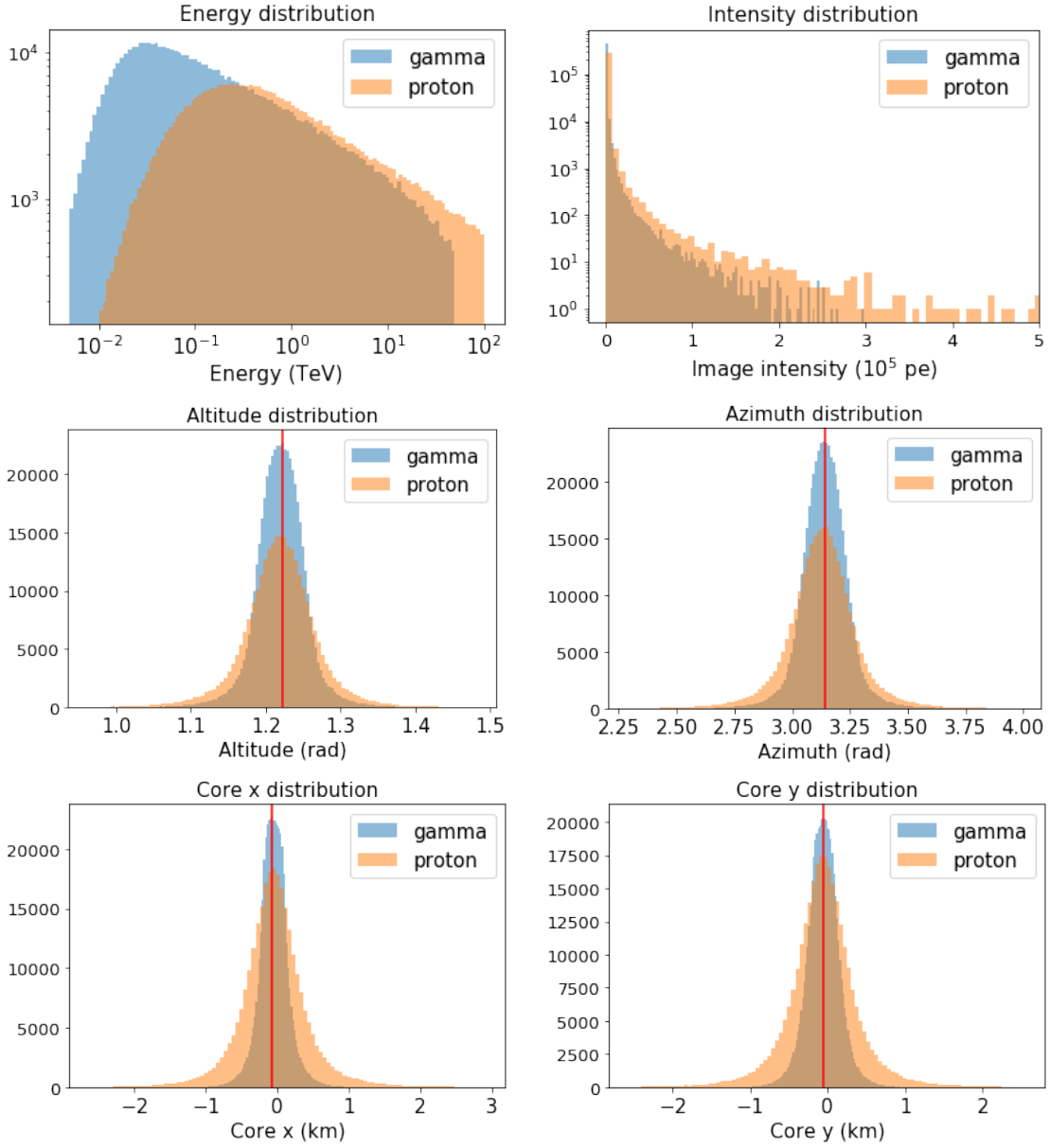


Figure 1.21 – Distributions of parameters for diffuse-gamma and proton events detected by the LST1 in the LST4 monotrigger dataset.

1.5.2 Data Preprocessing for Deep Learning

For the experiments with deep neural networks presented in this thesis, the LST4 monotrigger dataset has been calibrated and integrated with DL1DataHandler [Kim et al., 2019],

[25] The LST1 position in the array coordinates is (-70.93 m, -52.07 m).

using the neighbor peak integrator method. It is separated into a training set and a test set for each event type, with a ratio 80/20. The images have two channels, one for the pixel intensity (or charge, the unit being the number of photoelectrons) and the other containing the temporal information (or peak position) per pixel, as time delays from the beginning of the event captured by the telescope. See Section 1.3.3 for details. The images are not normalized. Indeed, the global intensity of IACT images is actually related to the energy of detected particles [Völk and Bernlöhr, 2009]. The energy labels are transformed from TeV to log10(TeV) and the impact point labels (core x and core y) from meters to kilometers in order to keep them in a range of quite small values to avoid the neural network loss gradient exploding during training.

As already highlighted in Section 1.3.3, deep neural networks are sensitive to the properties of their training data. Therefore, any modification of the data preprocessing may require model retraining. Besides, this bias towards the training data distribution may also be an issue when analyzing real data.

1.6 Deep Multi-Task Learning for CTA Data Analysis ?

CTA proposes new challenges to the analysis of IACT data. Due to the number of telescopes and their better sensitivity compared to existing IACTs, a tremendous amount of data will have to be analyzed in real time and reprocessed every year. State-of-the-art template-based methods (in their current implementation) are then too slow. On the other hand, Hillas + RF methods are robust and relatively fast, but lack sensitivity in the low part of the very high-energy gamma spectrum. However, good performance is crucial for the study of extragalactic sources and transient phenomena below 150 GeV, in the energy range were LSTs have been designed for an optimal sensitivity. Moreover, a major issue of Hillas + RF method is that it does not handle the degeneracies introduced into the parameter reconstruction by the atmospheric detection of gamma rays.

To overcome these issues, we need to investigate other methods. Recently, deep learning has become the leading approach to solve many computer vision tasks. We could expect that it will also be the case for gamma event reconstruction from IACT data. Besides, once trained, deep neural networks are fast at inference (producing a prediction for input data). They also require minimal image treatment before analysis. They could then exploit faint patterns in the images that are discarded by the Hillas + RF method. Deep learning is then worth exploring for IACT data analysis. Preliminary works on CTA data have shown encouraging results, as we will see in Section 2.4, although reconstructing each parameter independently.

However, analyzing CTA data with deep learning algorithms brings interesting challenges. First, CTA is composed of telescopes of different types that produce images of different shapes. Moreover, several cameras of CTA produce hexagonal grid images. However, the deep learning frameworks are designed to process Cartesian grid images. Then, the tremendous volume of data to analyze both in real time and off-line turns the analysis into a big data problem. Finally, we do not have annotated real data to train the models and neural networks need a lot of labeled data in their training phase. The deep learning models will then be trained on high-quality simulations, as the standard methods. The discrepancies between simulated and real data could be an issue, as we will see in Chapter 5.

1.7 Contributions

The contributions of this thesis are as follows:

- I introduce a new deep multi-task learning architecture to achieve gamma event reconstruction from IACT data in the context of single-telescope analysis. I show that this architecture outperforms the standard Hillas + RF method on the simulated data of the LST4 monotrigger dataset. In particular, it achieves very interesting sensitivity below 200 GeV, and could enhance the study of transient phenomena. Besides, although not optimized, the proposed model has an inference rate of the order of the LST1 readout rate. Finally, I also analyze the first exploitable real data produced by the LST1, and show that, despite the discrepancies between the simulated data used to train the model and the real data, the proposed architecture achieves a seven-sigma detection of the Crab Nebula.
- I present an original method to apply the convolution to any kind of pixel organization, in particular to the hexagonal grid of the LST camera. I demonstrate its interest over resampling methods for the LST1 data analysis. This generic method has been released as an open source library.
- I propose a framework to ease the deep learning procedure with CTA data, and to ensure the traceability and reproducibility of the experiments presented. It has also been released as an open source library.

1.8 Thesis Structure

The outline of this thesis is as follows. Chapter 2 presents a review of the deep learning concepts that will be used in the rest of the thesis and previous works applying deep learning to gamma astronomy. Chapter 3 proposes a solution to apply deep learning techniques to hexagonal pixel images without preprocessing and compare it to several resampling methods in various conditions. This work was previously presented in [Jacquemont et al., 2019a, Nieto et al., 2019b], but this chapter contains also new pieces of work. Chapter 4 proposes a deep learning architecture named γ-PhysNet to perform IACT data analysis in the single-telescope context. This work has been submitted as a conference paper to the 25th International Conference on Pattern Recognition. In Chapter 5, this architecture is used to analyze LST1 real data from the February 2020 Crab campaign. Finally, Chapter 6 suggests directions for future research. As a side note, the experiments presented in this thesis have been carried out with the help of the GammaLearn framework detailed in Appendix A.

2

Deep Learning in The Context of IACT Data Analysis

In this chapter, after a short introduction to deep learning, I present key concepts related to neural networks that will be used in the following of this thesis. As IACT data analysis is multi-task by nature, I then realize an overview of multi-task learning for neural networks. I also review attention mechanisms that help neural networks focus on the relevant features in the data, as it could be interesting to exploit the faint differences between certain gamma and proton events. Finally, I present previous works focusing on applying deep learning to IACT data analysis.

2.1 Introduction to Deep Learning

Deep learning is a field of machine learning that includes algorithms composed of several layers of artificial neurons. Standard machine learning algorithms are fed with engineered features to solve a task and are composed of few parameters to be tuned. On the other hand, deep learning algorithms extract the relevant features directly from the data. They require a minimal preprocessing by learning a huge number of parameters. Deep learning then belongs to representation learning.

Although the artificial neuron originates in the 1940s, the journey of deep learning has been eventful until the present days. The perceptron, proposed by Rosenblatt in 1957 [Rosenblatt, 1958], first generated enthusiasm in the artificial intelligence community before being put aside in the 1970s because of its limits, *i.e.*, training difficulties and lack of model understanding. The neural network field became again of interest in the late 1980s when Lecun et al. [LeCun et al., 1989a] used the backpropagation algorithm [Rumelhart et al., 1986] to train a convolutional neural network in order to recognize handwritten zip codes. However, the field really arose with the use of GPUs to train neural networks at scale [Raina et al., 2009], breaking the computing limitations.

Over the past decade, deep learning has emerged as the leading approach in many

computer vision tasks. Since the 2012 Imagenet Large Scale Visual Recognition Challenge breakthrough [Krizhevsky et al., 2012, Hinton et al., 2012], it has proven to be efficient, even beating human performance [He et al., 2016b] on some specific tasks and data. In particular, convolutional neural networks have achieved great success in various computer vision tasks.

In this section, we present the main components of convolutional neural networks and the key concepts used in this thesis.

2.1.1 From Neurons to Convolutional Neural Networks

Artificial neural networks are trainable models inspired by the biology of the animal brains. They are composed of artificial neurons organized in layers, a layer being composed of several neurons, as illustrated in Figure 2.1. The layers between the input and the output ones are named *hidden layers*. What makes a neural network deep is its number of hidden layers (more than one) and/or their width (their number of neurons). The total number of trainable parameters in deep neural networks ranges from hundreds of thousands to several millions.

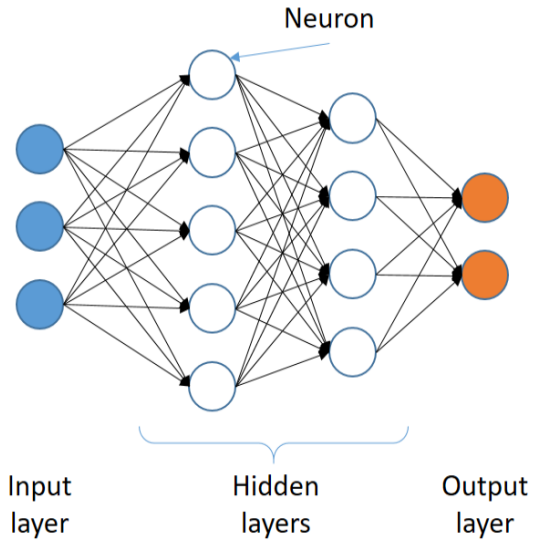


Figure 2.1 – Artificial neural network. The blue circles represent the input units (*e.g.*, the pixels for an image) and the orange ones the output (*e.g.*, the probability of each class for a classification problem). The white circles represent the hidden layers of the network, composed of five and four neurons respectively.

The Artificial Neuron. The artificial neuron itself, as illustrated in Figure 2.2, is a simple trainable unit that performs a linear operation followed by an *activation function* that is not linear. A neuron produces one output, also named *feature* for hidden layers. The linear operation is a weighted sum over the neuron input in addition to a bias. The weights and the bias are trainable and give to the neuron the ability to learn. The non-linear activation allows the neuron to learn functions that are not linear. As a consequence, deep neural networks, composed of several layers of several neurons, are able to model rather complex functions. We can formulate neural networks as mapping functions $f^W : x \mapsto y$ where W are the trainable weights.

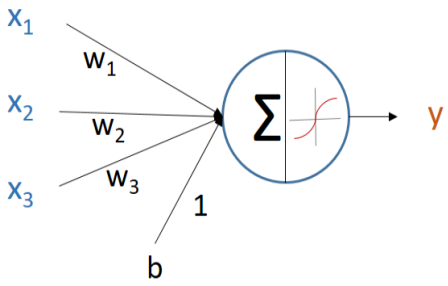


Figure 2.2 – Artificial neuron. The neuron performs a weighted sum over its input(x_1, x_2, x_3 and adds a bias to the result. The weights (w_1, w_2, w_3) and the bias (b) are trainable. The result of this linear operation is activated by a non-linear function (in red) to produce the neuron output.

There exists a large variety of activation functions in the deep learning literature [Bengio et al., 2017, Chapter 6]. In the rest of this thesis, we will mainly use the Rectified Linear Unit (ReLU)

$$ReLU(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

and the logistic function (also referred to as sigmoid, the family it belongs to)

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{2.2}$$

As we will see in Section 2.3, the sigmoid function is particularly interesting to smoothly bound a signal ($\sigma(x) \in [0,1]$). While the sigmoid function has been popular in the early ages of deep learning because of its similarity with the response characteristics of a biological neuron, the ReLU has proven to improve the performance [Nair and Hinton, 2010, Glorot et al., 2011].

Multi-Layer Perceptron. As written previously, the neurons are organized in layers. In *fully connected layers*, also named dense layers, every neuron is connected to all the neurons of the previous layer and outputs a scalar value (a feature), as illustrated in Figure 2.1. The output of a fully connected hidden layer is a feature map. Although a neural network composed exclusively of fully connected layers, called multi-layer perceptron, is a universal function approximator [Cybenko, 1989], it presents several drawbacks. It disregards the spatial information and is thus not invariant to the translation of the input data. The consequence is that we need more data and more hidden neurons so that the model can learn from sufficiently varied situations to solve a particular task. As a result, a severe disadvantage of multi-layer perceptron is that its number of trainable parameters can grow very quickly with the size of the input data and its number of layers L: $N_{total} = N_{input} \times N_{output} \times \prod_{l=1}^{L} N_l$ (biases are omitted for simplicity).

Convolutional Neural Networks. To overcome these issues, the convolution operation has been integrated in neural networks [LeCun et al., 1989b, LeCun et al., 1989a]. The convolutional neuron operates on a local neighborhood of its input, and produces a whole feature map by sliding over the entire input (see Chapter 3 for more details on the convolution). As the trainable weights of the neuron are shared across every location, the network is location invariant and can recognize patterns anywhere in the input data. Moreover,

its total number of trainable parameters is greatly reduced, and thus the model is easier to train. Convolutional models have been prevalent in the success of deep learning for computer vision. They achieve state-of-the-art results in image classification [Touvron et al., 2020], semantic segmentation [Yuan et al., 2019] and object detection [Zhang et al., 2020], that are crucial for real-life applications such as autonomous vehicles.

2.1.2 How Neural Networks Learn

For a dedicated task, the process of learning for a neural network consists in adjusting its weights to minimize its error with respect to this task.

Loss Function. The network error is measured through a cost function L parametrized by the weights (W) of the network, named the *loss*. For classification tasks the standard loss function is the cross-entropy:

$$L = -\ln(p_c) \tag{2.3}$$

where p_c is the prediction of the model for the ground truth class c.

For regression tasks, the most common loss function is the mean squared error or $L2$ loss:

$$L = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2 \tag{2.4}$$

where $\hat{y}$ is the prediction of the model, y the ground truth and N the size of y. The $L2$ loss is sensitive to outliers, on the contrary to the mean absolute error or $L1$ loss:

$$L = \frac{1}{N}\sum_{n=1}^{N}|\hat{y}_n - y_n| \tag{2.5}$$

Loss functions can be combined for regularization purpose, as we will see later in this section, or in the context of multi-task learning, as explained in Section 2.2.2.

Gradient Descent. Neural network weights are optimized through the gradient descent algorithm [Bengio et al., 2017, Chapter 8]. It consists in minimizing the loss by updating the model weights in the opposite direction of the loss gradient with respect to them. The learning speed is defined with a learning rate η:

$$W_{t+1} = W_t - \eta.\nabla_W L(W_t). \tag{2.6}$$

The Vanilla (batch) gradient descent updates the parameters for the entire training set based on the mean error of the network. Although it is guaranteed to converge to the global minimum (or a local minimum in case of non-convex surfaces), it is most often not tractable for common computer vision tasks because of memory limitations. Moreover, gradient descent applied to a high-capacity model is prone to overfitting. The model can simply memorize the training set instead of learning a representation of the data, leading to a gap between the training and the test errors.

In contrast, the mini-batch stochastic gradient descent (SGD) updates the parameters for every mini-batch of n training examples. This update is called an *iteration*. An *epoch* corresponds to all the iterations performed on the whole training set. Training a neural

network consists in realizing several epochs, until its error is sufficiently low. The SGD is tractable on modern GPUs (depending on the value of n) and leads to a more stable convergence than vanilla gradient descent. However, good convergence is not guaranteed and we need to carefully choose the learning rate. Scheduling the learning rate is an option to find a better local minimum. Another approach consists in adding a momentum to the weight update. It helps accelerate the optimization in the relevant direction and avoid saddle points.

Adam. Different algorithms have been developed to improve the SGD. In particular, the adaptive moment estimation (Adam) [Kingma and Ba, 2015] computes an adaptive learning rate, and applies a momentum to the past gradients. It estimates the first and second moments of the gradients, respectively the mean m_t and the variance v_t:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{2.7}$$

with g_t the gradient with respect to W at time step t. m_t and v_t are initialized as vectors of zeros, leading to estimates that are biased towards zero in the first steps. To address this issue, the moments are then bias corrected:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-(\beta_1)^t} \\ \hat{v}_t &= \frac{v_t}{1-(\beta_2)^t} \end{aligned} \tag{2.8}$$

and the parameters are updated with:

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \tag{2.9}$$

where the purpose of ε is to avoid a division by zero.

Although Adam is very popular, under certain conditions it generalizes worse than SGD (with momentum) and can lead to suboptimal convergence. To solve this issue, variants of Adam have been proposed [Reddi et al., 2018, Luo et al., 2019]. We can also alternate between Adam and SGD during training [Keskar and Socher, 2017].

Network Regularization. During the training phase, we can add a regularization term to the loss to enforce weight sparsity and to prevent the model from overfitting. The regularization term is often the $L2$ norm of the weights, often called $L2$ *penalty*, scaled by a *weight decay* factor (Tikhonov regularization).

To reduce overfitting, we can also stochastically switch off a certain number of neurons during training. This procedure, named *dropout* [Hinton et al., 2012], prevents neuron co-adaptation. It encourages the network to learn a sparse representation of the data [Srivastava et al., 2014].

Backpropagation. To compute the gradients of the loss with respect to every weight, we use the backpropagation algorithm [Rumelhart et al., 1986]. Considering the neural network as a computational graph, the backpropagation applies the reverse-mode differentiation to every node (weight) in a single pass. The backpropagation algorithm is essential for the tractability of neural network training as the latter often have millions of parameters. A comprehensible explanation of the algorithm is given in [Olah, 2015].

Deep Learning Frameworks. Modern deep learning frameworks, such as PyTorch [Paszke et al., 2017] or TensorFlow [Abadi et al., 2016], take care of backpropagation through automatic differentiation. They offer high-level interfaces to optimized implementations for all the building blocks of deep learning.

2.1.3 Comments

Deep learning is often seen as a black box. The field lacks theoretical fundamentals, and most of method validations come empirically. However, efforts are made by the community towards explainable and interpretable models (see Section 4.4 for details).

While deep neural networks are powerful algorithms to learn data representation, they are highly sensitive to data biases. The biases can be inherent to the data [Kortylewski et al., 2019, Papakyriakopoulos et al., 2020] or introduced by the human interacting with the algorithms [Wolf et al., 2017]. In the context of CTA data analysis, the training set contains slightly more gamma events than proton events, while in reality the signal-to-noise ratio is about 10^{-3}. However, this bias is desirable are we are mainly interested in reconstructing gamma events (protons are considered as background noise).

Deep neural networks are also sensitive to adversarial attacks [Carlini and Wagner, 2017, Eykholt et al., 2018] that consists in modifying the data, *i.e.*, designing adversarial examples, to cause the model make mistakes. Even a slight modification of the data can fool the model [Goodfellow et al., 2015]. Efforts are made to solve this issue and improve the robustness of the models. Adversarial training, for instance, consists in augmenting each mini batch of data with adversarial examples during the training phase [Shafahi et al., 2019]. However, defending neural networks against adversarial attacks is still an active field of research. In the context of CTA data analysis, it exists differences between simulations used for training and real data. For instance, the properties of the simulated telescope, such as its optical efficiency of the electronic noise of its camera, are approximations of the real one. Or physically, the models of shower development are also not perfect. Solving these discrepancies could be similar to defending against adversarial examples.

Neural networks have an important computational cost at training time. Millions of parameters are learned through several epochs (up to hundreds) with huge datasets. Depending on the model and the amount of data, the training time ranges from some hours to several days with modern GPUs. However, once trained, they are very fast at inference time.

2.2 Multi-task Learning

As we have seen in Chapter 1, the analysis of IACT data is multi-task by nature. Multi-task learning is a paradigm in which all the tasks a system has to address are learned simultaneously, through a common model.

Multi-task Learning Principle. Consider a system based on machine learning that has to address several related tasks, as vision systems in real-world applications (*e.g.*, autonomous driving system). Multi-task learning [Caruana, 1997] consists in learning all the tasks with a single model instead of learning independent models, one per task. From

a biological point of view, we can see multi-task learning as a way to mimic human learning. As a human being, to learn a new task we often use the knowledge acquired formerly by learning related tasks. From a machine learning point of view, it can be seen as a form of inductive transfer by introducing inductive bias [Ruder, 2017].

In the context of deep learning, recent methods based on convolutional neural networks have shown remarkable results on pose estimation [Pavllo et al., 2019] or instance segmentation [He et al., 2017].

Benefits. Multi-task learning paradigm aims to improve the generalization [Caruana, 1997] of learned models. Former approaches [Thrun, 1996] have shown that transferring knowledge across related tasks improves the generalization with fewer data. As different tasks have different noise patterns, multi-task learning acts as implicit data augmentation and helps to learn a more general representation of the data. It also reduces the risk of overfitting the shared parameters in an order of the number of tasks [Baxter, 1997]. Multi-task learning helps the model focus its attention on features that are relevant for all tasks. It biases the model to learn a representation suitable for all the tasks, improving its generalization capabilities. However, if the tasks learned are competing, multi-task learning can lead to learn the best compromise instead of the best model.

Finally, depending on the model architecture designed, multi-task learning is also more efficient in terms of memory consumption and inference speed, as we will see in Section 2.2.1.

In situations where we have to address only one task, we can still benefit from multi-task learning by adding a related auxiliary task.

Challenges. Successfully learning a shared representation across several tasks poses two key challenges. First, the design of the multi-task architecture is crucial to learn both shared and task-specific relevant features. Then, the way of combining the loss functions of the different tasks is also essential. We want to learn all tasks with an appropriate importance, and avoid easy tasks to dominate.

2.2.1 Architecture Definition

In the context of multi-task learning, the tasks to address are learned simultaneously, using a partially shared representation. While in early works the shared layers are designed by the human expert, recent approaches learn what to share. The former approach is referred to as hard parameter sharing and the latter partially or soft parameter sharing [Ruder, 2017].

2.2.1.1 Hard Parameter Sharing

It is the most frequently used architecture. A whole part of the network is shared between all tasks [Ruder, 2017]. As illustrated in Figure 2.3, the shared part is generally the encoder [Kendall et al., 2018, Luvizon et al., 2018, Ren and Jae Lee, 2018, Xu et al., 2018] or its first layers [Iizuka et al., 2016], to produce a shared representation of the data. The task-specific layers are generally fully connected layers, but we can also use convolution layers for particular vision tasks, such as semantic segmentation, as shown in Figure 2.3. These task-specific layers produce the prediction of each task. The human expert chooses the layers that are shared and the ones that are specific to the different tasks. It can be

challenging to decide the number of shared layers and the scale of the shared representation. To overcome this issue and save memory, UberNet [Kokkinos, 2017] proposes a pyramidal approach to solve simultaneously seven computer vision tasks.

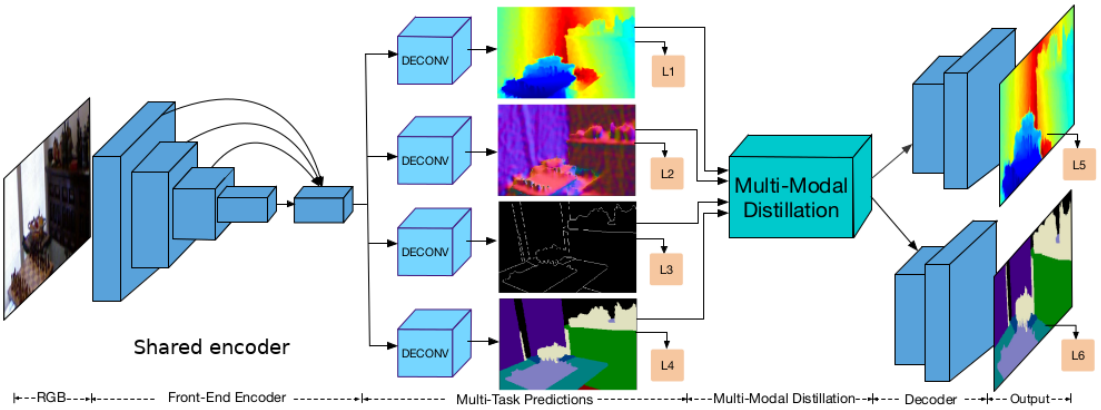


Figure 2.3 – PAD-Net architecture for depth estimation and scene parsing with surface normal estimation and contour detection as auxiliary tasks. PAD-Net is a hard parameter sharing multi-task model. The front-end encoder is shared across all the tasks. Source [Xu et al., 2018], modified.

Pros and Cons. Hard parameter sharing reduces the risk of overfitting [Baxter, 1997]. As a whole part of the network is shared across all tasks, it also reduces the computational cost and memory consumption at both training and inference time. However, the performance of hard parameter sharing networks is dependent on closely related tasks, and defining which layers to share is not obvious. Moreover, it requires to balance the different tasks at the loss level (see Section 2.2.2 for details about balancing methods) to avoid the easiest ones dominating the learning.

2.2.1.2 Partial or Soft Parameter Sharing

In soft parameter sharing architectures, each task is learned with its own network. To enforce a shared representation across the tasks, their networks exchange information or specific layers are constrained to have similar parameters. Yang et al. regularize model parameters of some layers by the tensor trace norm [Yang and Hospedales, 2017]. All the tasks are learned with a different model, but with the same network architecture. For each layer considered as shared, they build a tensor by concatenating the parameters of the different networks. Then, they train all the networks with a common gradient descent regularized by the tensors trace norm. Deep Collaboration [Trottier et al., 2017] proposes to combine the feature maps of the shared layers through collaborative blocks. A central aggregation module transforms the feature maps of the shared layer from the different networks into a global feature map through learnable operations. This global feature map is then injected into every task-specific network. As illustrated in Figure 2.4, Partially Shared Multi-task Convolutional Neural Network [Cao et al., 2018] is composed of task-specific networks and a shared network. To enforce the latter to learn informative representations shared by all the tasks, their feature maps are combined with the ones of the shared network, and vice versa.

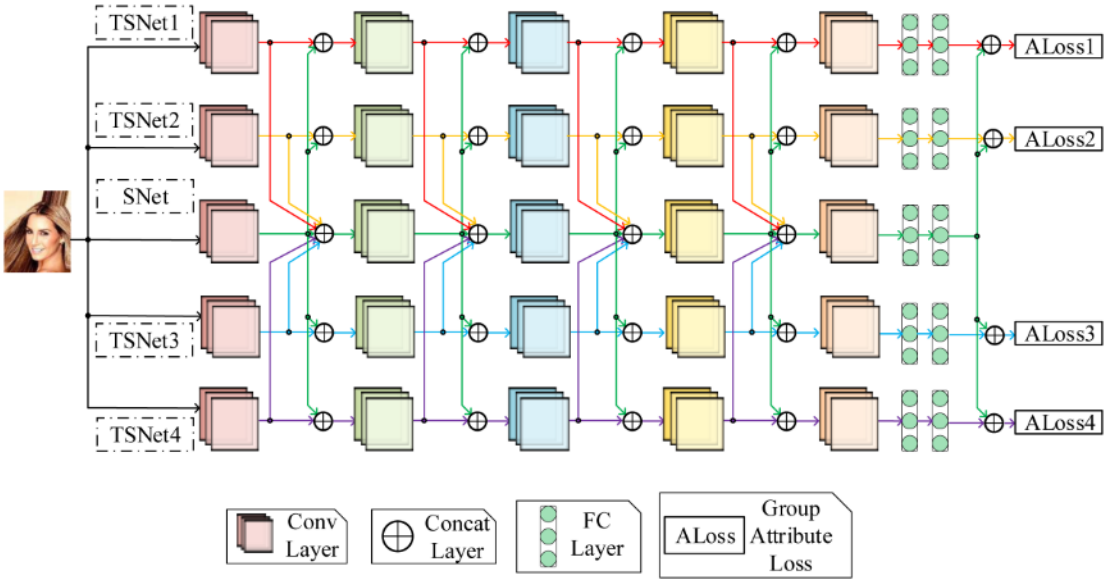


Figure 2.4 – Partially Shared Multi-task Convolutional Neural Network for face attribute learning. It is composed of task-specific networks and a shared network. After every convolutional layer, the task feature maps are concatenated with the ones of the shared network and vice versa. Source [Cao et al., 2018].

Pros and Cons. Soft parameter sharing architectures is less sensitive to the balance of the different tasks. The model learns what to share, and having a network per task prevents noxious tasks from contaminating the others. However, for the same reason, soft parameter sharing does not allow saving memory or training/inference time.

2.2.2 Balancing the Tasks

In the context of deep multi-task learning, balancing the tasks is critical, in particular for hard parameter sharing architectures. In the multi-task loss function, we need to weight the relative contribution of each task, to avoid easiest ones to take over the learning of the whole model. For most of the multi-task learning related papers [Kokkinos, 2017, Han et al., 2017, Luvizon et al., 2018, Ren and Jae Lee, 2018], the task weights are defined, when specified, by hand. The global loss function is defined as:

$$L_{total} = \sum_{t} \lambda_t L_t \tag{2.10}$$

with L_t and λ_t respectively the loss and the weight of task t. This handcrafted weighting needs an extensive optimization process to find optimal ones. However, adaptive methods have been proposed in order to automatically balance task importance. We present in the following some of these methods.

2.2.2.1 Uncertainty Estimation as a Proxy

To train a multi-task architecture performing semantic segmentation, instance segmentation and depth estimation from RGB images, Kendall et al. [Kendall et al., 2018] consider the homoscedastic uncertainty of each task. As illustrated in Figure 2.5, they use it as a proxy for task balancing.

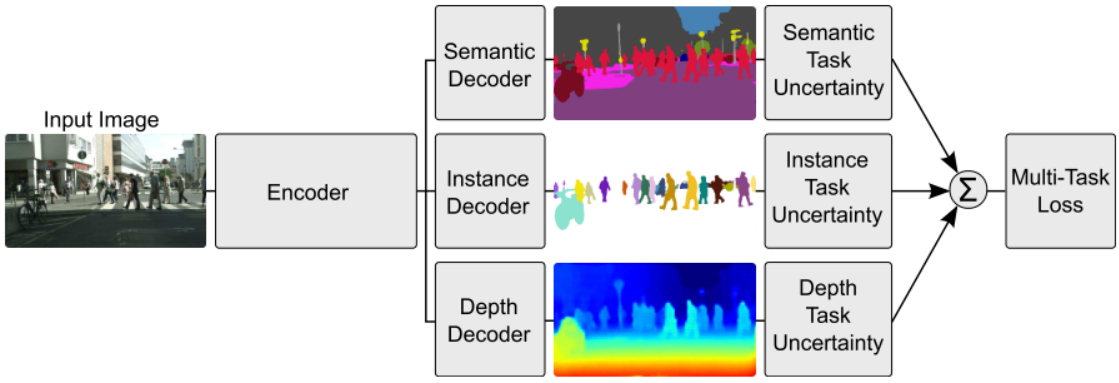


Figure 2.5 – Deep multi-task architecture for semantic segmentation, instance segmentation and depth estimation from RGB images. The tasks are balanced using their uncertainty as a proxy. Source [Kendall et al., 2018].

Principle. The method is derived from uncertainty estimation in Bayesian modelling [Kendall and Gal, 2017]. On the contrary to the epistemic uncertainty that is related to the model, the aleatoric uncertainty is either data dependent (heteroscedastic) or task dependent (homoscedastic). The latter varies between the different tasks we want to address, but stays constant for all the input data. The homoscedastic uncertainty captures the relative confidence between the tasks, and can be used as a basis to balance them.

Uncertainty Estimation for a Regression Task with MSE Loss. Consider $f^W(x)$ as the output of a neural network with weights W for input data x. For regression tasks, we can define the likelihood of the model as Gaussian:

$$p(y|f^W(x)) = \mathcal{N}(f^W(x), \sigma^2) \tag{2.11}$$

with y the label of x and σ the homoscedastic uncertainty. As detailed in [Kendall et al., 2018], maximising the log likelihood leads to the following minimization objective:

$$\boxed{L(W,s) = 0.5e^{-s}||y - f^W(x)||^2 + s} \tag{2.12}$$

with $s = log\,\sigma^2$ the log variance, e^{-s} the precision and $||y - f^W(x)||^2$ the MSE loss function.

Uncertainty Estimation for a Classification Task with Cross-Entropy Loss. For classification tasks, we can scale the output of the network and define the likelihood as:

$$p(y|f^W(x)) = Softmax\left(\frac{1}{\sigma^2} f^W(x)\right) \tag{2.13}$$

with y the class of example x.

Maximising the log likelihood leads to the following minimization objective:

$$\boxed{L(W,s) = 0.5e^{-s}\left(-\log \text{Softmax}(y = c, f^W(x))\right) + s} \tag{2.14}$$

with c the class of input x and $-\log \text{Softmax}(y = c, f^W(x))$ the cross-entropy loss.

Uncertainty Estimation for a Regression Task with L1 Loss. As mentioned in [Kendall et al., 2018], for regression tasks we can also define the likelihood of our model as Laplacian with mean given by the model output:

$$\begin{aligned} p(y|f^W(x)) &= \mathscr{L}(f^W(x), b) \\ &= \frac{1}{2b}\exp\left(\frac{-|y - f^W(x)|}{b}\right) \end{aligned} \tag{2.15}$$

with b the diversity of the distribution.

The log likelihood of the output, that represents the objective to *maximize* with respect to W and b, becomes:

$$\log p(y|f^W(x)) \propto -\frac{1}{b}|y - f^W(x)| - \log 2b. \tag{2.16}$$

Analogously to the Gaussian likelihood, $2b$ can be considered as the output's observation noise.

Turning it to a minimization objective, the loss function is then:

$$\begin{aligned} L(W, b) &= -\log p(y|f^W(x)) \\ &\propto \frac{1}{b}|y - f^W(x)| + \log 2b \end{aligned} \tag{2.17}$$

enabling the use of $L1$ loss for regression tasks in the context of task balancing with uncertainty estimation. In practice, as for the Gaussian likelihood, we train the network to predict the log diversity $v := \log 2b$ for numerical stability, leading to the following minimization objective:

$$\boxed{L(W, b) = 2e^{-v}|y - f^W(x)| + v} \tag{2.18}$$

with e^{-v} the precision and $|y - f^W(x)|$ the L1 loss.

Multi-Task Global Loss. In a multi-task context, we can combine Equations 2.12, 2.14 and 2.18 to compute the global loss of the model:

$$\begin{aligned} L_{total} =& 0.5e^{-s_1} MSE(y_1, f^{W_1}(x)) + s_1 + \\ & 0.5e^{-s_2} Xentropy(y_2 = c, f^{W_2}(x)) + s_2 + \\ & 2e^{-v_3} L1(y_3, f^{W_3}(x)| + v_3 + \dots \end{aligned} \tag{2.19}$$

The log variance s or log diversity v of each task is a trainable parameter, but not an additional output of the model. It does not depend on the model weights. It is learned independently, with a different optimizer than the one of the network.

Pros and Cons. Multi-task loss balancing through the homoscedastic uncertainty estimation is an adaptive and fully automatized method. It is efficient as we only need to learn one additional parameter per task.

2.2.2.2 GradNorm

Chen et al. [Chen et al., 2018] propose a method, named GradNorm, to balance the loss functions of a multi-task network by penalizing predominant tasks and encouraging weaker ones. In other words, λ_t from Equation 2.10 should decrease relatively to the other tasks if task t trains quickly, and increase if task t trains slowly.

Principle. GradNorm relies on the gradient of the different tasks with respect to the last common layer to estimate their training rate. The weight of each task is a trainable parameter, but not an additional output of the network, similarly to the uncertainty estimation method. This weight is learned to adjust the gradient norm so all the tasks train at similar rates.

Gradient Loss Computation. We first compute the norm of the gradient of each task weighted loss with respect to the last common layer. For task t at iteration i:

$$G_W^{(t)}(i) = \|\nabla_W(\lambda_t(i)\,L_t(i))\|_2 \tag{2.20}$$

with W the weights of the last shared layer, L_t the loss of task t and λ_t its trainable coefficient.

The average gradient norm across all tasks at iteration i is then:

$$\overline{G}_W(i) = \frac{1}{T}\sum_t G_W^{(t)}(i) \tag{2.21}$$

with T the number of tasks.

We then compute the inverse training rate $\widetilde{L}_t(i)$ and the relative inverse training rate $r_t(i)$:

$$\widetilde{L}_t(i) = \frac{L_t(i)}{L_t(0)}, \tag{2.22}$$

$$r_t(i) = \frac{\widetilde{L}_t(i)}{\frac{1}{T}\sum_t \widetilde{L}_t(i)}. \tag{2.23}$$

The first GradNorm's objective is to place gradient norms $G_W^{(t)}(i)$ on the same scale, $\overline{G}_W(i)$. The second objective is to encourage weak tasks and to penalize strong ones. The relative inverse training rate $r_t(i)$ is used to scale the gradient norm in that way. Balancing the tasks amounts then to minimizing

$$L_{grad}(i;\lambda_t(i)) = \sum_t \left| G_W^{(t)}(i) - \overline{G}_W(i)\,[r_t(i)]^\alpha \right| \tag{2.24}$$

with the gradient descent algorithm, considering $\overline{G}_W(i)\cdot[r_t(i)]^\alpha$ as a constant. α is a hyperparameter defining the asymmetry of the GradNorm method. If $\alpha = 0$, GradNorm enforces all the tasks to have equal gradient norm at the last common layer. Higher values of α enforce training rate balancing.

The final step is the normalization of the task weights after the gradient descent update so that $\sum_t \lambda_t(i) = T$.

Pros and Cons. As GradNorm relies on the gradient of the different tasks with respect to the last shared layer of the network, it captures their training rate well. It encourages the tasks training slowly and penalizes the ones training quickly. However, GradNorm is less efficient than the uncertainty estimation method as it requires an additional partial backward pass. Moreover, it has a hyperparameter, the asymmetry α, that could be difficult to tune.

2.2.2.3 Multi-task as Multi-Objective Optimization

Balancing the different tasks can be challenging, in particular if they compete. In this situation, using a proxy objective, as in uncertainty estimation or GradNorm, do not lead to an optimal solution for all the tasks. Sener and Koltum [Sener and Koltun, 2018] propose to consider multi-task learning as a multi-objective optimization to achieve Pareto optimality for each task.

Principle. Formulating multi-task learning as a multi-objective optimization, we can find solutions that are not dominated by any others, for all the tasks. The authors propose a Frank-Wolfe-based optimizer to scale the multiple-gradient descent algorithm to high-dimensional gradients of deep neural networks. They also provide an upper bound to the optimization objective that makes the computational cost of the method negligible, while producing a Pareto optimal solution.

Multi-Objective Optimization. Consider a multi-task model $f(W^{sh}, W^t)$ with W^{sh} the weights shared across the tasks, W^t the task-specific weights and $t = 1, \dots, T$ the tasks. Instead of combining the task loss through a weighted sum, the multi-objective optimization formulation defines a vector loss $\mathbf{L}$:

$$\mathbf{L}(W^{sh}, W^1, \dots, W^T) = (L^1(W^{sh}, W^1), \dots, L^T(W^{sh}, W^T))^\mathsf{T} \tag{2.25}$$

with L^t the loss of task t.

To find the Pareto set of optimal solutions, the authors derive the following optimization problem from the Karush-Kuhn-Tucker conditions:

$$\min_{\alpha^1, \dots, \alpha^T} \left\{ \left\| \sum_{t=1}^{T} \alpha^t \nabla_{W^{sh}} L^t(W^{sh}, W^t) \right\|_2^2, \text{with the constraint } \sum_{t=1}^{T} \alpha^t = 1, \alpha^t \geq 0 \forall t \right\} \tag{2.26}$$

that they solve with the Frank-Wolfe algorithm. However, the whole procedure needs one forward pass and T backward passes, the latter being computation costly.

For deep multi-task architectures composed of a shared encoder and task-specific decision networks (hard parameter sharing architectures), they propose to optimize an upper bound of Equation 2.26. They demonstrate that optimizing this upper bound also produces a Pareto optimal solution. The optimization problem becomes:

$$\min_{\alpha^1, \dots, \alpha^T} \left\{ \left\| \sum_{t=1}^{T} \alpha^t \nabla_{z} L^t(W^{sh}, W^t) \right\|_2^2, \text{with the constraint } \sum_{t=1}^{T} \alpha^t = 1, \alpha^t \geq 0 \forall t \right\} \tag{2.27}$$

with z the feature maps produced by the shared encoder. Solving this optimization problem requires one additional backward pass per task.

The overall procedure is described in Algorithm 1. The first step consists in applying the gradient descent to the task-specific weights. Then, we use the Frank-Wolfe solver to find the optimal task weights α^t. Finally, we apply the gradient descent to the shared weights, the task losses being weighted by α^t.

Algorithm 1: Multi-objective optimization procedure

for $t = 1$ **to** T **do**
  $W^t = W^t - \eta \nabla_{W^t} L^t(W^{sh}, W^t)$
end
$\alpha^1, \ldots, \alpha^t = \text{FRANK\_WOLFE\_SOLVER}(W)$
$W^{sh} = W^{sh} - \eta \sum_{t=1}^{T} \alpha^t \nabla_{W^{sh}} L^t(W^{sh}, W^t)$

Pros and Cons. The multi-objective optimization finds the optimal solutions, even if the tasks conflict. Moreover, it does not require any additional trainable parameter nor hyperparameter. However, it implies an additional backward pass per task, that is computationally expensive.

2.2.2.4 Dynamic Task Prioritization

In the context of multi-task learning, imbalances in task difficulty can lead to waste resources on easiest ones after mastering them. Guo et al. [Guo et al., 2018] propose to balance the training losses relying on the task difficulty.

Principle. We can represent the difficulty of the tasks with progress signals, or key performance metrics, that are realistic and intuitive metrics. The authors derive the computation of a task difficulty from the Focal Loss [Lin et al., 2017] applied to its key performance metric. Then, they scale the loss of each task with its difficulty.

Task Difficulty Computation and Loss Balancing. Consider $k_t \in [0,1]$ the key performance indicator of the task t. We first compute its exponential moving average:

$$\bar{k}_t(i) = \alpha k_t(i) + (1-\alpha)\bar{k}_t(i-1) \tag{2.28}$$

with $\alpha \in [0,1]$ the discount factor and i the iteration.

Now consider the task-level focusing parameter γ_t for the task t. It represents the rate at which the task is down weighted. The difficulty of the task is defined by:

$$\begin{aligned} D_t(i) &= \text{FL}(\bar{k}_t(i); \gamma_t) \\ &= -(1-\bar{k}_t(i))^{\gamma_t} \log\left(\bar{k}_t(i)\right) \end{aligned} \tag{2.29}$$

with FL the Focal Loss.

Finally, we scale the loss of each task with its difficulty:

$$L_{total} = \sum_t D_t L_t \tag{2.30}$$

Noteworthy, D_t need not to be differentiable. In this case it is considered as a constant (at the iteration time) and dynamic task prioritization amounts to loss weighting as for uncertainty estimation method and GradNorm.

Pros and Cons. Dynamic task prioritization is an efficient method to balance multi-task networks, as it requires few additional computations. It allocates more resources to difficult tasks to optimize the training of the model. However, it requires additional hyperparameters: α and γ. α controls the memory of the key performance indicator moving average. The authors do not specify if it is task-specific or common for all the tasks. They neither specify the value used for their experiments. γ is task-specific and is the rate at which the task is down weighted. This method requires then to tune at least $T+1$ additional hyperparameters. Moreover, we need to define relevant key performance indicators for each task.

2.2.3 Summary

Real-life applications often require to address several tasks simultaneously. Deep multi-task learning aims to solve them with a single network, improving generalization by enforcing a shared representation that suits all the tasks. However, defining the architecture of the model is crucial. In the context of CTA data analysis, computation time is a key point. Hard parameter sharing architectures allow reducing the memory consumption and the training/inference time by sharing an important part of the network across all the tasks. This kind of architecture needs to particularly pay attention to the strategy to balance the tasks. Adaptive methods allow to do it automatically, avoiding the easiest task taking over the training. In particular, the uncertainty estimation method adds the smallest overhead to the computational cost of the model.

2.3 Attention

As we have seen in Chapter 1, gamma and proton events can produce very similar images in the IACT camera. The hints to separate them lie in faint details in the shower contour and its intensity pattern. Attention is a mechanism that helps deep learning model focus on relevant features based on a defined context through trainable weights. From a biological point of view, we can draw an analogy with the human visual system that focuses on relevant parts of its visual field to assist perception [Saenz et al., 2002,Boynton, 2005]. Attention could help the model focus on these details and thus better distinguish between particles.

In deep learning, attention has been introduced by Bahdanau et al. [Bahdanau et al., 2015] in the natural language processing field. It is the main component of Transformer networks [Vaswani et al., 2017, Devlin et al., 2019] that achieve state-of-the-art performance on neural machine translation and image captioning. Over the recent years, attention has become an essential component neural networks for a large number of applications, including in the computer vision field. In the latter, we can distinguish three kinds of attention:

- pixel-wise attention, also named spatial attention,
- channel-wise attention,
- combination of both.

In the following of this section, we detail a relevant method for each attention type.

2.3.1 Spatial Attention: Self-Attention for Computer Vision

Self-Attention Principle. Attention is a combination of a query and a set of key/value pairs [Vaswani et al., 2017]. In sequence-to-sequence models, *e.g.*, for language translation, the value relates to the encoder hidden state for the input word that is being translated, the keys relate to the encoder hidden states for the whole input sequence, and the query relates to decoder hidden state. The output of the attention module is a weighted sum of the values, the weights being the result of a compatibility function of the query with the corresponding key. In self-attention modules, the queries, keys and values come from the output of the previous layer in the network.

From Natural Language Processing to Computer Vision Parmar et al. [Parmar et al., 2018] generalize the Transformer architecture used for neural machine translation to image generation. In particular they use restricted self-attention to focus on local neighborhoods. Wang et al. propose global self-attention as a non-local operation for video classification, image segmentation, object detection and pose estimation [Wang et al., 2018]. While for computer vision tasks attention modules are generally used in combination with convolution blocks, Parmar et al. [Parmar et al., 2019] propose stand-alone local self-attention models for image classification and object detection. On the other hand, Zhang et al. [Zhang et al., 2019] adapt the global self-attention for generative adversarial networks. In particular they reduce the number of channels of the query, key and value tensors by a factor r, denoted reduction ratio in the following of this thesis. This allows decreasing the computational cost of global self-attention that is expensive. In addition, they introduce a trainable parameter to scale the output of the attention module before summing back with the input. In the following of this thesis, self-attention refers to this latter method.

Self-Attention Computation. Consider a self-attention module inside a convolutional neural network. x represents the feature maps produced by the previous hidden layer. For convenience in the computation description, 2D feature maps are vectorized, leading to $x \in \mathrm{Re}^{C\times N}$, with C the number of channels and N the number of pixels per channel. We first transform x into three feature spaces f, g and h, corresponding to the queries, keys and values, through trainable operations:

$$f(x) = W_f x, \tag{2.31}$$

$$g(x) = W_g x, \tag{2.32}$$

$$h(x) = W_h x, \tag{2.33}$$

with $W_f \in \mathrm{Re}^{C'\times C}$, $W_g \in \mathrm{Re}^{C'\times C}$, $W_h \in \mathrm{Re}^{C'\times C}$ and $C' = \frac{C}{r}$. r is the reduction ratio, a hyperparameter controlling the bottleneck of the attention module. For computation efficiency, Zhang et al. choose $r = 8$.

To produce the attention map, we perform the matrix multiplication of the transpose of f with g:

$$s = f(x)^{\mathsf{T}} g(x) \tag{2.34}$$

with $s \in \mathrm{Re}^{N\times N}$ and apply a softmax per row to the result

$$\beta_{j,i} = \frac{\exp\left(s_{i,j}\right)}{\sum_{i=1}^{N} \exp\left(s_{i,j}\right)}. \tag{2.35}$$

$\beta_{j,i}$ represents the attention level of the model to location i when synthesizing the j^{th} region in the input data. The attention map β is then applied to the values h through matrix multiplication and the result is scaled back to C channels:

$$o = v(\beta\, h(x)), \text{ where } \quad v = W_v z \tag{2.36}$$

with $o \in \mathrm{Re}^{C\times N}$ the output of the attention layer, v the scaling operation and $W_v \in \mathrm{Re}^{C\times C'}$ the scaling trainable weights.

Finally, the output of the attention layer o is scaled and summed with the input feature maps x:

$$y = \gamma o + x \tag{2.37}$$

with γ the learnable scaling scalar. γ is initialized as 0 to let the network first learn local dependencies in the data through the convolution layers before focusing on long-range ones.

The whole self-attention procedure is summarized in Figure 2.6.

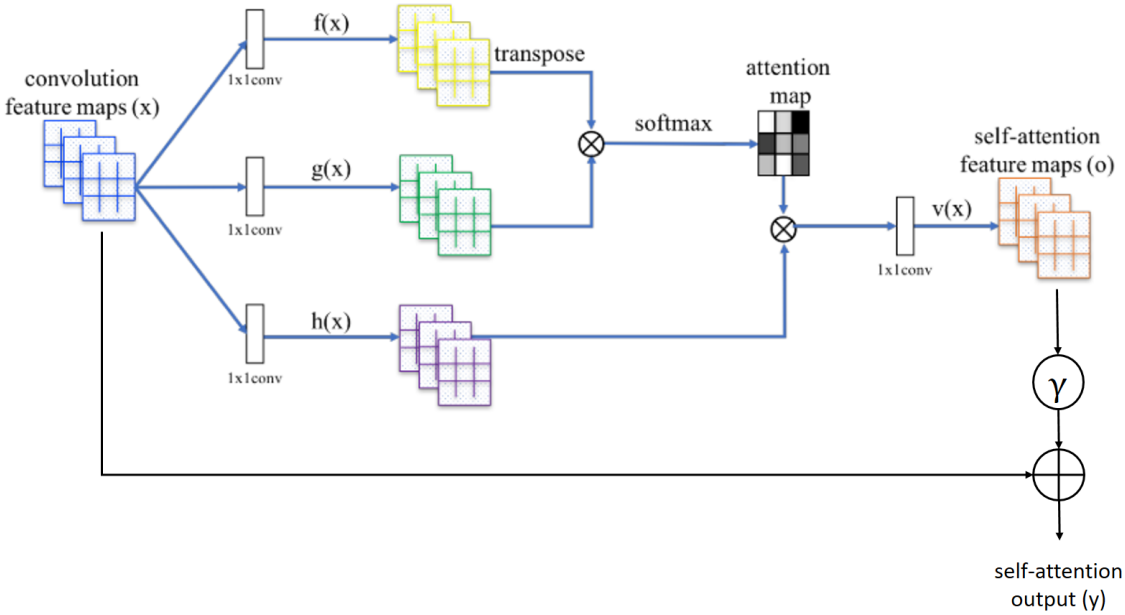


Figure 2.6 – Self-attention computation. The input feature maps are transformed to queries, keys and values (f, g, h). The queries and the keys are combined to produce the attention map. The latter is used to weight the contribution of all the pixels at every location in the values. The resulting values are transformed channel-wise, scaled by γ and summed with the input feature maps. f, g, h and v can be efficiently computed as 1×1 convolutions. Source [Zhang et al., 2019], modified.

Pros and Cons. Global self-attention adds to convolutional networks the ability to learn long-range dependencies for each location (*i.e.*, pixel, in images) progressively during the training phase. Increasing the reduction ratio allows the model to be more efficient. However, it is still a costly attention mechanism, especially in the first layers of the networks, where feature maps are close to the input in size.

2.3.2 Channel-Wise Attention: Squeeze-and-Excitation

In computer vision, global and local self-attention can be considered as spatial attention mechanisms. The network learns to capture long-term dependencies in data, with respect to space, by weighting each pixel. On the contrary, to capture channel relationships, Hu et al. [Hu et al., 2018] introduce a lightweight channel-wise attention denoted Squeeze-and-Excitation.

Principle. To model the dependencies between channels of the convolution layer output, the authors propose a mechanism to perform feature recalibration. The objective is to emphasize relevant feature channels and diminish less useful ones. This mechanism consists of three operations: squeeze, excite and scale. The squeeze operation produces a channel descriptor of the input and is followed by an adaptive recalibration, the excitation, and a scale operation that weights the input channels. The excitation acts as a bottleneck parametrized by a reduction ratio.

Squeeze-and-Excitation Computation. Consider a Squeeze-and-Excitation module inside a convolutional neural network. $u \in \mathrm{Re}^{C\times W\times H}$ represents the output of the previous hidden layer, with C the number of channels, W the width and H the height of the feature maps. To capture global spatial information, we first apply to u the squeeze operation F_{sq}. It consists in applying global average pooling to each channel:

$$\begin{aligned} z_c &= F_{sq}(u_c) \\ &= \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H} u_c(i,j) \end{aligned} \tag{2.38}$$

with z_c the channel descriptor of channel c.

We then recalibrate the channel descriptors with the excitation operation F_{ex}. To capture non-linear dependencies between channels, F_{ex} is a simple gating mechanism with a sigmoid activation:

$$\begin{aligned} s &= F_{ex}(z) \\ &= \sigma(W_{exp}\,\delta(W_{comp}\,z)) \end{aligned} \tag{2.39}$$

with σ the sigmoid function, δ the ReLU function, $W_{comp} \in \mathrm{Re}^{C'\times C}$ and $W_{exp} \in \mathrm{Re}^{C\times C'}$ the trainable weights of the fully connected layers forming the bottleneck around the ReLU. The final sigmoid function smoothly bounds each value of s between zero and one to produce the channel weights. The bottleneck helps reduce the module complexity and improve generalization. It is controlled by the reduction ratio $r = \frac{C}{C'}$.

Finally, we scale each channel of the input feature map u:

$$\begin{aligned} \widetilde{x}_c &= F_{scale}(u_c, s_c) \\ &= s_c \cdot u_c \end{aligned} \tag{2.40}$$

where s_c is a scalar.

The whole Squeeze-and-Excitation procedure is illustrated in Figure 2.7.

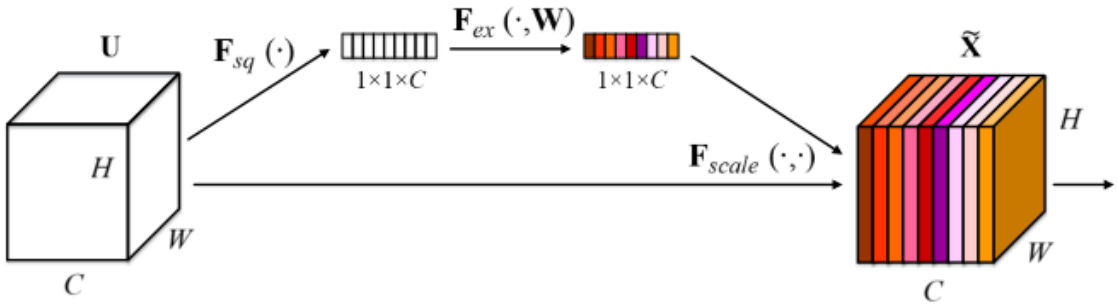


Figure 2.7 – Squeeze-and-Excitation computation. The input feature maps are first squeezed per channel through global average pooling. The excitation block learns from the channel descriptors the scale that is then applied to the input features maps to emphasize relevant channels. Source [Hu et al., 2018], modified.

Pros and Cons. Squeeze-and-Excitation is a light-weight attention mechanism that can be used throughout the network. It dynamically enforces feature discriminability. However, the squeeze operation reduces the spatial information to a unique descriptor per channel. It does not capture spatial long-range dependencies.

2.3.3 Spatial and Channel-Wise Attention: Dual Attention

To make the most of both channel-wise and spatial attention methods, Sun et al. [Sun et al., 2020] proposes dual attention. Their goal is to improve U-Net [Ronneberger et al., 2015] interpretability and robustness. Dual attention is similar to SCA-CNN [Chen et al., 2017] and CBAM [Woo et al., 2018]. However, SCA-CNN is designed for image captioning and its attention modules take into account the hidden state of the LSTM [Cheng et al., 2016] producing the caption. Besides, although CBAM improves the Squeeze-and-Excitation method, its spatial attention has less complexity than the one of dual attention.

Principle. Dual attention combines Squeeze-and-Excitation with a spatial attention path. The latter, simpler than self-attention, compresses the number of input channels to one. It then applies a sigmoid on the resulting pixel values to produce an attention map that scales the output of the Squeeze-and-Excitation.

Dual Attention Computation. The dual attention is composed of a channel-wise attention path, that is performed with a Squeeze-and-Excitation module, and a spatial attention path. The Squeeze-and-Excitation module is detailed in Section 2.3.2.

The spatial attention path is much simpler than the self-attention mechanism described in Section 2.3.1. Consider a dual attention module inside a convolutional neural network. $x \in \mathrm{Re}^{C\times W\times H}$ represents the output of the previous hidden layer. We first divide the number of channels of x by 2 with a normalized 1×1 convolution f:

$$z = \delta(Bn(f * x)) \tag{2.41}$$

with δ the ReLU [Nair and Hinton, 2010] function, Bn the batch normalization [Ioffe and Szegedy, 2015] method.

Then, a 1×1 convolution g is applied to z to reduce its number channels to 1, followed by a sigmoid function σ that maps the resulting pixels into the range $[0, 1]$:

$$s = \sigma(g * z) + 1 \tag{2.42}$$

where we add one so that the spatial attention can only amplify the features.

Finally, we combine the spatial attention map s and the result of the Squeeze-and-Excitation module with the Hadamard product. The whole procedure is illustrated in Figure 2.8.

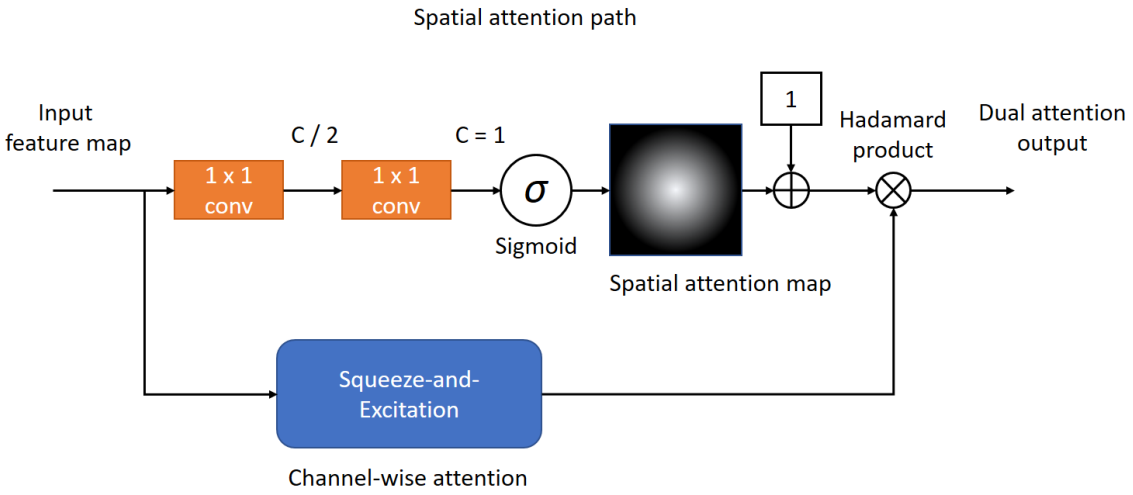


Figure 2.8 – Dual attention computation. The dual attention is composed of a channel-wise attention path, that is performed with a Squeeze-and-Excitation module (see Section 2.3.2 for details), and a spatial attention path. The latter consists of two 1×1 convolutions to reduce the number of channels to one and a sigmoid function to map the resulting pixels in the range $[0, 1]$. We then add one to the spatial attention map so that it can only amplify the features, and use it to weight the features produced by the channel-wise attention path.

Pros and Cons. Dual attention performs spatial and channel-wise attention by combining the Squeeze-and-Excitation and a simple spatial attention path. As both are light weight, it is an efficient attention mechanism. However, the spatial attention path learns pixel importance channel-wise thanks to two convolutions with 1×1 kernels. It does not capture long-range dependencies between pixels, as self-attention does.

2.3.4 Summary

Over the last years, attention mechanisms have become prevalent in deep learning architecture design. They help capture long-range and multi-level dependencies in input data. By focusing on relevant features, they allow improving the performance and the robustness of the models.

Spatial attention methods focus on the feature importance at the pixel level. Self-attention learns long-range dependencies between locations in the data. These dependencies are the same for all the channels. Self-attention has an important computational cost, in particular in the first layers of the model, where the feature maps are still large.

On the other hand, channel-wise attention methods focus on the feature importance at the channel level. Squeeze-and-Excitation learns multi-level dependencies through channel descriptors. All the pixels of a channel are weighted with the same learned feature importance. Squeeze-and-Excitation is light weight and can be used throughout the network.

To make the most of both attention types, we can combine them. Dual attention merges a simple spatial-attention path and the Squeeze-and-Excitation method. Its spatial-attention mechanism is much simpler than self-attention. It does not model the pixel dependencies as finely but it is thus less costly.

2.4 Deep Learning for IACT Data Analysis

Recently, some effort has been made to explore deep learning techniques to solve astrophysical problems [Kim and Brunner, 2016, Huennefeld, 2017, Brunel et al., 2019]. In particular, deep learning has been investigated for IACT data analysis, from muon image analysis [Feng et al., 2016] to gamma event reconstruction of H.E.S.S. and CTA data.

2.4.1 H.E.S.S. Data Analysis with Deep Learning

For the H.E.S.S. data analysis, Holch et al. [Holch et al., 2017] propose a shallow neural network to realize gamma/proton separation (*i.e.*, classification) and energy and direction reconstruction (*i.e.*, regression). The model is composed of three convolutional layers and three fully connected ones. As the LST of CTA, H.E.S.S. telescopes produce hexagonal pixel images. The authors transform them into square pixel ones via a rebinning method before feeding the network. To handle the stereoscopic information, they combine the images coming from the four telescopes into a single one per event. They obtain a good performance for the classification task without any selection cut. For the regression tasks, they present preliminary results showing that their architecture does far better than random guessing. However, their approach, limited to single task models, does not avoid the degeneracies introduced by the atmospheric detection of the gamma rays. It could also impact the computational cost of the full event reconstruction.

In order to improve data handling for stereoscopic analysis, Shilon et al. [Shilon et al., 2019] propose a combination of a convolutional neural network and a recurrent neural network, denoted CRNN, to realize the gamma/proton separation. CRNN is composed of a three convolutional layer backbone, a recurrent layer (LSTM [Cheng et al., 2016]) and two fully connected layers. The image produced by each telescope is considered as one element of a sequence, ordered by image total intensity. The authors show that this method relax the discrepancy on the classification task between the performance on simulations and on real data. To solve the direction reconstruction task, they adopt a different strategy. They combine the images produced by the four telescopes into a single one through a channel representation, one telescope per channel. They propose a neural network composed of five 2-1-CL layers and four fully connected ones. A 2-1-CL consists of two convolutional layers, each followed by a non-linearity, and a subsampling layer. This architecture achieves performance close to the state-of-the-art one on simulated data. However, thanks to the analysis of real data, the authors observe an evident discrepancy

between the performance on simulated data and the one on real data. Moreover, they do not realize the energy reconstruction task.

To solve the real data discrepancy for the gamma/proton separation, Parsons et al. [Parsons and Ohm, 2019] propose to combine the IACT images latent representation and the parameters obtained with the standard method. The designed architecture consists of an image path and a parametric path. The image path is composed of two convolutional layers, a fully connected layer and a recurrent one. It produces the latent representation of the data. The parametric path is composed of a recurrent layer and a fully connected one. Its input is the standard method parameters. The outputs of both image and parametric paths are concatenated and fed to an LSTM, followed by a final fully connected layer. As in [Shilon et al., 2019], the authors consider the images coming from the four telescopes as sequences. However, compared to a baseline, they still observe a discrepancy between the performance on simulated and real data. Moreover, the benefits of their model compared to [Shilon et al., 2019] is not clear.

2.4.2 CTA Data Analysis with Deep Learning

To analyze CTA data in a stereoscopic mode, Mangano et al. [Mangano et al., 2018] present narrow convolutional neural networks to solve gamma/proton classification, energy and direction reconstruction tasks, one network per task. The authors combine the images of the four LSTs present in the North site by summing their pixels, after having transformed them to square pixel images with oversampling (see Section 3.3.1 for details about resampling methods). They apply a strong selection to the data, as among other cuts, they keep only the events that triggered the four telescopes. The proposed architecture is composed of four convolutional layers and one fully connected one. On the energy and direction reconstruction, their model performs slightly worse than the baseline that is the expected CTA performance (at the time of their paper). However, the comparison might not be relevant. First, the expected performance for both energy and direction reconstruction tasks is computed with all the 19 telescopes from the North site. On the other hand, Mangano et al. apply a much stronger selection to the data that allows selecting easier to reconstruct events. They do not compare the classification performance of their architecture with other methods.

Nieto et al. [Nieto et al., 2017] follow a different strategy. As a proof of concept, they probe two very deep networks, a ResNet-50 [He et al., 2016b] and an Inception V3 [Szegedy et al., 2015], for gamma/proton classification in a single-telescope analysis context. The authors divide the data into three energy bins and train one model per bin. Both networks have similar results. However, they clearly underperform compared to a standard analysis chain based on geometrical parameters (Hillas) and boosted decision trees.

2.4.3 Summary

The deep learning papers about the H.E.S.S. data analysis are in a much advanced state than the ones about the CTA data analysis. This is expected as the H.E.S.S. is a mature observatory while the CTA is in the construction phase. Shilon et al. [Shilon et al., 2019] present results on simulated data that are better than the state-of-the-art for the gamma/proton separation and are close to the state-of-the-art for the direction reconstruction.

However, they do not solve energy reconstruction. Moreover, they observe a discrepancy with real data, that is not solved by Parsons et al. [Parsons and Ohm, 2019].

The published works about the CTA data analysis show preliminary results. These experiments are either realized in very constrained conditions or focused on a single task. The authors present proofs of concept that the deep learning is worth exploring.

All of these papers present promising results. However, the adaptation to real data is challenging. This is expected as simulations are by nature an imperfect representation of the reality. Besides, all these contributions have handled the different reconstruction problems as single tasks, without considering the strong interdependence between them. Considering the IACT data analysis as a multi-task problem could improve the generalization ability of the model and its robustness.

3

Indexed Operations for Deep Learning

The goal of this thesis is to perform full event reconstruction from IACT data with deep learning, in particular from CTA images. As detailed in Section 1.3, the CTA is composed of telescopes whose cameras have various shapes and pixel organizations, including hexagonal lattices. We have seen in Section 2.1.1 that Convolutional Neural Networks have greatly contributed to deep learning success. However, classic convolutional kernels have been developed for rectangular and regular pixel grids as found in traditional images. In this chapter, I discuss the standard solutions to overcome this issue. I also introduce IndexedConv, an original method to apply convolution to any kind of pixel organization. This method has been published as a conference paper [Jacquemont et al., 2019a] and presented at VISAPP 2019.

3.1 Convolution and Deep Learning

As we have seen in Section 2.1.1, convolution plays a key role in modern neural network architectures. It has been a crucial element in the development of deep learning itself and in the improvement of performance during the last decade.

3.1.1 The Convolution Operator

Mathematically, convolution is a bi-linear operator defined for continuous functions by the equation:

$$h(t) = \int_{-\infty}^{\infty} f(x)\, g(t-x)\, dx \tag{3.1}$$

where h, the result function also denoted $f * g$, indicates how the shape of g modifies the shape of f. The convolution is commutative, meaning that $f * g = g * f$. Figure 3.1 illustrates the process of convolution for a boxcar function (f) and a ramp function (g). g is reflected and slides over f. At each position of g the integral of the dot product $f.g$ is computed as exemplified by point a.

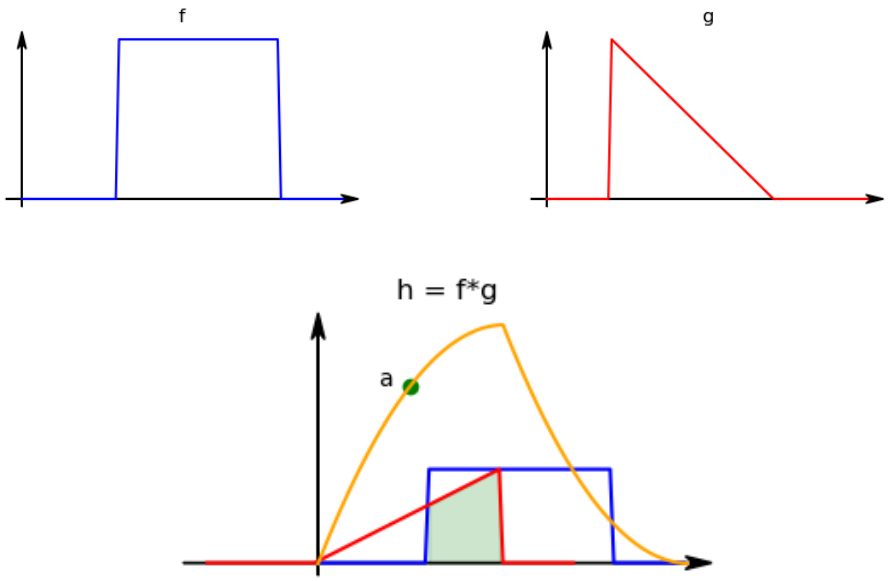


Figure 3.1 – Convolution example of function f (in blue) and function g (in red). The function h (in orange) represents the result of the convolution. The amount of green corresponds to the area of the product $f(x).g(t-x)$ computed by the convolution integral yielding the green point belonging to $h(t)$.

It is worth noticing that, according to the Convolution Theorem, convolution in the temporal (or spatial) domain is equivalent to a point-wise multiplication in the frequency domain (and vice versa):

$$\mathscr{F}(f*g) = \mathscr{F}(f)\cdot\mathscr{F}(g) \tag{3.2}$$

with $\mathscr{F}$ the Fourier transform operator.

In the discrete space, convolution is given by

$$h(n) = \sum_{m=-\infty}^{\infty} f[m]\,g[n-m]. \tag{3.3}$$

Usually, in practical applications m belongs to a finite space M. We can then consider convolution as a linear operation performed on data (the g function) over which neighborhood (of size M) relationships between elements can be defined. The function (f), the signal is convoluted with, is called *convolution kernel*. The convolution operation is then a weighted sum over the input neighborhood, the set of weights being the convolution kernel. It can be rewritten as:

$$O_j = \sum_{k=1}^{K} w_k I_{N_{jk}} \tag{3.4}$$

where K is the number of elements in the kernel, w_k is the value of the k-th weight in the kernel, and N_{jk} is the index of the k-th neighbor of the j-th neighborhood of input data I.

In the specific case of 2D images we define convolution as:

$$O_{ij} = \sum_{k=-K}^{K}\sum_{h=-H}^{H} w_{kh} I_{(i-k)(j-h)} \tag{3.5}$$

where the convolution kernel W is a square matrix of size $(2K+1, 2H+1)$ and neighborhood is implicitly defined through corresponding relative locations from the center pixel. We can define analogous expressions for N dimension data.

Convolution is widely used in signal processing as a filtering operator. In particular in the image processing domain, convolution filtering is applied for edge detection [Davis, 1975, Shen and Sethi, 1996, Basu, 2002], image blurring and sharpening and feature extraction [Getreuer, 2013]. In the widespread canny edge detection [Canny, 1986] for example, the first two steps of the algorithm involve convolution. It starts with a Gaussian kernel to smooth the image and remove the noise. Then it applies edge detection filters (Sobel filters for instance). Finally, a threshold and a hysteresis are applied to produce the refined edges. This example is typical of classical image processing. The parameters of the convolution kernels are defined by an expert, depending on the use case. On the contrary, in deep learning applications, the network learns these parameters during the training phase.

3.1.2 Convolution in Deep Neural Networks

In this section I detail the general formulation of convolution for neural networks. I also list concepts widely used with convolution in deep learning context.

Inside convolutional neural networks, convolution inputs have generally multiple channels (also called features). All input channels contribute to the output. The convolution kernel has then an additional dimension with a size equal to the number of channels in the input. Convolution is simply obtained as the sum of dot products over all the individual channels to produce output values. Equation 3.6 shows the 2D image convolution case with C input channels.

$$O_{ij} = \sum_{c=1}^{C} \sum_{k=-K}^{K} \sum_{h=-H}^{H} w_{ckh} I_{c(i-k)(j-h)} \tag{3.6}$$

with the kernel W of shape C, K, H. As convolution outputs one feature, in CNN there are as many convolution kernels as expected output features for a particular layer.

As the kernel is the same for every location over the input data, the output values of convolution only depend on the input values situated in the local neighborhood to which the kernel is applied. This induces translation invariance property, which is particularly important for the classification performance of CNN, as highlighted in Section 2.1.1.

Stride. It is worth noting that the convolution can be computed over a subset of the input elements. When the kernel slides over the input, it does not necessarily visit each location. On regular lattices this results in visiting one every n location in each direction, an amount generally referred as *stride*. The wider the stride, the smaller the size of the output feature.

Dilation and Deformable Convolution. Moreover, the neighbors implied in convolution at a particular location do not need to be adjacent. We can define neighborhoods arbitrarily, in terms of shape and location of the neighbors as illustrated in Figure 3.2. In case of regular lattices the amount of separation between the elements of a convolution kernel in each direction is referred to as *dilation* or *atrous* convolution [Holschneider et al., 1990]. The wider the dilation, the further away from the center the kernel reaches

out in the neighborhood. Moreover, the dilation is not necessarily fixed and the same for every pixel of the neighborhood. Reference [Dai et al., 2017] introduces deformable convolution that learns dilations through offset fields.

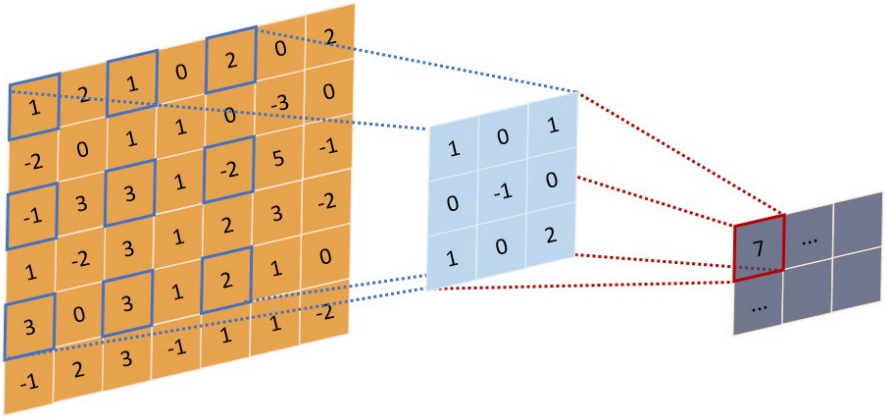


Figure 3.2 – Example of a dilated or atrous convolution on an image. The input image is in orange. The neighbors implied in the convolution at the particular location of this example are highlighted in blue. The convolution kernel is the blue matrix and the output image is the gray one.

Padding. However, convolution cannot be performed on location where part of the neighborhood is not defined, such as at the border of an image. In this case, either the location is skipped and the output is smaller than the input, or neighborhoods are extended beyond the reach of the input, which is referred to as *padding*. Input values in the padded region can be set to zero (zero padding) as illustrated in Figure 3.3, or reproduce the same values as the closest neighbors in the input (same padding).

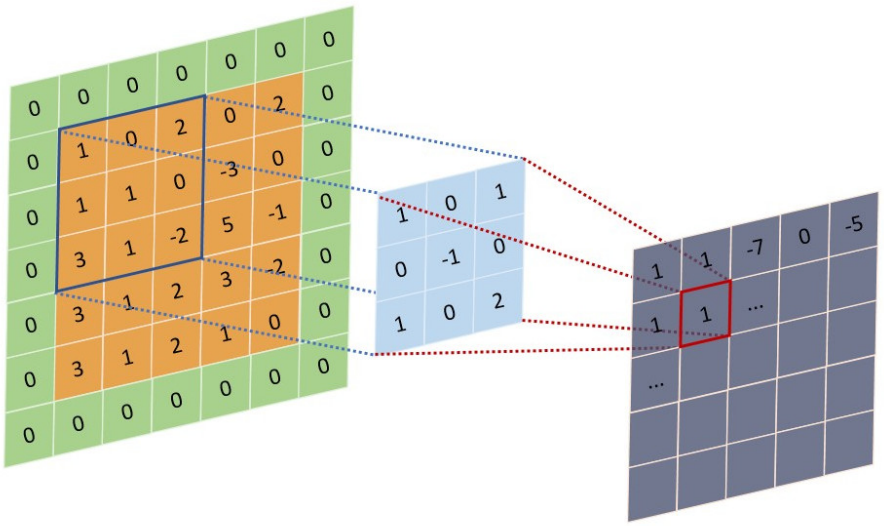


Figure 3.3 – Example of a zero padded convolution on an image. The input image is in orange, the zero padding in green. The convolution kernel is the blue matrix and the output image the gray one. Thanks to the padding, the output image has the same size as the input image.

Separable Convolution. From Equation 3.6, a convolution kernel has then $C \times K \times H$ parameters. As neural networks are becoming deeper and deeper, their total number of parameters explode . To reduce it, [Sifre and Mallat, 2014] proposes *depthwise separable convolution* that is the main component of MobileNets [Howard et al., 2017, Sandler et al., 2018, Howard et al., 2019], and Xception [Chollet, 2017]. It is composed of a spatial convolution performed over every input channel with the same kernel of shape $1, K, H$, and a 1×1 convolution to combine the resulting channels. This results in $K \times H + C$ kernel parameters.

3.1.3 Implementations for Deep Learning

The use of Graphic Processing Units (GPU) played a dramatic role in the emergence of deep learning, accelerating the training time of Neural Networks by a very large factor (70 in [Raina et al., 2009]) compared to CPU implementation. Indeed, GPUs are designed for massively parallel computations, in particular matrix operations. Deep learning is a highly parallelizable process, as data are considered as batches and most of Neural Networks operations can be achieved through matrix to matrix multiplication.

There exist mainly three implementations (with variations) of the convolution operation for execution on GPU, in particular in the CUDA language in the cuDNN library [NVIDIA, 2014] for NVIDIA GPU boards. These implementations enable highly optimized computations for deep learning applications.

3.1.3.1 GEneral Matrix to Matrix Multiplication (GEMM)

The GEMM cuDNN implementation of convolution has been proposed in [Chetlur et al., 2014]. It lowers the convolution into a matrix multiplication following BLAS implementation for CPU [Chellapilla et al., 2006], as illustrated in Figure 3.4.

The first step of the process, the *im2col* operation, unfolds and duplicates the input into a 2D matrix where each column reports the values of the neighbors to consider for each of the input samples and each of the channels. The *im2col* operation is easily reversible. This is critical for deep neural networks training steps where the backward gradient propagation is applied in order to optimize the network parameters.

The next step consists in rearranging the kernel weights into a 2D matrix (F_m on Figure 3.4) such as the scalar product of one row of F_m by one column of the unfolded input corresponds to the scalar product performed by the convolution, as defined in Equation 3.6.

The final step is the matrix multiplication itself. The first row of the output matrix corresponds to the output of the first filter, *i.e.*, the first channel of the output features. The output needs then to be reshaped to match input dimensions (e.g. 2D images).

This approach requires a significant amount of memory to store the duplicated information of the input. The memory needed for the unfolded input is typically equal to:

$$\frac{Memory_{input} \times Size_{kernel}}{Stride}.$$

However, thanks to the opportunity for vectorization and cache friendliness of the general matrix multiply operations (GEMM), the resulting gains in efficiency outweigh the additional memory consumption.

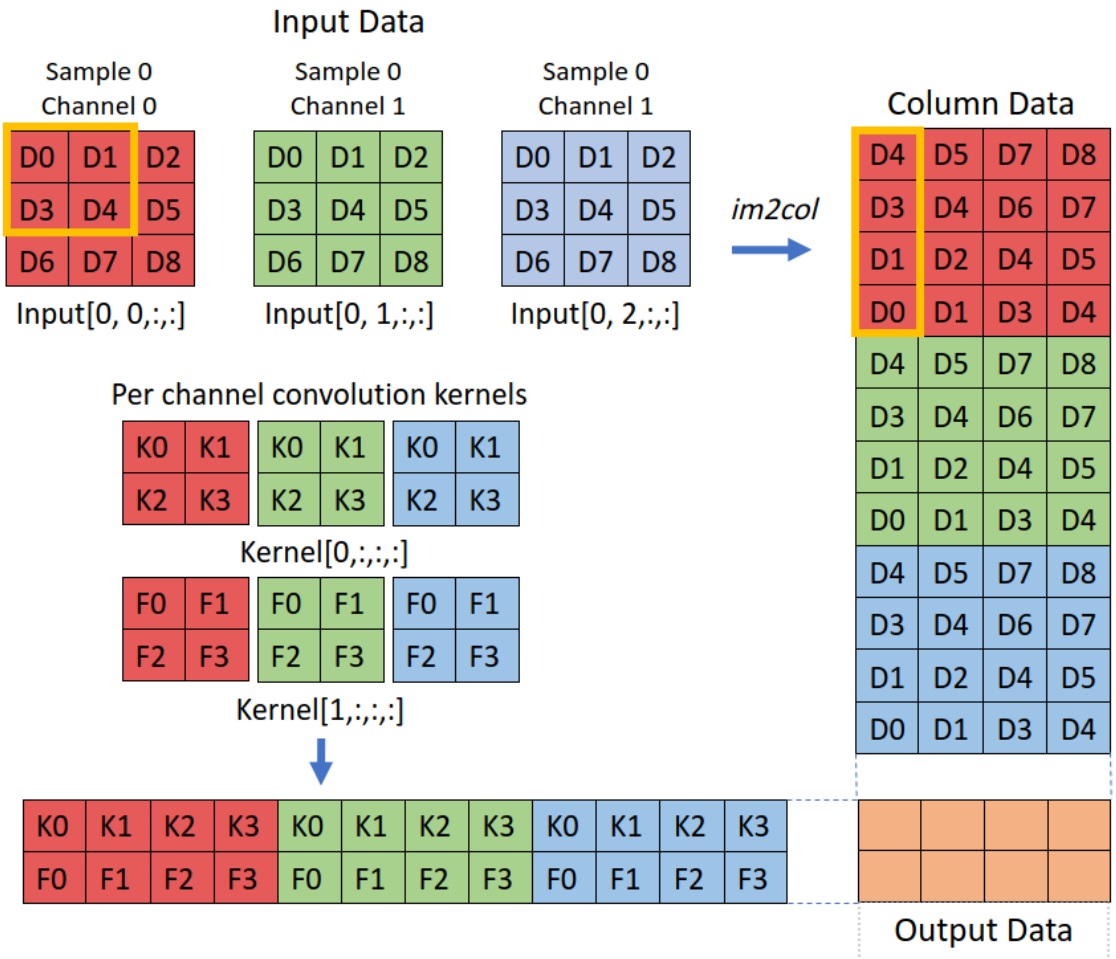


Figure 3.4 – GEMM implementation of the convolution. The input data is in the shape (batch size, feature maps, width, height). In the example presented, it is composed of three features (red, green and blue) of size 3×3. We focus on the first element of the batch. The convolution consists of two kernels of size $3 \times 2 \times 2$, 3 corresponding to the number of feature maps in the input data. Each kernel produces one output feature map. The *im2col* operation unfolds and duplicates the input along the columns of the Column Data matrix. The orange window, corresponding to the convolution kernel size, slides over the features of the input. At every position, the patch of input data involved is vectorized and copied in the corresponding column, one feature at a time. The convolution kernels are then vectorized into the kernel matrix, one kernel per row. Finally, the kernel matrix is multiplied with the Column Data matrix to produce the output of the convolution, one feature map per row. The output needs to be reshaped to have the same number of dimensions as the input. In the example presented, this results in a final output of shape (batch size, 2, 2, 2).

3.1.3.2 Fast Fourier Transform based Convolution (FFT)

As mentioned in Section 3.1.1, according to the Convolution Theorem the Fourier transform of the convolution of two signals is equal to the element-wise multiplication of the Fourier transform of the same signals. In modern CPU and GPU architectures, Arithmetic Logic Units contain binary multipliers computing multiplication in a single clock cycle. Multiplication is then less expensive to compute than convolution. The benefit of FFT-based convolution depends on the computational cost of the FFT of the input and the kernel and the inverse FFT of the result of the element-wise multiplication. As these two operations have a high complexity, FFT-based convolution may be more interesting deeper in the CNN [Jordà et al., 2019] where features are smaller and layers contain several convolutions. Thus the FFT of the input of the convolution layer can be amortized by reusing it across all the convolutions.

It is also worth noticing that both the input and the kernel must have the same size. The smaller has to be padded to the size of the wider, increasing the memory needed. This is particularly costly in the first layers of CNN, where the kernels are small compared to the features.

3.1.3.3 Winograd Algorithm

The Winograd algorithm for convolution relies on the fact that convolution can be expressed as a polynomial multiplication [Barabasz and Gregg, 2019] and the minimal filtering algorithms introduced in [Winograd, 1980]. The fast algorithm implemented in cuDNN and described in [Lavin and Gray, 2016] compute the Winograd algorithm for image tiles of size depending on the convolution kernel size. The Winograd algorithm reduces the number of multiplications required but increases the number of additions.

3.1.3.4 Performance

In [Jordà et al., 2019] the authors analyze the performance of these three algorithms implemented in cuDNN in various conditions including kernel size and batch size and variation in implementation. They provide recommendations to select the convolution algorithm depending on the convolution parameters. Actually, cuDNN has a benchmark mode to select the best algorithm depending on the convolution parameters and the input size.

3.2 Unconventional Sensors and Deep Learning

Traditional images, as the ones taken by consumer cameras, have regular pixel grids, with rectangular pixel shape, as illustrated in Figure 3.5. However, some imaging sensors present different shapes and do not have regularly spaced nor rectangular pixel lattices. This is particularly the case in science experiments where sensors use various technologies and must answer specific technological needs. Examples (displayed in figure 3.6) of such sensors in physics include the IceCube experiment [Huennefeld, 2017] and the KM3NeT experiment [Katz, 2012], two neutrino observatories situated respectively in the Antarctic and the deep Mediterranean seas. They are both composed of three-dimensional arrays of light sensor modules. However, we can consider that these sensors produce images, as there is a spatial dependency between their sensing units (their pixels).

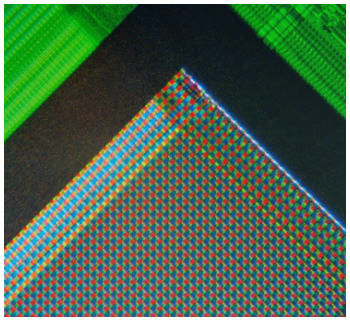
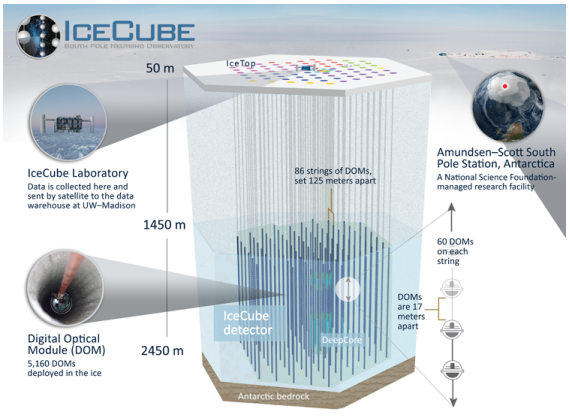


Figure 3.5 – Micrograph of the corner of the photosensor array of a webcam digital camera. Credit: Wikimedia/Natural Philo.



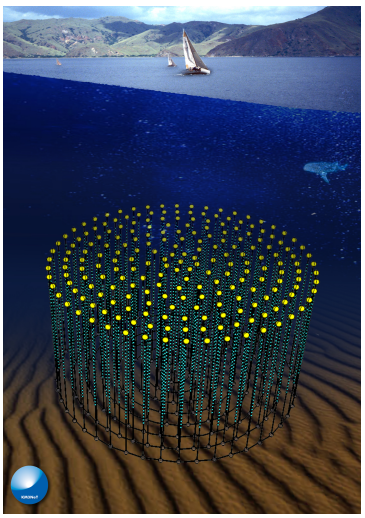


Figure 3.6 – Example of neutrino observatories presenting unconventional sensor grids. On the left, the IceCube cubic-kilometer particle detector. Credit: IceCube Collaboration. On the right, the KM3NeT three-dimensional arrays. Credit: KM3Net Collaboration.

Among the unconventional grid images, the hexagonal lattice present in some telescopes of the CTA is a particular case. Although their pixels are not rectangular, their grid is regular. Even if hexagonal grid data processing is not usual for general public applications, several other specific sensors make use of hexagonal sampling. The Lytro light field camera [Cho et al., 2013] is a consumer electronic device example. Several Physics experiments other than CTA also make use of hexagonal grid sensors, such as the H.E.S.S. camera [Bolmont et al., 2014] or the XENON1T detector [Scovell, 2013], as illustrated in Figure 3.7. Hexagonal lattice is also used for medical sensors, such as DEPFET [Neeser et al., 2000] or retina implant system [Schwarz et al., 1999].

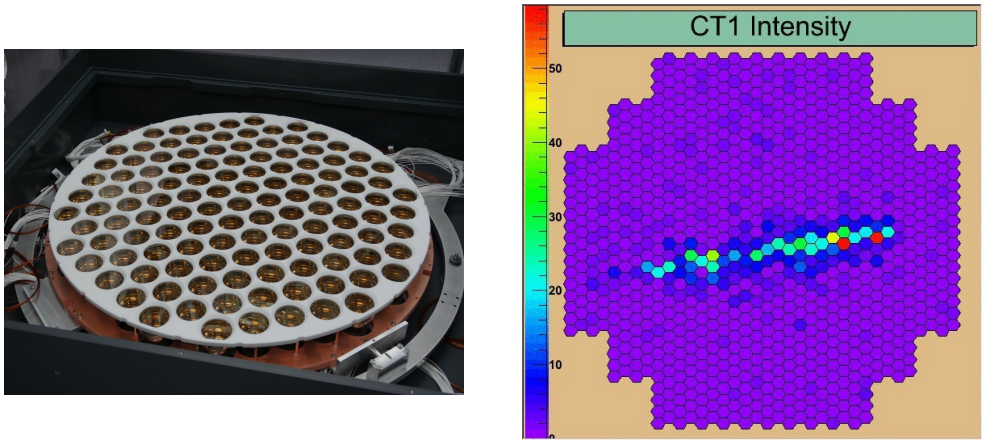


Figure 3.7 – Example of physics experiments presenting non-Cartesian sensor grids. On the left, the XENON1T photo-multiplier tube layout. Credit: XENON Collaboration. On the right, an image from the H.E.S.S. camera. Credit: The H.E.S.S. collaboration

Moreover, hexagonal lattice is a well-known and studied grid [Sato et al., 2002, Shima et al., 2010, Asharindavida et al., 2012, Hoogeboom et al., 2018] and offers advantages compared to square lattice [Middleton and Sivaswamy, 2001] such as higher sampling density and a better representation of curves. In addition, some more benefits have been shown by [Sousa, 2014, He and Jia, 2005, Asharindavida et al., 2012] such as equidistant neighborhood, clearly defined connectivity, and smaller quantization error.

The data produced by these unconventional sensors might benefit from the recent advances in computer vision, especially in deep learning. As we have seen in Section 2.1.2, there exist several deep learning frameworks to ease the task of training Deep Neural Networks with computers. However, all these frameworks are designed to handle standard images with regular grid and rectangular pixel shape. They process images as matrices and thus benefit from GPU accelerated hardware (mainly via CUDA) for efficient computations.

Processing unconventional lattice images with standard deep learning frameworks requires then specific data manipulation and computations that need to be optimized on CPUs as well as GPUs. In the following sections of this chapter, I will review different methods to handle unconventional images with deep learning in the special case of hexagonal pixel images.

3.3 Handling Non-Rectangular Pixels: The Particular Case of Hexagonal Pixels

A common approach to use traditional convolution neural network framework out of the box with unconventional images is to resample them into a Cartesian grid. For regular lattices, such as hexagonal ones, it is also possible to apply geometrical transformation to the images to shift them into Cartesian grids, with the help of a specific addressing system. In that case, an adapted convolution operator can be used to respect the original layout of the images. Finally, an alternative emerged during this thesis. Geometric deep learning [Bronstein et al., 2017], and in particular graph convolution, manages to apply convolution to graphs or meshes [Verma et al., 2018].

3.3.1 Resampling to a Cartesian Grid

The most common method to apply deep learning with existing frameworks to unconventional pixel images is resampling them to a rectangular grid. It consists in mapping the original pixel layout onto a Cartesian one through a linear operation. In this section I describe five interpolation methods: oversampling, rebinning, nearest neighbor, bilinear and bicubic interpolations. As we want to solve the issue of applying deep learning to CTA data, I focus on their implementation for hexagonal grid images.

3.3.1.1 Oversampling

Oversampling hexagonal pixel images to Cartesian grid is widely used in IACT data analysis [Feng et al., 2016, Holch et al., 2017, Nieto et al., 2017, Shilon et al., 2019]. Among unconventional pixel grids, hexagonal lattice is a particular case as this is a regular grid. As illustrated in Figure 3.8, it consists in dividing each hexagonal pixel into $n \times n$ pixels and results in a rectangular grid. This rectangular lattice is then stretched to a square one. We can weight the pixel value obtained by n^{-2} to keep the overall intensity of the image. This is especially important for IACT data analysis because, as we have seen in Section 1.2, the overall intensity of the image is deeply related to the energy of the incoming particle that produced the image. Usually, n is set to two to minimize the memory footprint of the resampled image.

Pros and Cons. Oversampling is a simple method. However, it induces in an increase of the size of the image in pixels, needing more memory to store the resampled images. This also results in more operations during convolutions as the amount of locality is multiplied by n^2. Moreover, oversampling alters the neighborhood of hexagonal pixels, as illustrated in Figure 3.15. Finally, the resampled image is slightly stretched along one axis and compressed along the other.

3.3.1.2 Rebinning

As introduced in [Holch et al., 2017, Shilon et al., 2019], we can consider hexagonal pixels as bins collecting photons and the whole image as a hexagonal histogram. This histogram can be finely sampled [Nieto et al., 2019b] and converted to a square histogram of arbitrary resolution, as illustrated in Figure 3.9.

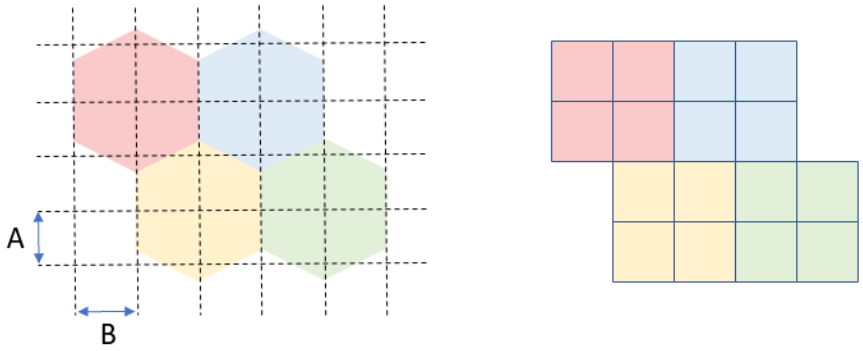


Figure 3.8 – Oversampling hexagonal pixels with n set to two. *On the left:* Dividing hexagonal pixels in $n \times n$ results in rectangular pixels ($A \approx 0.866 \cdot B$). *On the right:* The rectangular pixels are slightly stretched to square ones.

Pros and Cons. The rebinning operation preserves the overall intensity of the images. It can be efficiently implemented as a sparse matrix-vector multiplication. The size of the resampled image is arbitrary and so does not necessarily induce an increase of the memory consumption. However, as oversampling, rebinning alters the hexagonal neighborhood albeit less strongly if we choose carefully the resampling grid. It also tends to smooth the resampled image.

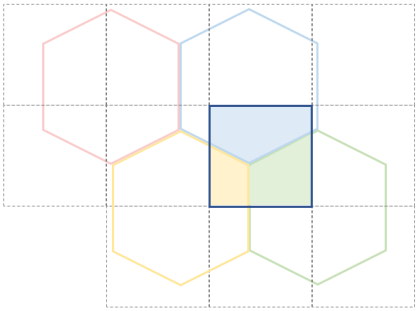


Figure 3.9 – Rebinning hexagonal pixels with a finely sampled grid. Hexagonal pixels are finely sampled. A Cartesian grid is applied to the hexagonal one. The resampled pixel value is the sum of the contributions of corresponding hexagonal pixels, the yellow, green and blue surfaces in the example presented.

3.3.1.3 Nearest Neighbor Interpolation

For nearest neighbor interpolation, a square grid of arbitrary resolution is applied to the hexagonal one. As illustrated in Figure 3.10, the nearest neighbor algorithm is then used to assign pixel values.

Pros and Cons. As for the rebinning method, the size of the resampled image is arbitrary and so does not necessarily induce an increase of the memory consumption. However, to preserve the overall pixel intensity of the image that is important for IACT data analysis, a normalization needs to be applied. Moreover, as the rebinning method, nearest neighbor interpolation alters the hexagonal neighborhood.

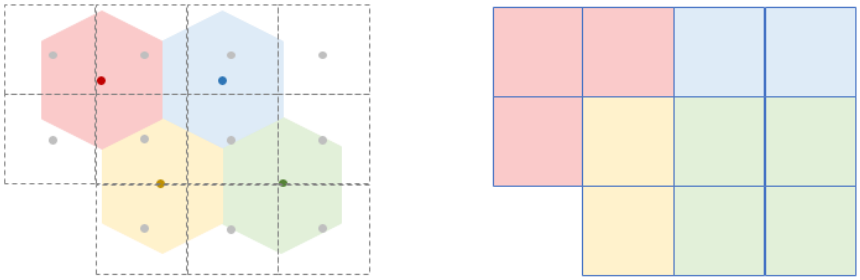


Figure 3.10 – Interpolation of hexagonal pixels to square ones with the nearest neighbor algorithm. A Cartesian grid is applied to the hexagonal one. The colored dots represent the center of the hexagonal pixels, the grey ones the center of the resampled pixels. The value of the latter is defined according to their nearest neighbor in the hexagonal lattice.

3.3.1.4 Bilinear Interpolation

Bilinear interpolation considers the closest neighborhood of three hexagonal pixels surrounding the output square pixel location. This neighborhood is obtained with Delaunay triangulation as illustrated in Figure 3.11. The value of the resulting square pixel is the sum of the three hexagonal neighbors weighted by their distance to the square one.

Pros and Cons. Again, the size of the resampled image is arbitrary. By choosing it carefully we can preserve memory consumption. However, as for the nearest interpolation method, we need to apply a normalization to preserve the overall pixel intensity of the image. As the rebinning method, bilinear interpolation smooths the image.

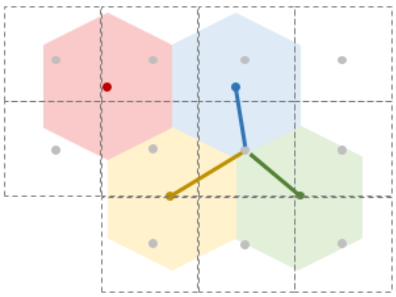


Figure 3.11 – Interpolation of hexagonal pixels to square ones with the bilinear interpolation. A Cartesian grid is applied to the hexagonal one. The colored dots represent the center of the hexagonal pixels, the grey ones the center of the resampled pixels. The value of the latter is the weighted sum of their three nearest neighbors in the hexagonal lattice.

3.3.1.5 Bicubic Interpolation

Bicubic interpolation is similar to bilinear interpolation, except it considers the closest neighborhood of twelve hexagonal pixels surrounding the output square pixel location, as illustrated in Figure 3.12.

Pros and Cons. The bicubic interpolation method has the same advantages and drawbacks as the bilinear interpolation. However, the smoothing effect is stronger because more pixels are involved.

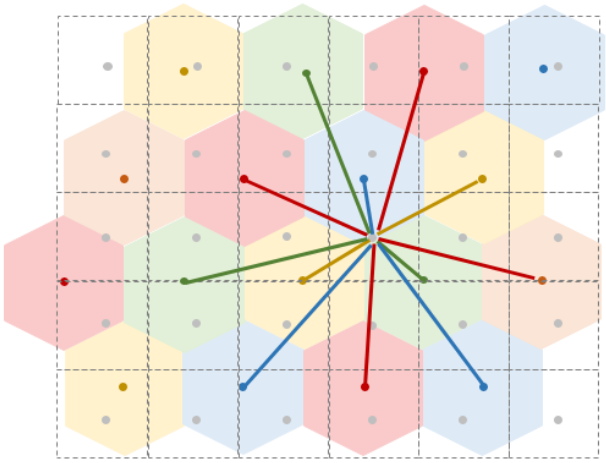


Figure 3.12 – Interpolation of hexagonal pixels to square ones with the bicubic interpolation. A Cartesian grid is applied to the hexagonal one. The colored dots represent the center of the hexagonal pixels, the grey ones the center of the resampled pixels. The value of the latter is the weighted sum of their 12 nearest neighbors in the hexagonal lattice.

3.3.1.6 Discussion

All the resampling methods presented allow using out-of-the-box deep learning frameworks with hexagonal pixel images. We can implement them as mapping operation, as in the DL1 Data Handler library [Kim et al., 2019]. The time needed for the resampling is then negligible compared to the network training duration. Resampling can be done while building the batches. Thus, we do not need to store resampled images on disk.

However, it is worth noticing that for camera shapes other than rectangular, these methods add fake pixels (generally set to zero) to fill the holes on the borders of the resampled image. This is the case for LST images that are almost hexagonal. Fake pixels represent then roughly 38 % of the resampled image, increasing dramatically the memory needed. Moreover, the oversampling method multiplies the needed space by n^2. Figure 3.13 illustrates the application of the five resampling methods described on LST images representing respectively the pixel indices in the hexagonal space. It exhibits the addition of fake pixels.

Figure 3.13 also highlights the smoothing effect of bilinear and bicubic interpolations and the rebinning method. The effect is particularly important for bicubic interpolation that involves twelve pixels from the input image for each resampled pixel. Smoothing is a desired property when resampling images for upscaling purpose. However, in the case of IACT data analysis, it might suppress useful details in the shape of the shower, as illustrated in Figure 3.14.

For the same reason that induces smoothing, these three resampling methods suffer from border effect. On the border of the hexagonal grid image, they take into account fake pixels to compute the resampled pixel.

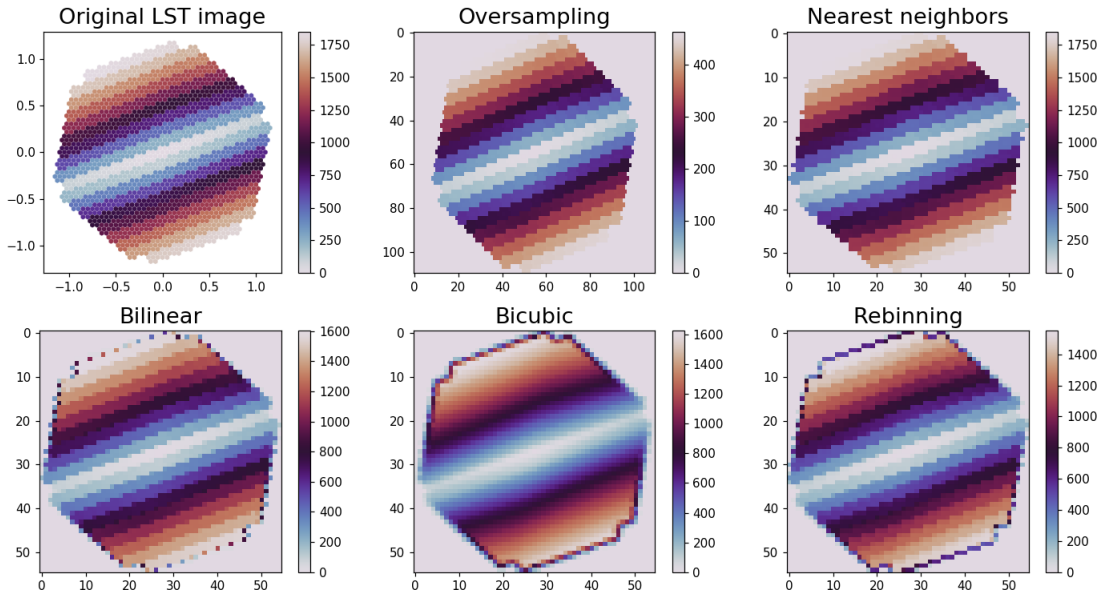


Figure 3.13 – Illustration of the five resampling methods described in Section 3.3.1. They are applied to an LST image representing the pixel indices in the hexagonal space.

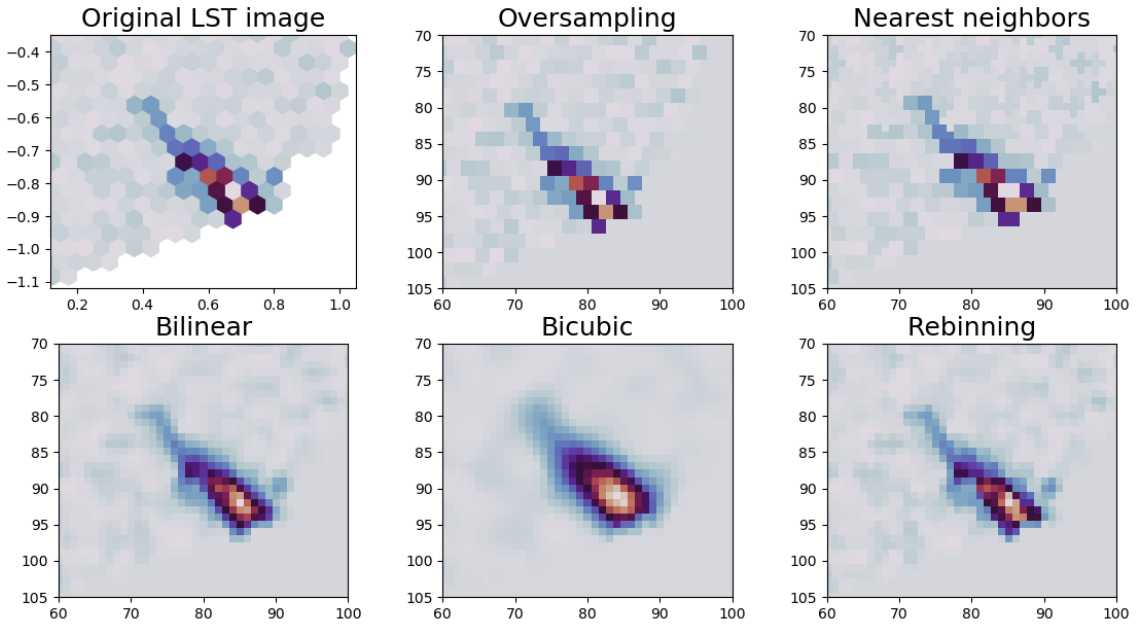


Figure 3.14 – Illustration of the five resampling methods described in Section 3.3.1. They are applied to an LST image representing a gamma event. The figure shows a zoom on the shower pixels.

Finally, resampling hexagonal grid images alters the pixel neighborhood. As discussed in Section 2.1.1, convolution is a crucial operator for deep learning performance. The convolution operation relies on pixel neighborhood to compute its output (see section 3.1.1 for more details). However, as illustrated in Figure 3.15, the natural neighborhood of hexagonal pixels cannot be preserved after resampling them to square pixels.

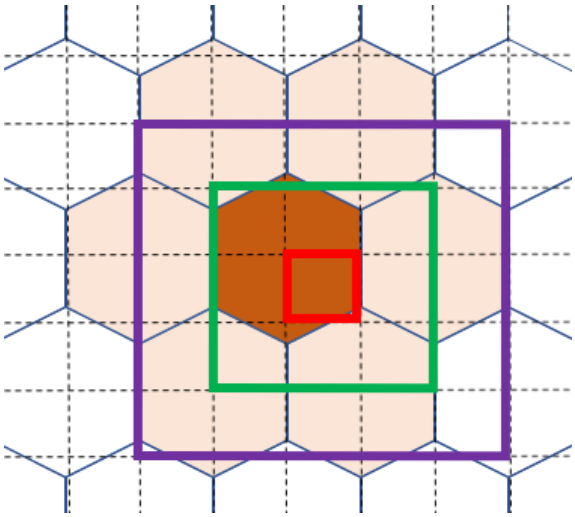


Figure 3.15 – Resampling hexagonal grid images to Cartesian grid ones alters the pixel neighborhood. Illustration with the oversampling method. The hexagonal neighborhood is composed of seven pixels (in brown and light brown), including the pixel of interest. After resampling to Cartesian grid, the pixel of interest (highlighted in red) is smaller and the standard 3×3 neighborhood (in green) only covers a fraction of the original hexagonal neighborhood. A wider neighborhood, for instance 5×5 (in purple), would include pixels that do not belong to the original neighborhood.

3.3.2 Hexagonal Grid Addressing and Custom Convolution

Another approach to apply deep learning technique to hexagonal grid images is to shift them into a Cartesian grid using a suitable pixel addressing system. It is then possible to design an adapted convolution operator depending on the chosen addressing system.

Several addressing systems exist to handle images with such lattice, among others: offset [Sousa, 2014], ASA [Rummelt, 2010], HIP [Middleton and Sivaswamy, 2001], axial — also named orthogonal or 2-axis obliques [Asharindavida et al., 2012, Sousa, 2014]. Both the offset and axial addressing systems seem to be appropriate for deep learning as they allow to store hexagonal grid images in a single matrix.

3.3.2.1 Offset Addressing and Hexagonal Convolution

The offset addressing system is used in combination with Hexagonal Convolution in [Shilon et al., 2019]. As illustrated in Figure 3.16, this addressing system consists in shifting one column over two by half a pixel space (the hexagonal pixel image is rotated by 90° compared to the examples presented in Section 3.3.1). The authors then introduce Hexagonal Convolution to apply convolution to the grid shifted while preserving the hexagonal neighborhood. The Hexagonal Convolution is composed of two convolution kernels. The first kernel of size 2×2 with dilation $(2, 1)$ and stride $(2, 1)$ is separately applied to even (in orange on the figure) and odd (in red) columns with corresponding padding. The resulting columns of these two operations are alternatively merged. The second kernel of size 1×3 is applied to the image shifted with suitable padding. The results of kernel 1 and kernel 2 operations are finally summed.

Pros and Cons. The Hexagonal Convolution combined with the offset addressing system allows applying deep learning to hexagonal images without resampling them. However, they still need to be stored into square arrays. In the case of cameras with non rectangular frame, fake pixels are added, as for resampling methods detailed in Section 3.3.1. Moreover, to respect the hexagonal neighborhood, the Hexagonal Convolution consists of three convolution operations, increasing the computation time.

3.3.2.2 Axial Addressing and Masked Convolution

The axial addressing system also has interesting properties. It is complete, unique, convertible to and from the Cartesian lattice and efficient [He and Jia, 2005]. As the offset addressing system, it offers a straightforward conversion from hexagonal to Cartesian grid, stretching the converted image, as shown in figure 3.17, but preserving the true neighborhood of the pixels. In [Hoogeboom et al., 2018] it is used in combination with masked convolution to respect the hexagonal kernel. In this paper, the authors present group convolutions for square pixels and hexagonal pixel images. A group convolution consists in applying several transformations (e.g. rotation) to the convolution kernel to benefit from the axis of symmetry of the images.

Pros and Cons. As for the Hexagonal Convolution, masked convolution combined with the axial addressing system allows applying deep learning to hexagonal images without resampling them. They also need to be stored into square arrays with fake pixels added in the case of cameras with non rectangular frame. Moreover, to respect the hexagonal

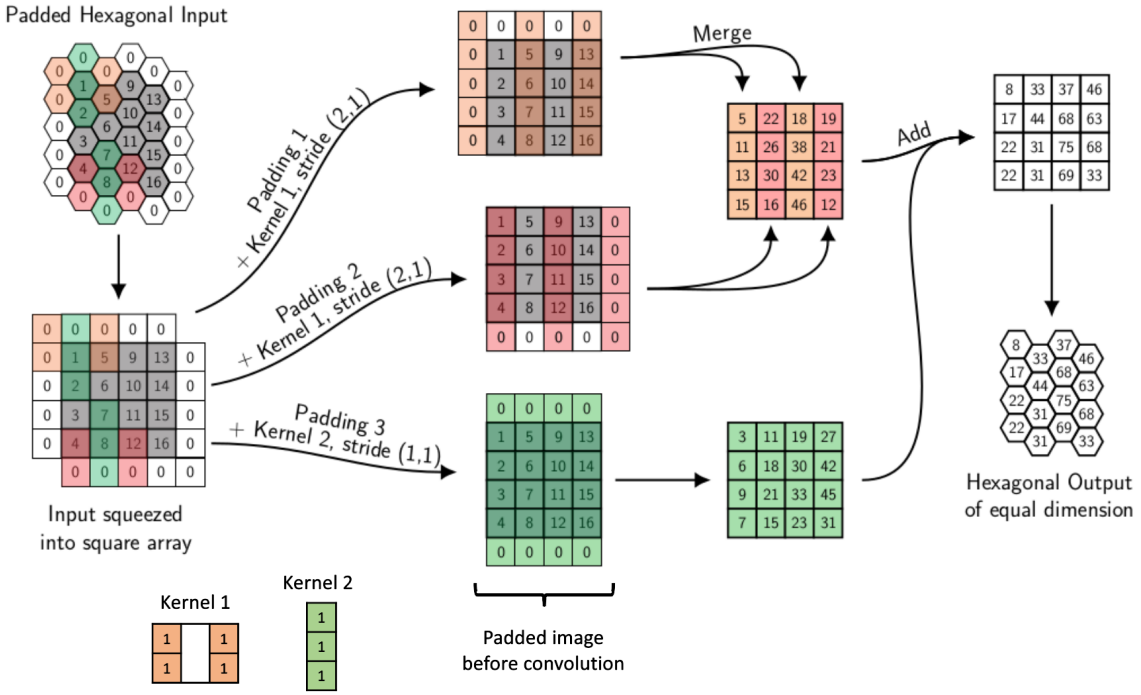


Figure 3.16 – Hexagonal Convolution in combination with the offset addressing system. Source: [Shilon et al., 2019], modified. In this example, the pixels of the input image have values ranging from 1 to 16. The are surrounded by padding pixels of value equal to 0. Relying on the offset addressing system, the input is squeezed into the Cartesian grid. For simplicity, the weights of both kernels are set to 1. The kernel 1 is separately applied to even (in orange) and odd (in red) columns with corresponding padding. The resulting columns of these two operations are alternatively merged. The kernel 2 is applied to the image shifted with suitable padding. The results of kernel 1 and kernel 2 operations are finally summed.

neighborhood, a mask is applied to the convolution kernel, slightly increasing the computation time.

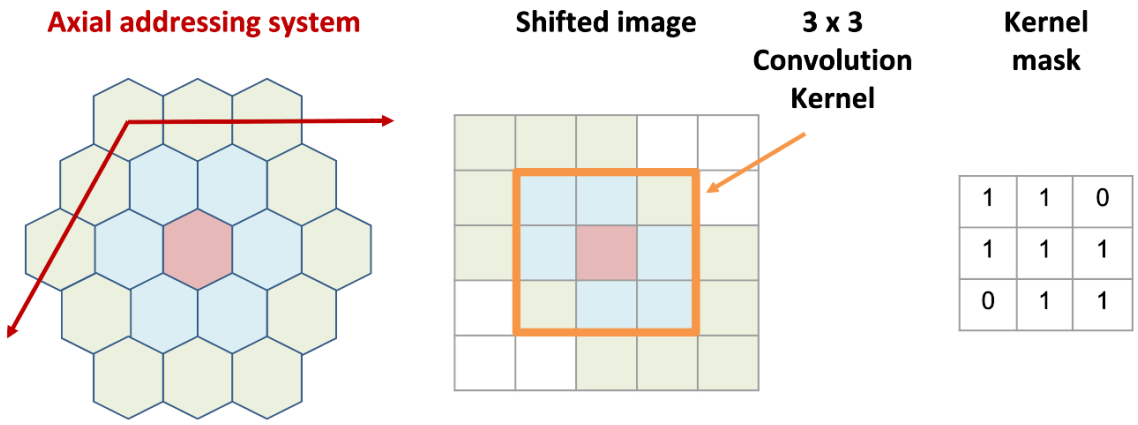


Figure 3.17 – Using the axial addressing system to apply convolution. The blue pixels represent the real neighborhood of the pixel of interest (in red). The hexagonal pixel image is shifted into the Cartesian grid relying on the axial addressing system. We can then apply the standard 2D convolution (for instance 3×3, as illustrated by the orange border) to the shifted image. To respect the hexagonal neighborhood, we need to apply a mask to the convolution kernel.

3.3.3 Graph Convolution

A different approach that spread during this thesis intends to apply deep learning to non-Euclidean space and generalize convolution to data represented by non-regular grids [Henaff et al., 2015]. For this type of data we can build a weighted undirected graph describing the relationship between the data points. Graphs are widely used to represent networks and interactions [Yanardag and Vishwanathan, 2015] or even protein structure [Fout et al., 2017]. A particular case of graph is mesh used to represent 3D shapes, as illustrated in Figure 3.18. The nodes of the mesh are then referred to as point cloud. Several methods have been proposed to generalize the convolution operator to graph structured data.

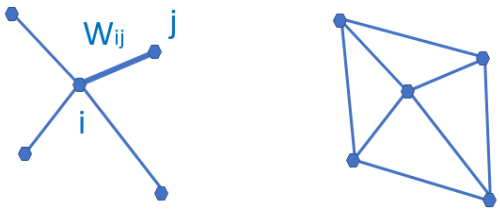


Figure 3.18 – Example of representation of irregular grid data. *On the left:* an undirected graph. *On the right:* a mesh.

Spectral filtering [Bronstein et al., 2017] consists in mapping the features to the spectral domain by projecting them on the eigenvectors of the graph Laplacian. Convolution amounts to scaling the spectral signal in the eigenbasis.

Local graph filtering [Monti et al., 2017] formulates convolution as template matching with local patches on the graph. Different methods exist to establish the correspondence between the convolution weights and the nodes belonging to the local neighborhood, mostly with hand-designed pseudo-coordinates. In [Verma et al., 2018], the authors propose to learn this mapping using the features of the previous layer in the CNN.

In the case of IACT hexagonal grid images the graph could be obtained by applying Delaunay triangulation to the pixel position. We could then use the methods presented to build our model.

Pros and Cons. At the time of writing this thesis, none of these methods is implemented as an optimized routine on GPU, increasing the computation time. Even so, a quickly growing Github repository[1] lists the implementation with the PyTorch framework of the graph convolution methods presented in the state-of-the-art literature.

3.3.4 Summary

In this section I have discussed different methods to apply deep learning technique to hexagonal grid images. The most widespread ones, especially in the IACT data analysis field, consist in resampling the data into Cartesian grid images. We can also apply a geometric transformation through a suitable pixel addressing system in combination with an adapted convolution operator, such as masked convolution or Hexagonal Convolution. A third possibility could consider hexagonal lattices as special cases of graph and apply one of the proposed graph convolution methods.

However, such approaches may have several drawbacks:

- resampling methods introduce distortions that can potentially result in lower accuracy or unexpected results;
- resampling and geometric transformation impose additional processing, often performed at the CPU level which slows inference in production;
- resampling and geometric transformation increase the memory consumption of images with non-rectangular shape;
- geometric transformation with masked convolution adds unnecessary computations as the mask has to be applied to the convolution kernel at each iteration;
- geometric transformation with Hexagonal Convolution multiply the computation time by three, as convolution is decomposed in three convolutions, a merging and an addition;
- resampling or geometric transformations can change the image shape and size;
- graph convolution adds unnecessary complexity as hexagonal grid is a regular one;
- graph convolution methods are not yet optimized on GPU.

[1] https://github.com/rusty1s/pytorch_geometric

In order to overcome these issues and be able to work on unaltered data, I present in the next section a way to apply convolution and pooling operators to any grid, given that each pixel neighbor is known and provided. This solution is denoted Indexed Convolution and Indexed Pooling in the following of this thesis.

3.4 Indexed Convolution

In the previous section I have reviewed several methods to apply deep learning technique to unconventional images. I have focused on the hexagonal grid since this is one of the most common lattices besides the Cartesian one. In particular, several telescopes of the CTA have a camera with this pixel organization, including the LST. A large part of this thesis is devoted to LST data analysis.

I introduce in this section a new solution, named Indexed Convolution and the related Pooling operator, based on the GEMM implementation of convolution. These original operators enable to apply the widespread deep learning frameworks to any kind of lattice. Although we first developed it to process the hexagonal pixel images of the CTA, these methods are very general solutions, easily applicable to other domains, including ones with irregular grid data. This work has been realized in collaboration with Orobix[2] within the GammaLearn project.

3.4.1 From GEMM to Indexed Convolution

Starting from the GEMM interpretation and implementation of convolution described in Section 3.1.3.1, we can extend convolution from rectangular lattices to any kind of pixel grid. The key step is the *im2col* operation.

Given an input vector of data and a matrix of indices describing every neighborhood relationships among the elements of the input vectors, *im2col* operation consists in picking elements from the input vector according to each neighborhood in the matrix of indices. The result is a column matrix containing, as for rectangular lattices, neighborhoods from different input channels concatenated along individual columns. At this point, convolution can be computed as a matrix multiplication.

The above procedure can be implemented in modern multidimensional array frameworks, as Numpy, TensorFlow and PyTorch, in a vectorized fashion with advanced indexing. It consists in indexing multidimensional arrays with other multidimensional arrays of integer values. The integer arrays provide the shape of the output and the indices at which the output values must be picked out of the input array.

Figure 3.19 illustrates the global process of the Indexed Convolution.

In the case of Indexed Convolution, the matrix of indices describing neighborhoods is already known and depends on the sensor itself. We can use this matrix to index into the input tensor, executing the *im2col* operation in one pass, both on CPU and GPU devices. Since the indexing operation is differentiable with respect to the input (but not with respect to the indices), a deep learning framework implementing automatic differentiation capabilities (like PyTorch or TensorFlow) can provide the backward pass automatically as needed.

[2] https://orobix.com/

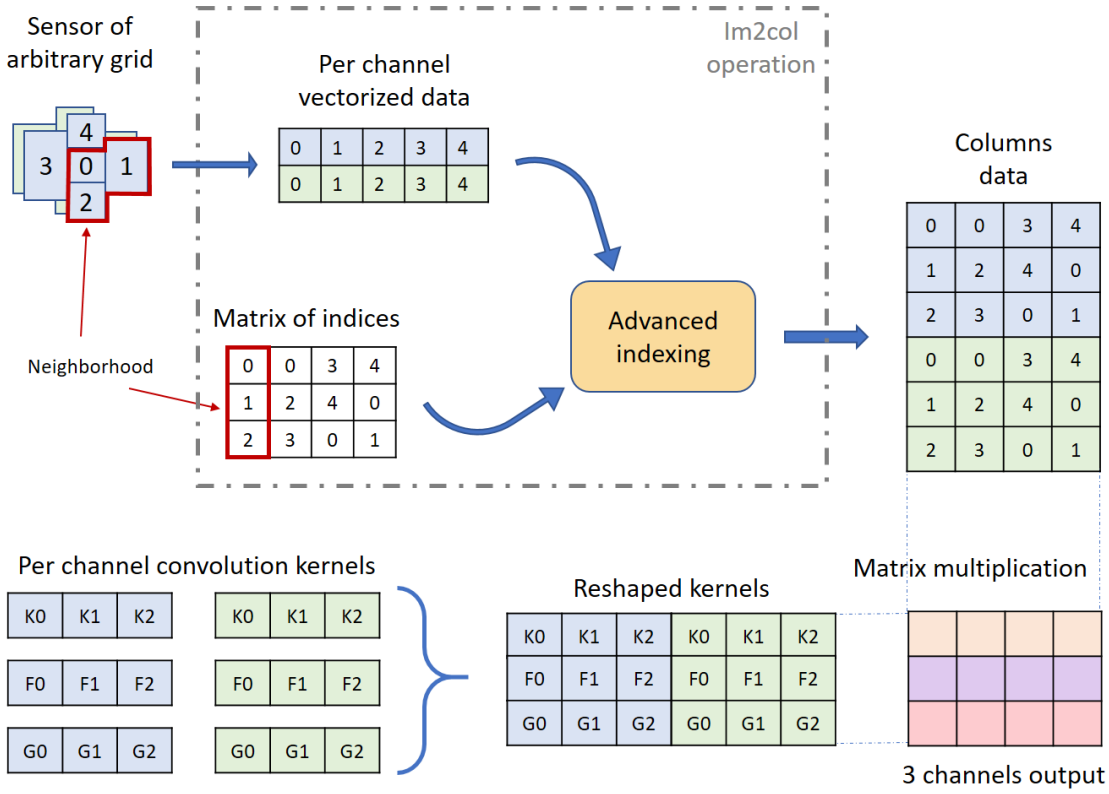


Figure 3.19 – Overview of the Indexed Convolution process. A sensor with an arbitrary grid produces the input data. They are composed of two channels (blue and green). For convenience, the value indicated in the pixels is their index in the grid. The matrix of indices describes the neighborhoods in the sensor grid as columns. The first step of the procedure consists in vectorizing the input data per channel. The *im2col* operation is finalized through advanced indexing the vectorized input with the matrix of indices. This results in the column data. Then, the convolution kernels are reshaped into a matrix so that the weights corresponding to the different output channels are arranged in rows. Finally, the matrix multiplication between the reshaped kernels and the column data yields the output of the convolution.

We now present a PyTorch implementation of such *Indexed Convolution* in the hypothetical case illustrated in Figure 3.19.

Data Description. We consider in the following example an input tensor with shape (B, C_{in}, W_{in}), where B is the batch size equal to 1, C_{in} is the number of channels equal to 2, or features, and W_{in} is the width equal to 5, *i.e.*, the number of elements per channel,

```
input = torch.ones(1, 2, 5)
```

and a specification of neighbors as a tensor of indices with shape (K, W_{out}), where K is the size of the convolution kernel equal to 3 and W_{out} equal to 4 is the number of elements per channel in the output

```
indices = torch.tensor([[ 0, 0, 3, 4],
                        [ 1, 2, 4, 0],
                        [ 2, 3, 0, 1]])
```

where values, arbitrarily chosen in this example, represent the indices of 4 neighborhoods of size 3 (neighborhoods are laid out along columns). The number of columns corresponds to the number of neighborhoods, *i.e.*, dot products, that will be computed during the matrix multiply, hence they correspond to the size of the output per channel.

Convolution Description. The weight tensor describing the convolution kernels has a shape of (C_{out}, C_{in}, K), where C_{out} equal to 3 is the number of channels, or features, in the output. The bias is a column vector of size C_{out}.

```
weight = torch.ones(3, 2, 3)
bias = torch.zeros(3)
```

im2col Operation. At this point we can proceed to use advanced indexing to perform the *im2col* operation and build the column matrix according to indices.

```
col = input[..., indices]
```

Here we are indexing a (B, C_{in}, W_{in}) tensor with a (K, W_{out}) tensor, but the indexing operation has to preserve batch and input channels dimensions. To this end, we employ the Python ellipsis notation [...], which prescribes indexing to be replicated over all dimensions except the last. This operation produces a tensor shaped (B, C_{in}, K, W_{out}), *i.e.*, $(1, 2, 3, 4)$.

As noted above, the column matrix needs values from neighborhoods for all input channels concatenated along individual columns. This is achieved by reshaping the *col* tensor so that C_{in} and K dimensions are concatenated:

```
B = input.shape[0]
W_out = indices.shape[1]
col = col.view(B, -1, W_out)
```

The columns in the *col* tensor are now a concatenation of 3 values (the size of the kernel) per input channel, resulting in a $(B, K \times C_{in}, W_{out})$ tensor. Note that the *col* tensor is still organized in batches.

Reshaping the Convolution Kernels. At this point, kernel weights must be arranged so that weights corresponding to different output channels are concatenated along columns as well:

```
C_out = weight.shape[0]
weight_col = weight.view(C_out, -1)
```

which leads from a (C_{out}, C_{in}, K) to a $(C_{out}, K \times C_{in})$ tensor.

Matrix Multiplication. We need then to perform batch-wise matrix multiplication between the *weight_col* and *col* matrices to realize the convolution. The *weight_col* is expanded along the batch dimension to compute in a parallel way the matrix multiplication for each element of the batch:

```
out = torch.bmm(weight_col.expand(nbatch, -1, -1), col)
```

to obtain a B, C_{out}, W_{out} tensor.

In case *bias* is used in the convolution, it must be added to each element of the output, *i.e.*, a constant is summed to all values per output channel. In this case, *bias* is a tensor of shape C_{out}, so we can perform the operation by relying on broadcasting on the first B and last W_{out} dimension:

```
out += bias.unsqueeze(1)
```

Padding the Input. Padding can be handled by prescribing a placeholder value, e.g. -1, in the matrix of indices. The following instruction shows an example of such a strategy:

```
indices = torch.tensor([[-1, 0, 3, 4],
                        [ 1, 2, 4, 0],
                        [ 2, 3, 0, 1]])
```

The location can be used to set the corresponding input to the zero padded value, though multiplication of the input by a binary mask. Once the mask has been computed, the placeholder can safely be replaced with a valid index so that advanced indexing succeeds.

```
indices = indices.clone()
padded = indices == -1
indices[padded] = 0

mask = torch.tensor([1.0, 0.0])
mask = mask[..., padded.long()]
col = input[..., indices] * mask
```

PyTorch Implementation. Indexed Convolution implementation in PyTorch is available as an open source package, IndexedConv [Jacquemont and Vuillaume, 2019].

3.4.2 Extension to Pooling

3.4.2.1 Pooling Operation

In deep neural networks, convolutions are often associated with pooling layers. They allow feature maps down sampling thus reducing the number of network parameters and so the time of the computation. In addition, pooling improves feature detection robustness by achieving (local) spatial invariance [Scherer et al., 2010].

The pooling operation can be defined as:

$$O_i = f(I_{N_i}) \tag{3.7}$$

where O_i is the output pixel i, f a function, I_{N_i} the neighborhood of the input pixel i of a given input feature map I. The pooling function f provided on Equation 3.7 is applied to I_{N_i} using a sliding window. f can be of various forms, for example an average, a softmax, a convolution or a max. The use of a stride greater than two on the sliding window translation enables to subsample the data, as illustrated in Figure 3.20. With convolutional networks, a max-pooling layer with stride two and width two is typically considered moving to a two times coarser feature maps scale after having applied some standard convolution layers. This proved to reduce network overfitting while improving task accuracy [Krizhevsky et al., 2012].

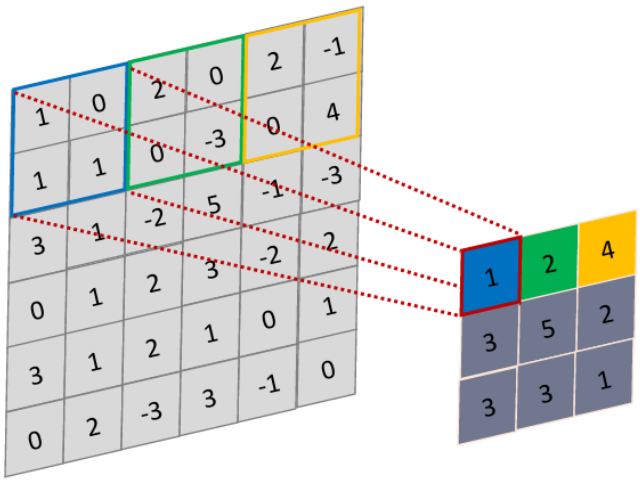


Figure 3.20 – Example of the pooling operation with the maximum function. The input data has shape 6×6. The pooling kernel is of size 2×2, with a stride equal to 2. At every location the kernel slides over, the maximum pixel value is kept to build the output. The stride of two leads to a subsampling of the data by factor of two. The output shape is 3×3.

3.4.2.2 Indexed Pooling

Following the same procedure as for convolution described in Section 3.4.1, we can use the matrix of indices to produce the column matrix of the input and apply, in one pass, the pooling function to each column.

For instance, a PyTorch implementation of the indexed pooling, in the same hypothetical case as presented in Section 3.4.1, with *max* as the pooling function is:

```
col = input[..., indices]
out = torch.max(col, 2)
```

3.5 Application Example: The Hexagonal Case

The Indexed Convolution and Pooling can be applied to any pixel organization, as soon as one provides the list of the neighbors of each pixel. Although the method is generic, we first developed it to be able to apply deep learning technique to the hexagonal grid images of the Cherenkov Telescope Array. This section proposes a method to efficiently handle hexagonal data without any preprocessing as a demonstration of the use of Indexed Convolutions. I first describe how to build the index matrix for hexagonal lattice images needed by the Indexed Convolution.

For easy comparison, we want to validate our method on datasets with well-known use cases (e.g. a classification task) and performances. To our knowledge, there is no reference hexagonal image dataset for deep learning. So, following HexaConv paper [Hoogeboom et al., 2018] I constructed two datasets with hexagonal images based on well-established square pixel image datasets dedicated to classification tasks: CIFAR-10 and AID. This enables our method to be compared with classical square pixels processing in a standardized way.

We also want to compare our method with the standard resampling techniques described in Section 3.3.1 on CTA data. These techniques are widely used to apply deep learning to IACT data, in the H.E.S.S. experiment [Holch et al., 2017, Shilon et al., 2019] as well as in the CTA experiment [Nieto et al., 2017, Mangano et al., 2018]. Comparing on CTA data is essential for this thesis as it focuses on CTA data analysis.

3.5.1 Indexing the Hexagonal Lattice and the Neighbors' Matrix

As described in Section 3.4.1, in addition to the image itself, we need to feed the Indexed Convolution (or Pooling) with the list of the neighbors considered for each pixel of interest, the matrix of indices. In the case of images with a hexagonal grid, provided a given pixel addressing system, a simple method to retrieve these neighbors is proposed.

Following the discussion of Section 3.3.2, our method relies on the axial addressing system to build an index matrix of hexagonal grid images. Assuming that a hexagonal image is stored as a vector and that we have the indices of the pixels of the vector images represented in the hexagonal grid, we can convert it to an index matrix thanks to the axial addressing system. Then, building the list of neighbors, the matrix of indices, consists in applying the desired kernel represented in the axial addressing system to the index matrix for each pixel of interest.

An example is proposed on Figure 3.21, with the kernel of the nearest neighbors in the hexagonal lattice. Regarding the implementation, we have to define in advance the kernel to use as a mask to be applied to the index matrix, for the latter example:

```
kernel = [[1, 1, 0],
          [1, 1, 1],
          [0, 1, 1]]
```

A more general procedure can be defined for hexagonal kernels. The size of the kernel is defined as:

$$k = 2 \times radius \times dilation + 1 \tag{3.8}$$

where the radius defines the perimeter of the neighborhood to consider (*e.g.*, one amounts to the nearest neighbors, two to the nearest and the second neighbors). The dilation pa-

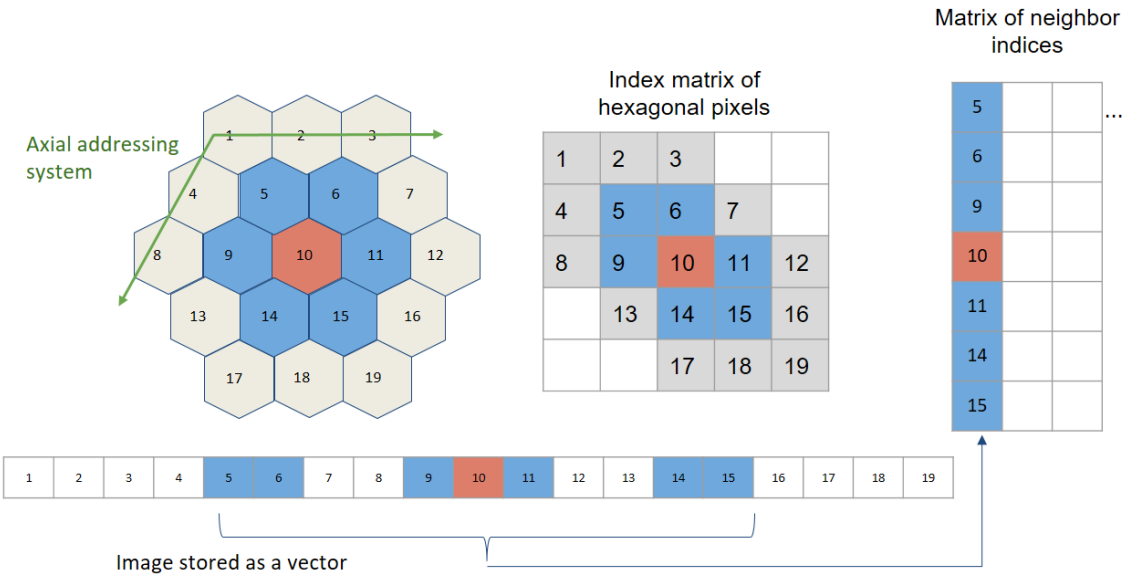


Figure 3.21 – Building the matrix of indices for an image with a hexagonal grid. The image is stored as a vector, and the indices of the vector are represented in the hexagonal lattice. Thanks to the axial addressing system, this representation is converted to a rectangular matrix, the index matrix. The neighbors of each pixel of interest (in red) are retrieved by applying the desired kernel (here the nearest neighbors in the hexagonal lattice, in blue) to the index matrix.

rameter is related to *atrous* convolutions (see section 3.1.1 for more details). We can described the kernel with:

$$kernel(i,j) = \begin{cases} 1, & \text{if } i-j <= radius \times dilation \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

with $i = 0 \ldots k-1$ and $j = 0 \ldots k-1$.

This procedure is implemented in the IndexedConv package.

3.5.2 Experiment on CIFAR-10

The Indexed Convolution method, in the special case of hexagonal grid images, is validated on the CIFAR-10 dataset. For this experiment and the one on the AID dataset (see Section 3.5.3), I compare our results with the two baseline networks of HexaConv paper [Hoogeboom et al., 2018]. These networks do not include group convolutions and are trained respectively on square and hexagonal grid image versions of CIFAR-10. The network trained on the hexagonal grid CIFAR-10 consists of masked convolutions. To allow a fair comparison, I use as far as possible the same experimental conditions.

The CIFAR-10 dataset is composed of 60,000 tiny color images of size 32x32 with square pixels. Each image is associated with the class of its foreground object. This is one of the reference databases for image classification tasks in the machine learning community. By converting this square pixel database into its hexagonal pixel counterpart, this enables to compare hexagonal and square pixel processing in different case studies for image classification. This way, the same network with:

- Standard convolutions (square kernels),

- Indexed Convolutions (square kernels),
- Indexed Convolutions (hexagonal kernels),

are trained and tested, respectively on the dataset for the square kernels and its hexagonal version for the hexagonal kernels. For reproducibility, the experiment is repeated 10 times with different weight initialization, but using the same random seeds (*i.e.*, same weight initialization values) for all three implementations of the network.

3.5.2.1 Building a Hexagonal CIFAR-10 Dataset

The first step is to transform the dataset in a hexagonal one. Compared to a rectangular grid, a hexagonal grid has one line out of two shifted of half a pixel (see figure 3.22). Square pixels (orange grid) cannot be rearranged directly in a hexagonal grid (blue grid). For these shifted lines, pixels have to be interpolated from the integer position pixels of the rectangular grid. The interpolation chosen here is the average of the two consecutive horizontal pixels. A fancier method could have been to take into account all the six square pixels contributing to the hexagonal one, in proportion to their surface involved. In that case, both the pixels retained for our interpolation method would cover 90.4% of the surface of the interpolated hexagonal pixel. It justifies neglecting the surfaces not taken into account. However, it is worth noticing that the input hexagonal data provided to our model is slightly degraded compared to the one that feed classical regular grid based models.

Figure 3.23 shows a conversion example, we can observe that the interpolation method is rough as we can see on the back legs of the horse so that hexagonal processing experiments suffer from some input image distortion. However, my preliminary experiments did not show strong classification accuracy difference between such conversion and a higher quality one.

Then the images are stored as vectors and the index matrix based on the axial addressing system is built. Before feeding the network, the images are standardized and whitened using a PCA, following [Hoogeboom et al., 2018].

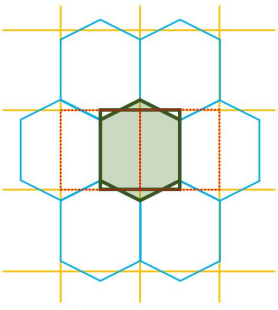


Figure 3.22 – Resampling of a rectangular grid (orange) image to a hexagonal grid one (blue). One line of two in the hexagonal lattice is shifted by half a pixel compared to the corresponding line in the square lattice. The interpolated hexagonal pixel (with a green background) is the average of the two corresponding square pixels (with red dashed borders).

Figure 3.23 – Example of an image from the CIFAR-10 dataset resampled to hexagonal grid.

3.5.2.2 Network

The network used for this experiment is described in Section 5.1 of [Hoogeboom et al., 2018] and relies on a ResNet architecture [He et al., 2015]. As shown in figure 3.24, it consists of a convolution, three stages with four residual blocks each, a pooling layer and a final convolution. Down sampling between two stages is achieved by a convolution of kernel size 1×1 and stride two. After the last stage, feature maps are squeezed to a single pixel (1×1 feature maps) by the use of an average pooling over the whole feature maps. Then a final 1×1 convolution (equivalent to a fully connected layer) is applied to obtain the class scores. I have implemented three networks in PyTorch, one with built-in convolutions (square kernels) and two with Indexed Convolutions (one with square kernels and one with hexagonal kernels). Rectangular grid image versions have convolution kernels of size 3×3 (9 pixels) while the one for hexagonal grid images has hexagonal convolution kernels of the nearest neighbors (7 pixels). The number of features per layer is set differently, as shown in Table 3.1, depending on the network so that the total number of parameters of all three networks are close, ensuring the comparison to be fair. I train these networks with the same hyperparameters detailed in Appendix B, Table B.1.

Table 3.1 – Number of features for all three hexagonal and square networks used on CIFAR-10.

	conv1	stage 1	stage 2	stage 3
Hexagonal kernels	17	17	35	69
Square kernels	15	15	31	61

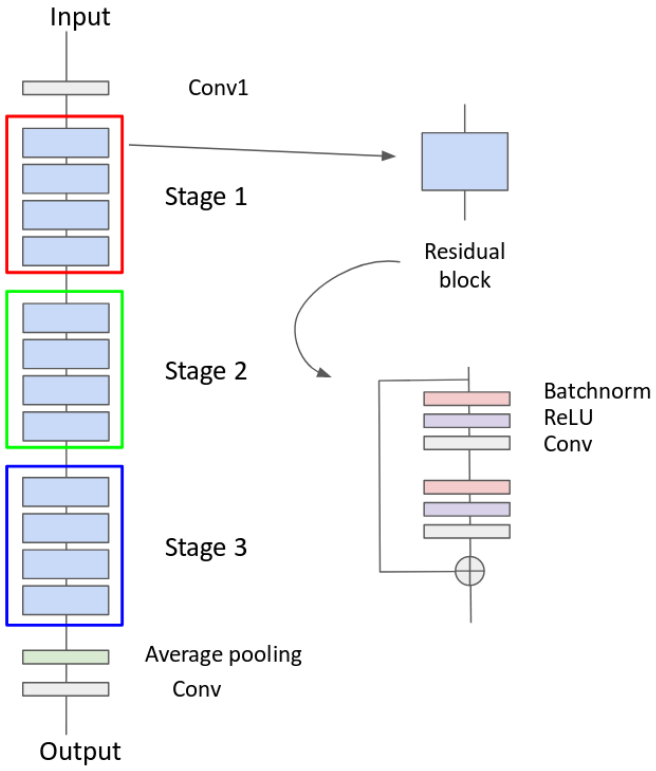


Figure 3.24 – ResNet model used for the experiment on CIFAR-10.

3.5.2.3 Results

As shown in Table 3.2, all three networks with hexagonal Indexed Convolutions, square Indexed Convolutions and square standard convolutions exhibit similar performances on the CIFAR-10 dataset. The difference between the hexagonal kernel and the square kernel with standard convolution on the one hand, and between both square kernel is not significant, according to the Student T test. For the same number of parameters, the hexagonal kernel model gives slightly better accuracy than the square kernel one in the context of Indexed Convolution, even if the images have been roughly interpolated for hexagonal image processing. However, to satisfy this equivalence in the number of parameters, since hexagonal convolutions involve fewer neighbors than the squared counterpart, some more neurons are added all along the network architecture. This leads to a larger number of data representations that are combined to achieve the task. The hexagonal convolution seems to provide richer features for the same price in the parameters count. This may also compensate for the image distortions introduced when converting input images to hexagonal sampling. Such distortions actually sat hexagonal processing in an unfavorable initial state but the hexagonal processing compensated and slightly outperformed the standard approach.

[Hoogeboom et al., 2018] carried out a similar experiment and observed the same accuracy difference between hexagonal and square convolutions processing despite a shift in the absolute accuracy values (88.75 for hexagonal images, 88.50 for square ones). This can be explained by different image interpolation methods, different weight initialization and the use of different frameworks.

Table 3.2 – Accuracy results for all three hexagonal and square networks on CIFAR-10. *i. c.* stands for Indexed Convolutions.

Hexagonal kernels (i.c.)	Square kernels (i.c.)	Square kernels
88.51 ± 0.21	88.27 ± 0.23	88.39 ± 0.48

3.5.3 Experiment on AID

Similar to the experiment on CIFAR-10, the Indexed Convolution is validated on Aerial Images Dataset (AID) [Xia et al., 2016]. The AID dataset consists of 10,000 RGB images of size 600x600 within 30 classes of aerial scene type. Similar to section 3.5.2, the same network with standard convolutions (square kernels) and then with Indexed Convolutions (square kernels and hexagonal kernels) is trained and tested, respectively on the dataset for the square kernels and its hexagonal version for the hexagonal kernels. The experiment is also repeated ten times, but with the same network initialization and different random split between training set and validating set, following [Hoogeboom et al., 2018].

3.5.3.1 Building a Hexagonal AID Dataset

After resizing the images to 64x64 pixels, the dataset is transformed to a hexagonal one, as shown Figure 3.25, in the same way as in Section 3.5.2.1. Then the images are standardized.

Figure 3.25 – Example of an image from the AID dataset resized to 64x64 pixels and resampled to hexagonal grid.

3.5.3.2 Network

The network considered in this experiment is still a ResNet architecture but adapted to this specific dataset. I follow the setup proposed in Section 5.2 of [Hoogeboom et al., 2018]. Three networks have been implemented and trained in the same way described in Section 3.5.2.2, with the number of features per layer described in Table 3.3.

Table 3.3 – Number of features for all three hexagonal and square networks used on AID.

	conv1	stage 1	stage 2	stage 3
Hexagonal kernels	42	42	83	166
Square kernels	37	37	74	146

3.5.3.3 Results

As shown in Table 3.4, all three networks with hexagonal convolutions and square convolutions do not exhibit a significant difference in performances on the AID dataset. Again, no accuracy loss is observed in the hexagonal processing case study despite the rough image resampling.

However, unlike on the CIFAR-10 experiment, I do not observe a better accuracy of the model with hexagonal kernels, as emphasized in [Hoogeboom et al., 2018]. Still, results are very similar while the hexagonal input has been degraded in the data preparation process.

Table 3.4 – Accuracy results for all three hexagonal and square networks on AID. *i. c.* stands for Indexed Convolutions.

Hexagonal kernels (i.c.)	Square kernels (i.c.)	Square kernels
79.81 ± 0.73	79.88 ± 0.82	79.85 ± 0.50

3.5.4 Experiment on CTA Data

In collaboration with other CTA members (D. Nieto, A. Brill, Q. Feng, B. Kim and T. Miener), we also compare the performance on CTA images classification between Indexed Convolutions with hexagonal kernels and the mapping methods described in Section 3.3.1, namely oversampling, rebinning, nearest neighbor, bilinear and bicubic interpolations. The classification task is solved in the context of single-telescope analysis, using only the pixel charge information. This work has been published as a conference paper [Nieto et al., 2019b] at the 36^{th} International Cosmic Ray Conference (2019).

3.5.4.1 CTA Prod3b Dataset

The dataset used for this experiment is different from the one used for the rest of this thesis and described in Section 1.3.4. It is also made of Monte Carlo simulated events for the CTA, but from a previous production. The third large-scale Monte Carlo production [Bernlöhr et al., 2013] (Prod3b) was originally realized to help define the layouts of both North and South site of the CTA. Simulated events have been reduced, from raw data sequences to calibrated and integrated images of pixel charges, on the EGI[3] by means of

[3] www.egi.eu

the DL1 Data Writer in DL1DH [Kim et al., 2019].

For this experiment, we only keep data related to the final layout (S8) [Acharyya et al., 2019] of the Southern installation. It is composed of four large-size telescopes (LSTs), 25 medium-size telescopes (MSTs), and 70 small-size telescopes (SSTs). Three models of MST and three models of SST have been originally simulated, but we restrict ourselves to the single-mirror MST with FlashCam as its camera (MST-F) and the single-mirror SST-1M respectively.

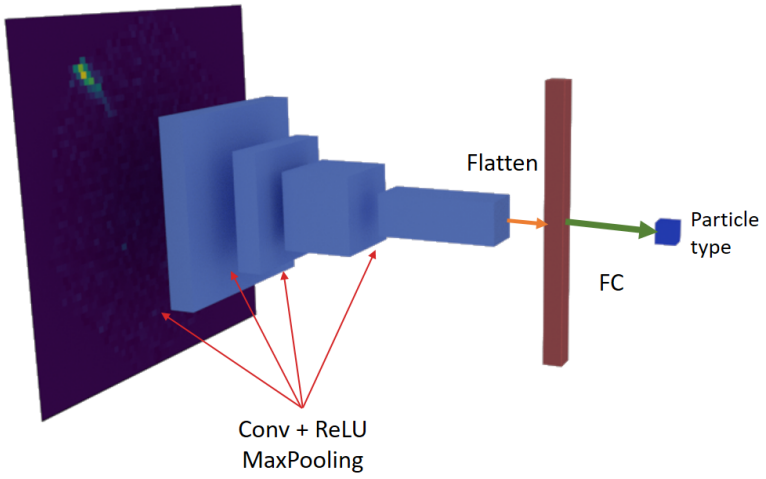
The particle showers have been simulated with a Zenith angle of 20° and an Azimuth angle of 0° (North pointing). The selected dataset contains both diffuse gamma-ray and proton showers with balanced statistics in number of events, accounting for nearly 400 thousand events (1.4 million images, since most events trigger more than one telescope). The selected events are randomly drawn from the source dataset and then split following an 8/2 ratio into a train/validation dataset and a test dataset.

3.5.4.2 Network

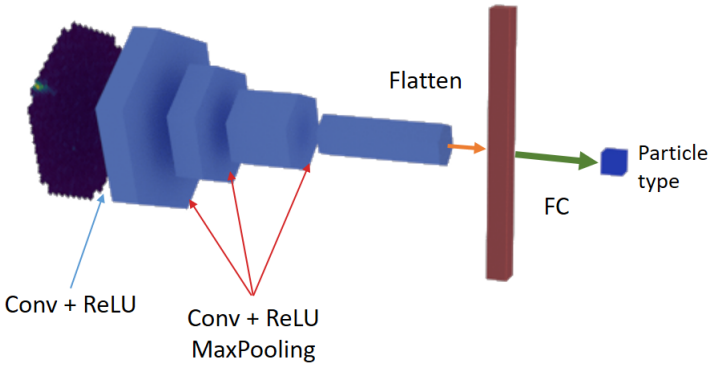
The network chosen for this experiment is a simple CNN with proven classification capabilities. This is the model denoted *single-tel* in [Nieto et al., 2019a]. It is used unaltered when trained on the resampled images and slightly modified when trained on original CTA images following the Indexed Convolution strategy, as illustrated in Figure 3.26. The *single-tel* model is composed of four convolutional layers with 32, 32, 64, and 128 filters and a kernel size of 3×3 in each layer, with ReLU activation followed by a max-pooling layer with a kernel size (and stride) of 2. The output of the last convolutional layer is flattened and fed to a fully connected layer with an output size of 2, the number of classes.

The Indexed Convolution version of the model has hexagonal convolution kernels of size 7, corresponding to the first neighbors of the pixel to process. To fairly compare the presented mapping methods with unmapped (*i.e.*, hexagonal) images, the first pooling layer is removed. Indeed, for this experiment, all the resampling methods output images of the same resolution as oversampling, increasing the size of the images. The idea behind this adaptation is to have roughly the same number of pixels of interest (*i.e.*, without taking into account the artificial pixels added by the mapping methods) in the feature maps. Thus, except for the first convolution layer, we apply convolution at the same level of fineness with respect to the original pixel size.

No dropout or batch normalization is set for any of the models. We set the loss function to categorical cross-entropy. These models are trained on 90% of the training set (the rest has been used for validation) with the same parameters detailed in Appendix B, Table B.4. We train on each type of telescope independently. Image resampling methods are explored with CTLearn framework [Brill et al., 2019] with the help of DL1 Data Handler [Kim et al., 2019] for preprocessing, while the experiments with Indexed Convolution are conducted with GammaLearn. For reproducibility, we repeat the experiments 10 times for every resampling method and 5 times for Indexed Convolution with different weight initialization, but using the same random seeds (*i.e.*, same weight initialization values) for all resampling methods and Indexed Convolution.



(a) Single-tel network for resampled images.



(b) Indexed Convolution version of single-tel model for hexagonal pixel images.

Figure 3.26 – Comparison between single-tel models used for resampled images (a) and hexagonal pixel images (b). As the size of the resampled images is increased compared to hexagonal pixel images, the first pooling is omitted in the Indexed Convolution version of the network. The blue blocks represent the feature maps produced by the convolution layers.

3.5.4.3 Results

In Figure 3.27 are presented the evolution of the AUC during training for the three types of telescopes and all the tested methods. The final results are summarized in Table 3.5 and Figure 3.28. The values for accuracy and AUC from the validation and test datasets are comparable for all methods and telescope types, almost always within the standard deviation that ranges from 0.2% to 0.5%. However, nearest neighbor interpolation seems to consistently underperform, while Indexed Convolution and bilinear interpolation are consistently superior, although not significantly better than oversampling, rebinning and bicubic interpolation. The difference in performance between the probed methods stands within standard deviations.

These results need to be confirmed, as this experiment only focus on the classification task of IACT data analysis. Moreover, the network used is a very simple CNN. However, as discussed in Section 2.1.1, the advent of very deep architectures pushed the limit of classification performance up to 84.4% top 1 accuracy on Imagenet with EfficientNet-B7 [Tan and Le, 2019]. In the section 3.5.5 I will extend this experiment on the energy and arrival direction reconstruction tasks, for Indexed Convolution and bilinear interpolation, with the popular ResNet [He et al., 2016b].

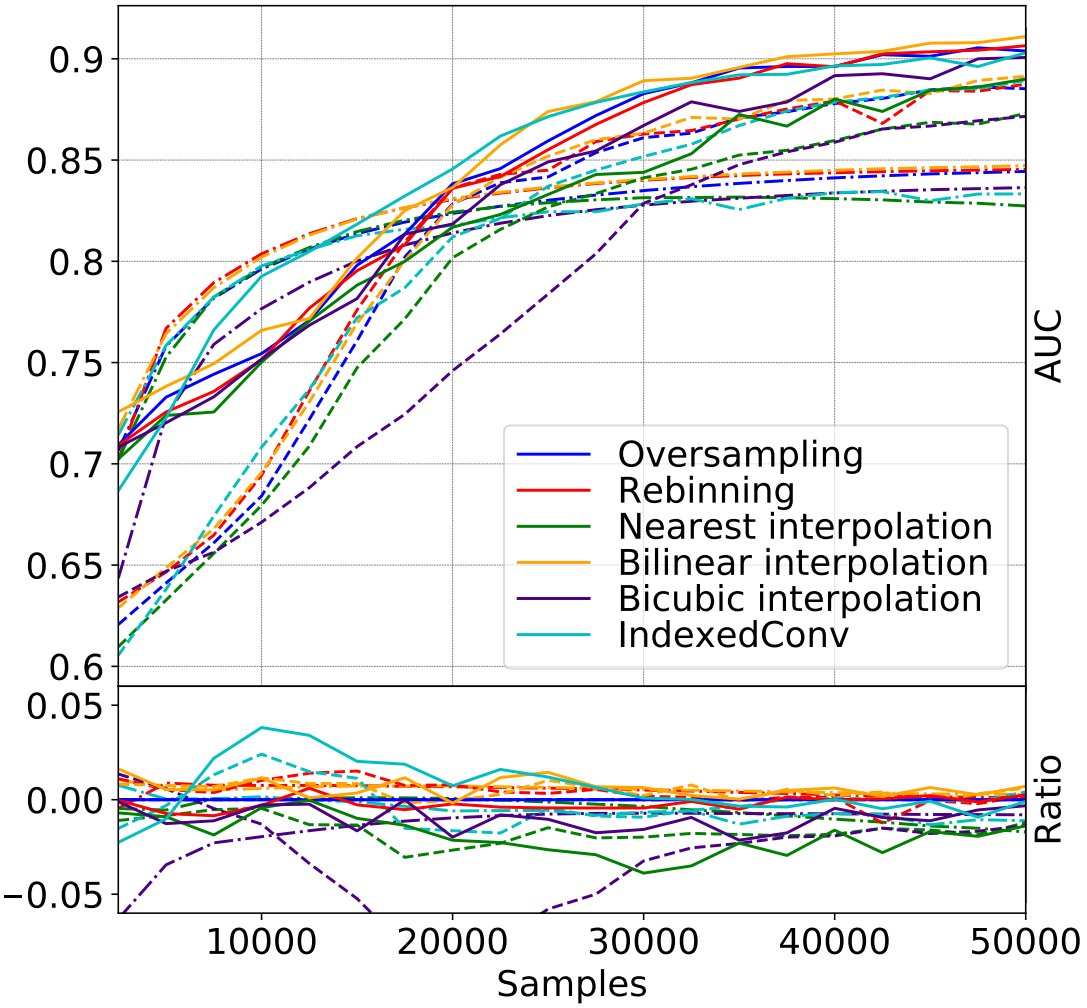


Figure 3.27 – Example of the evolution of the AUC during a given training run for the LST (dash dotted), the MST-F (dashed) and the SST-1M (solid). The lower plot is the ratio to oversampling method shown for comparison purposes. Source [Nieto et al., 2019b].

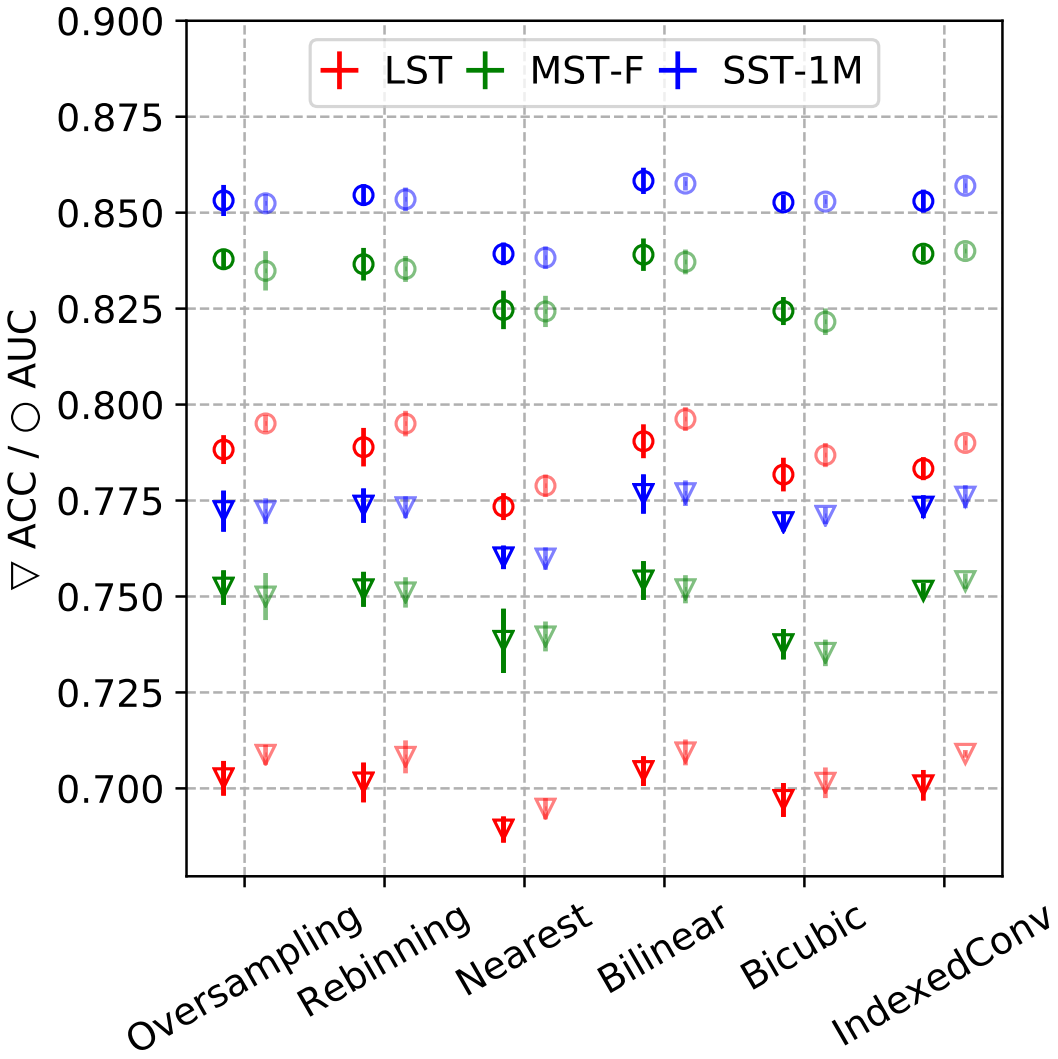


Figure 3.28 – Average and standard deviation of the learning metrics for all runs. Bright and pale markers depict train and test sets, respectively. Source [Nieto et al., 2019b].

LST		Oversampling	Rebinning	Nearest interp.	Bilinear interp.	Bicubic interp.	Indexed conv.
Validation	ACC	0.703±0.004	0.702±0.005	0.689±0.003	**0.704±0.004**	0.697±0.004	0.703±0.004
	AUC	0.788±0.004	0.789±0.005	0.773±0.004	**0.790±0.004**	0.782±0.004	0.786±0.003
Test	ACC	**0.709±0.003**	0.708±0.004	0.695±0.003	**0.709±0.003**	0.701±0.004	**0.709±0.001**
	AUC	0.795±0.002	0.795±0.003	0.779±0.003	**0.796±0.003**	0.787±0.003	0.790±0.002
MST-F		Oversampling	Rebinning	Nearest interp.	Bilinear interp.	Bicubic interp.	Indexed conv.
Validation	ACC	0.752±0.004	0.752±0.004	0.738±0.005	**0.754±0.004**	0.738±0.004	**0.754±0.002**
	AUC	0.838±0.003	0.836±0.004	0.825±0.005	0.839±0.004	0.824±0.004	**0.840±0.002**
Test	ACC	0.750±0.006	0.751±0.004	0.740±0.004	0.752±0.004	0.735±0.003	**0.754±0.002**
	AUC	0.835± 0.005	0.835±0.003	0.824 ± 0.004	0.837±0.003	0.822±0.003	**0.840±0.002**
SST-1M		Oversampling	Rebinning	Nearest interp.	Bilinear interp.	Bicubic interp.	Indexed conv.
Validation	ACC	0.772±0.005	0.774±0.004	0.760±0.003	**0.777±0.005**	0.769±0.002	**0.777±0.003**
	AUC	0.853±0.004	0.854±0.003	0.839±0.003	**0.858±0.003**	0.853± 0.002	0.857±0.003
Test	ACC	0.772± 0.003	0.773±0.003	0.760±0.003	**0.777±0.003**	0.771±0.002	0.776±0.003
	AUC	0.852±0.002	0.853±0.003	0.838±0.003	**0.858±0.002**	0.853±0.002	0.857±0.002

Table 3.5 – Average and standard deviation of the learning metrics obtained from all the training runs, for both the validation and the test set. Source [Nieto et al., 2019b]

3.5.5 Extending the Experiment on CTA Data

Following the results presented in Section 3.5.4.1, I take the comparison further between resampling methods and Indexed Convolution. We will see in Section 4.2 that a very deep network, the ResNet-56 [He et al., 2016b], has much better results on the classification task than the simple convolutional network used in Section 3.5.4.1. As detailed in Section 1.5, the data available to us contain time information about the development of the showers. This supplementary information is crucial to obtain the best possible results, in particular with single-telescope analysis. Finally, the complete analysis of IACT data consists in particle classification, *i.e.*, separating gamma events from the background, as well as reconstructing its energy and arrival direction.

To compare resampling methods and the Indexed Convolution strategy in a more realistic context than in Section 3.5.4.1, I train a deeper network to perform event classification, energy and arrival direction reconstruction (one network per task), using as input data the pixel charge and the time information. Unlike the previous comparison, I carry out this experiment only for the LST to benefit from the experiments involving the Indexed Convolution strategy described in Section 4.2 and previously realized. As I have limited GPU resources, I compare only with the best resampling method from Section 3.5.4.1, the bilinear interpolation.

For reproducibility, I repeat the experiments five times with different weight initialization, but using the same random seeds (*i.e.*, same weight initialization values) for both methods. To evaluate the performance on the classification task, I rely on the average area under the ROC curve (AUC) and F1 score across the five runs. I choose the latter instead of accuracy because it is a combination of precision and recall that are both important in the context of CTA. I also compare the performance on the energy and the direction reconstruction tasks with their resolution curves (see Section 1.4.4 for more details). As I repeat the experiment five times for all the models, I illustrate the variability due to the different initializations by drawing the resolution curves as surfaces. The envelope of the surface represents the min / max per bin and the dots represent the average resolution per bin of the five random seeds. This "average" resolution is not related to any physical reality as the resolution is a statistical measure of the error of a particular model. However, it gives a trend of the model performance and is useful for readability.

3.5.5.1 Dataset

To compare bilinear interpolation and Indexed Convolution with the very deep network presented in Section 3.5.5.2, I use the LST4 monotrigger dataset (see Section 1.5 for more details) as this thesis mainly focuses on LST data analysis. In addition to pixel charges, I also use the time information per pixel as a supplementary channel. As this comparison does not involve oversampling method, I adopt a different strategy than in Section 3.5.4.1. The resolution of resampled images with bilinear interpolation being arbitrary, I set it 55×55 pixels. As a result, the number of pixels of interest is the same for the original hexagonal grid images and the resampled images.

A series of selection cuts (see Section 1.4.2 for details) on image intensity, shower size and truncated showers are applied to the data in order to keep good quality events. More specifically, events:

- with a total intensity less than 300 photoelectrons,

- that do not pass the tail-cut cleaning (picture threshold: 6, boundary threshold: 3, minimum picture pixel neighbors: 2),
- with a leakage more than 0.2 (border width: 2, intensity).

are discarded. These cuts are not necessary to obtain interesting results with deep learning (see Section 4.2.4). I choose to apply them to benefit from the Indexed Convolution models already trained for the experiment described in Section 4.2 that was realized earlier.

3.5.5.2 Network

The network used for this experiment is the ResNet-56 presented in [He et al., 2016b]. As IACT images are rather small (*e.g.*, 1855 pixels for the LST), I choose the CIFAR-10 version. I train one network per task and change the output size of the ResNet to two and one for respectively the classification and direction reconstruction, and the energy reconstruction task. I use the standard cross-entropy loss for the classification task and the $L1$ loss for regression tasks, as it achieved better results than the $L2$ loss usually used for regression in preliminary experiments. The models are the same for the bilinear interpolation method and Indexed Convolution, except the size of the convolution kernels. For 2D resampled images, the kernels are of size 3×3, while for the hexagonal images the kernels are of size 7, representing the nearest neighbors of the pixel of interest.

I train all models with the same hyperparameters detailed in Appendix B, Table B.5.

3.5.5.3 Results

The results presented in Table 3.6 exhibit similar performance on the classification task for both methods. However, the standard deviation of Indexed Convolution strategy is 0.001 for both AUC and F1, while the results of the bilinear interpolation method spread more significantly (5 times higher for the AUC and 6 times for the F1 score).

Table 3.6 – AUC and F1 score of the gamma / proton classification task for the bilinear interpolation method and Indexed Convolution

Method	AUC	F1 score
Bilinear interpolation	0.950±0.005	0.944±0.006
Indexed Convolution	**0.954±0.001**	**0.949±0.001**

On the energy reconstruction task, as presented in Figure 3.29, the average performance of both methods is very close. However, Indexed Convolution's resolution is constantly lower than the one of the bilinear interpolation, or at least equivalent. Moreover, it demonstrates a smaller dispersion under 100 GeV, about half of the dispersion of the bilinear interpolation method.

For the direction reconstruction task, as presented in Figure 3.30, the Indexed Convolution strategy outperforms the bilinear interpolation method. Its average resolution is significantly lower for every bin, up to 0.04° at 3 TeV. Again, its results across the five runs are significantly less spread, up to 0.02° at 1.3 TeV.

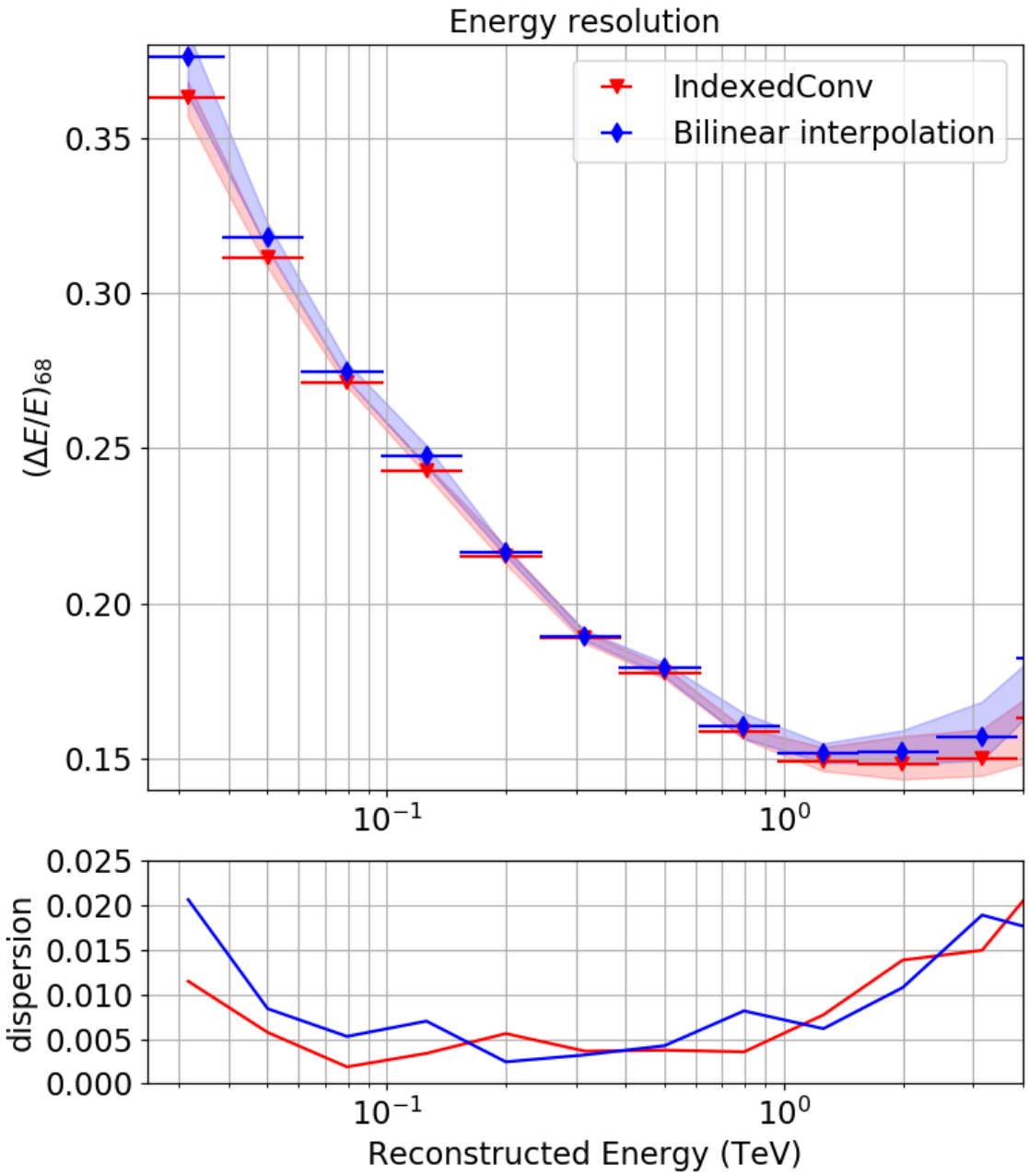


Figure 3.29 – Energy resolution as a function of the energy in the LST energy range (lower is better). Comparison of the performance on the energy regression task between the bilinear interpolation method and Indexed Convolution. The lower plot represents the variability per bin across the different seeds.

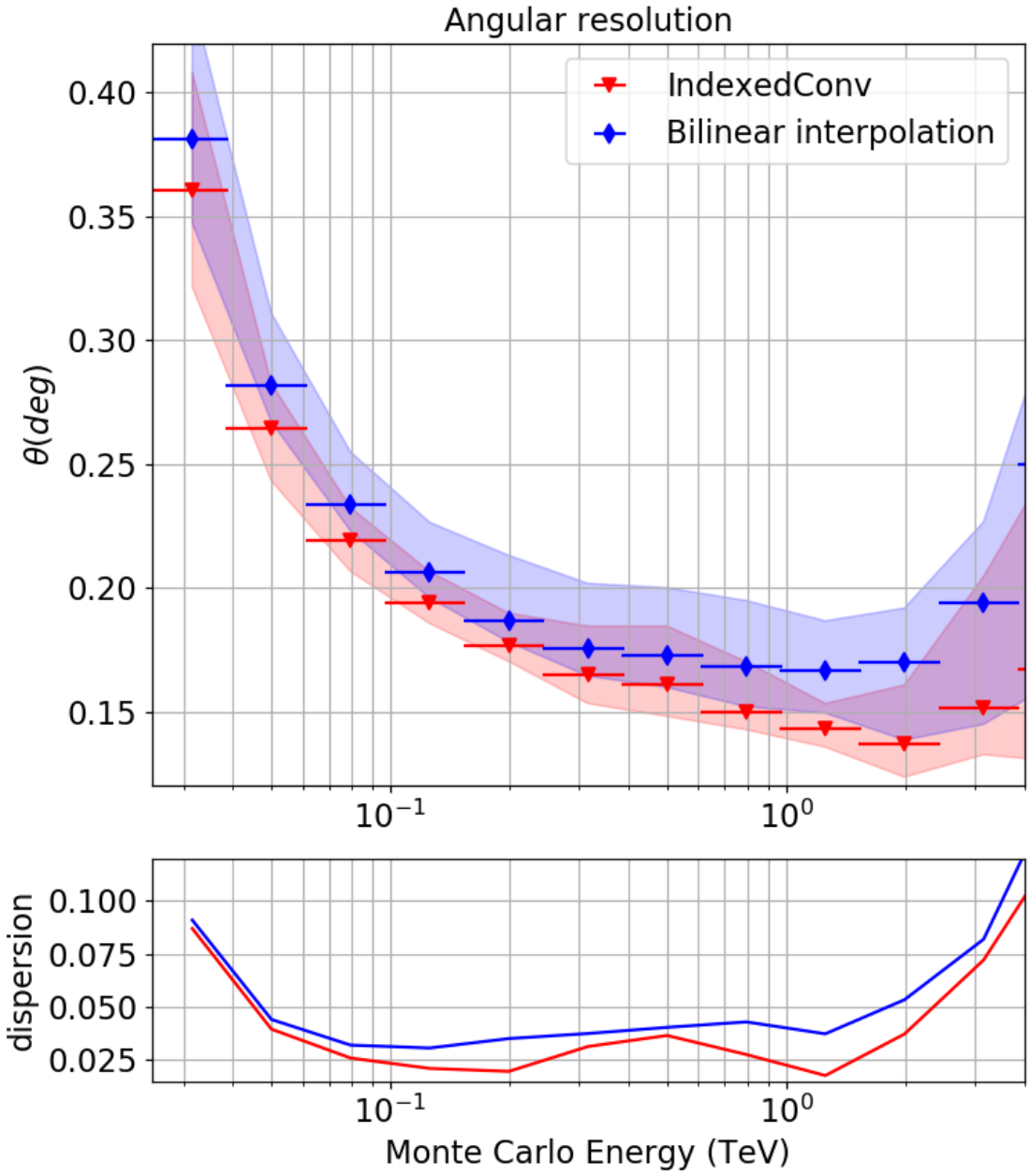


Figure 3.30 – Angular resolution as a function of the energy in the LST energy range (lower is better). Comparison of the performance on the direction regression task between the bilinear interpolation method and Indexed Convolution. The lower plot represents the variability per bin across the different seeds.

Performing full event reconstruction with a very deep network, the Indexed Convolution with hexagonal kernels exhibits better performance than the bilinear interpolation method. The angular resolution in particular is significantly lower. Moreover, the results of Indexed Convolution on the classification and the direction reconstruction tasks are much less spread. The predictions are more stable. This might stress a better robustness of Indexed Convolution with hexagonal kernels on hexagonal grid images. Robustness is crucial for IACT data analysis. As we will see in Chapter 5 and Section 6.2.3, there are discrepancies between Monte Carlo simulations and real data. More robust models will allow a better analysis of real data. For example, the difference of dispersion of 0.02° on the direction reconstruction task is of the order of the size of the Crab Nebula recently measured by the H.E.S.S. observatory [Collaboration et al., 2019].

3.6 Indexed Convolution Computing Performance

The current implementation of Indexed Convolution with PyTorch as backend shows a decrease in computing performances compared to the standard convolution method implemented in PyTorch. We observe an increase of RAM usage of factors varying between 1 and 3 and training times of factors varying between 4 and 8 on GPUs (depending on the GPU model), of factor 1.5 on CPU (but slightly faster than masked convolutions on CPU) depending on the network used. These drawbacks are mainly related to the use of unoptimized codes and could be fixed by the use of optimized CUDA and C++ implementations. However, in [Anderson et al., 2017] the authors present two new GEMM-based algorithms to reduce the memory consumption of the GEMM-based convolution. Especially, they reduce the cost of the *im2col* operation from $O(K^2 C_{in} HW)$ to $O(C_{out} HW)$ and $O(KW)$ with K the kernel size, C_{in}, H, W respectively the number of channels, the height and the width of the input and C_{out} the number of channels of the output. In [Dukhan, 2019], the author proposes the Indirect Convolution algorithm. It consists in replacing the costly *im2col* operation with an indirection buffer. These two methods may be worth trying to reduce the memory consumption of Indexed Convolution, although the Indirect Convolution algorithm has limited applicability to the backward pass on convolution. Noteworthy, optimizing Indexed Convolution is out of the scope of this thesis.

3.7 Summary

In this chapter, I have discussed the issue of applying deep learning algorithms to unconventional grid images, in particular hexagonal pixel images. As highlighted in Chapter 2, deep learning allowed great success in computer vision but deep learning frameworks are designed for Cartesian grid images. I have reviewed different methods to handle hexagonal lattice images with standard deep learning libraries, such as PyTorch or TensorFlow. Most of them imply to fit hexagonal grid into Cartesian one either through resampling or shifting the images. However, these methods have important drawbacks, among others image distortion and increase of memory consumption.

Indexed Convolution. To overcome these issues we have proposed Indexed Convolution and Pooling operators that can apply convolution to any type of pixel grid given that the neighborhood of each pixel is provided. I have validated our method on standard images as well as on the special case of hexagonal lattice images, exhibiting similar performances as standard convolutions. We have also compared Indexed Convolution strategy with various resampling methods on CTA data analysis, more precisely on the classification task, for different telescope types, with a narrow convolutional network. The results show that Indexed Convolution is consistently, together with bilinear interpolation, superior. However, it is worth noticing that the latter experiment has been done on the classification task only, while IACT data analysis also requires energy and direction reconstruction, with a very simple network. I then have extended the comparison between Indexed Convolution and bilinear interpolation on classification and energy and direction reconstruction tasks with a very deep network (ResNet-56, CIFAR-10 version). In these conditions, the Indexed Convolution with hexagonal kernel strategy outperforms the bilinear interpolation method. The average performance for every task is slightly but constantly better. The most interesting insight of this experiment is that the results across the five runs spread less with Indexed Convolution. This is significant for the physics, in particular for the direction reconstruction task. This may also demonstrate a better robustness of Indexed Convolution with hexagonal kernels over resampling methods to process IACT data.

Advantages. Both resampling and image shifting methods make possible the use of out the box operators already available in current deep learning frameworks. However, they increase the size of the transformed image. They add useless pixels of padding value for the resampled image to be rectangular and / or multiplying the number of existing pixels. This increases the amount of localities to process for convolution compared to Indexed Convolution. Moreover, they are restricted to regular grids. The main advantage of Indexed Convolution (and Pooling) compared to resampling and image shifting methods is that it is much more general and can be applied to any grid of data, enabling unconventional image representation to be addressed without any preprocessing. It only needs the index matrices to be built prior the training and inference processes, one for each convolution of different input size. No additional preprocessing of the image is then required to apply convolution and pooling kernels.

Drawback. Besides, the current implementation of Indexed Convolution shows a decrease in computing performances compared to the standard convolution method imple-

mented in PyTorch. This could be fixed with an optimized CUDA implementation. I also have highlighted two leads to improve the Indexed Convolution process, but optimizing Indexed Convolution is out of the scope of this thesis.

$\sim \gamma \sim$

In this chapter, I have demonstrated the superiority of the Indexed Convolution strategy over resampling methods for single task analysis of IACT data, in particular LST data. In the following of this thesis, I will implement all the network architectures with Indexed Convolution with hexagonal kernels.

4

γ-PhysNet: Addressing LST Data Analysis as a Multi-Task Problem

In the previous chapters, I have presented gamma astronomy and Imaging Atmospheric Cherenkov Telescope (IACT) data analysis. I have explained why we need to explore deep learning for this task, especially for analysis of the Cherenkov Telescope Array (CTA) data, which will amount to several petabytes per year. I have reviewed Deep Learning for astrophysics and in particular IACT data analysis. I have highlighted that previous works address each task separately, although they are deeply related. IACT data analysis can indeed be considered as a multi-task problem. It may then benefit from the deep multi-task learning techniques discussed in Section 2.2.

In this chapter, I propose a single multi-task learning architecture, called γ-PhysNet, to achieve full event reconstruction (*i.e.*, gamma/proton classification, energy and direction regression) from LST data, in the context of single-telescope analysis. The contribution of this architecture is its multi-task block inspired by the physics of the phenomenon that is expected to avoid the degeneracies introduced by the atmospheric reconstruction. I first demonstrate the interest of multi-task learning over single task learning, in the context of single-telescope analysis. As for other multi-task learning problems, we will see that an auxiliary task, here, the virtual impact point reconstruction, helps regularize the model. I then carry out an extensive study of attention methods and their benefits for γ-PhysNet. I detail this study for three different sets of data selection cuts (high, mid and low cuts) to address relevant use cases.

Finally, I open the black box of γ-PhysNet to understand its internal mechanics. I compute the receptive field of its backbone. I also use a visual explanation method to analyze its behavior for some well reconstructed and badly reconstructed events.

4.1 γ-PhysNet: a Deep Multi-Task Architecture for LST Single-Telescope Analysis

As the computation time is crucial for CTA data analysis, in particular for real-time analysis, I design γ-PhysNet as a hard parameter sharing multi-task architecture to reduce the number of required operations as opposed to using a specific network for each of the task to address. Moreover, hard parameter sharing reduces the risk of overfitting (see Section 2.2 for details). As illustrated in Figure 4.1, γ-PhysNet is composed of a shared encoder and a multi-task block. The backbone encoder is fed with two-channel IACT data (pixel charge and temporal information, also denoted peak position, see Section 1.3.4 for details). The encoder provides the multi-task block with latent features to separate gamma events from background noise (mainly proton events) and reconstruct the most critical physical parameters that are the energy and arrival direction of the gamma ray. The model also reconstructs the virtual impact point parameter as an auxiliary regression task. Even though the impact point is not needed by astronomers for higher-level analysis, physics shows that this parameter provides meaningful information to solve energy and direction reconstruction tasks. The shower captured by the telescope's camera for given energy and direction is indeed different depending on the impact point. The further the impact point is from the telescope position, the more elongated the ellipsoid of the shower formed in the camera is. Furthermore, rotating the impact point around the telescope makes the ellipsoid of the shower rotate around the virtual projection of the event source in the camera. Finally, the impact point also modulates the intensity of the measured Cherenkov image. Thus, we can expect a multi-task architecture, to avoid these degeneracies.

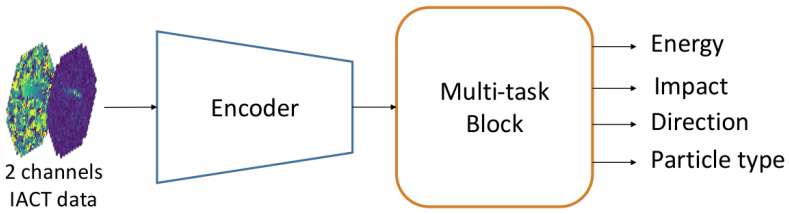


Figure 4.1 – Proposed architecture (γ-PhysNet) for IACT data analysis

4.1.1 Backbone Encoder

To choose the convolutional part of γ-PhysNet, I have probed different architectures. Preliminary experiments have compared the following typical networks:

- shallow convolutional neural networks composed of 3, 4 and 5 convolution layers (of various widths), with max and average pooling as subsampling layers,
- a deep neural network, the VGG-16 [Simonyan and Zisserman, 2015],
- very deep networks, DenseNets [Huang et al., 2017] with various numbers of layers (up to 97) and different ResNets [He et al., 2016b, He et al., 2016a], including the CIFAR version, as LST images are rather small (1855 pixels).

The ResNets were globally superior, and among them the ResNet-56 (CIFAR version) with full pre-activation obtained the best results in terms of performance and training stability. The backbone encoder of γ-PhysNet is then the convolutional part of the ResNet-56 with customization. More precisely, it is implemented with IndexedConv [Jacquemont et al., 2019a] as LST images have hexagonal pixels. As demonstrated in Section 3.5.5, these specific operators obtain better results on hexagonal grid images than standard resampling methods. Moreover, they allow avoiding additional preprocessing and making efficient use of smaller hexagonal kernels, adapted to the raw pixel configuration.

4.1.2 The Multi-Task Block: the Heart of γ-PhysNet

The multi-task block design shown in Figure 4.2 is directly derived from the physics of the tasks to address. Inspired by [Iizuka et al., 2016], it is composed of two fully connected sub networks, a global feature network dedicated to energy regression and a local feature network.

Global Feature Network. Energy can be considered as a global parameter with regard to the input images, as for a given arrival direction and impact point, the number of photoelectrons in the acquired image is roughly proportional to the primary gamma ray energy [Völk and Bernlöhr, 2009]. The first sub-network then applies a global average pooling to the feature maps produced by the encoder. The resulting vector is then processed by a fully connected layer of 256 neurons, followed by ReLU activation. Finally, a fully connected layer outputs the energy regression.

Local Feature Network. On the other hand, gamma/proton classification, and arrival direction and impact point regression can be considered as local parameters as they depend on local and spatial information. As a first approximation,

- the position of the shower in the image depends on the arrival direction,
- its orientation depends on the impact point,
- its shape (elongated or not) depends on the direction and the impact point,
- its contour and intensity pattern depend on the particle type.

All these dependencies stress the need to exploit the local and spatial information of the latent feature maps produced by the ResNet encoder. Therefore, the second sub-network starts with a flatten operation that rearranges all the pixels of the backbone feature maps as a vector. Thereby, the operators of the different task branches can learn the relevant relationships between the features extracted by the encoder. The local feature network is then composed of:

- a classification branch that consists of a fully connected layer (2 neurons),
- an arrival direction branch and an impact point branch, both consisting of 2 fully connected layers (256 and 2 neurons respectively). The first layer is shared between both tasks and is followed by a ReLU activation.

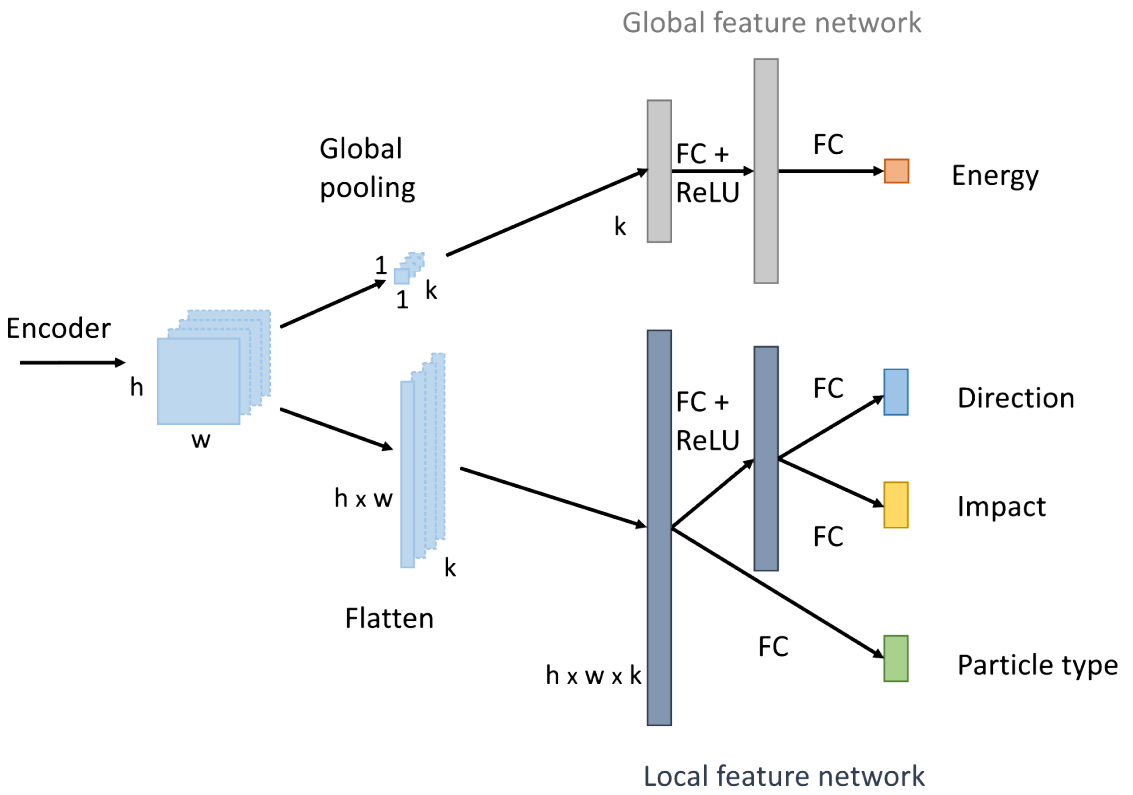


Figure 4.2 – Physically inspired multi-task block.

In addition, in the search of a model that captures better the physics behind the gamma event reconstruction, I have also probed simpler and more complex architectures for this multi-task block, as illustrated in Figure 4.3. All of them underperformed compared to the architecture detailed in this section. However, all the models with a dedicated branch from the encoder feature maps for the energy reconstruction obtained better results than the model 4.3b on all the tasks. This confirmed that the energy reconstruction relies on different features than the other tasks. Furthermore, the models that associate direction and impact reconstruction in the same branch also performed better, confirming that both tasks depend on the same spatial features. The model described in 4.3d tried to answer the physical intuition that knowing the particle type, the direction and the impact point helps reconstruct the energy. This intuition is related to the degeneracies introduced by the atmospheric detection of the gamma rays. However, this model slightly underperformed compared to γ-PhysNet.

4.1.3 Augmenting the Backbone with Attention

The backbone of γ-PhysNet is the convolutional part of a ResNet-56 [He et al., 2016b] (CIFAR-10 version) implemented with Indexed Convolution, as detailed in Section 4.1.1. It is composed of an initial convolution and three residual stages whose first layer is a subsampling performed with a strided convolution.

To better adapt to the context of IACT data analysis, I augment the backbone with attention. By learning to focus on relevant features, it may be easier for the network to exploit the information of the shower among the noise. Moreover, it may help to handle truncated shower that can disturb the reconstruction. As illustrated in Figure 4.4, I insert

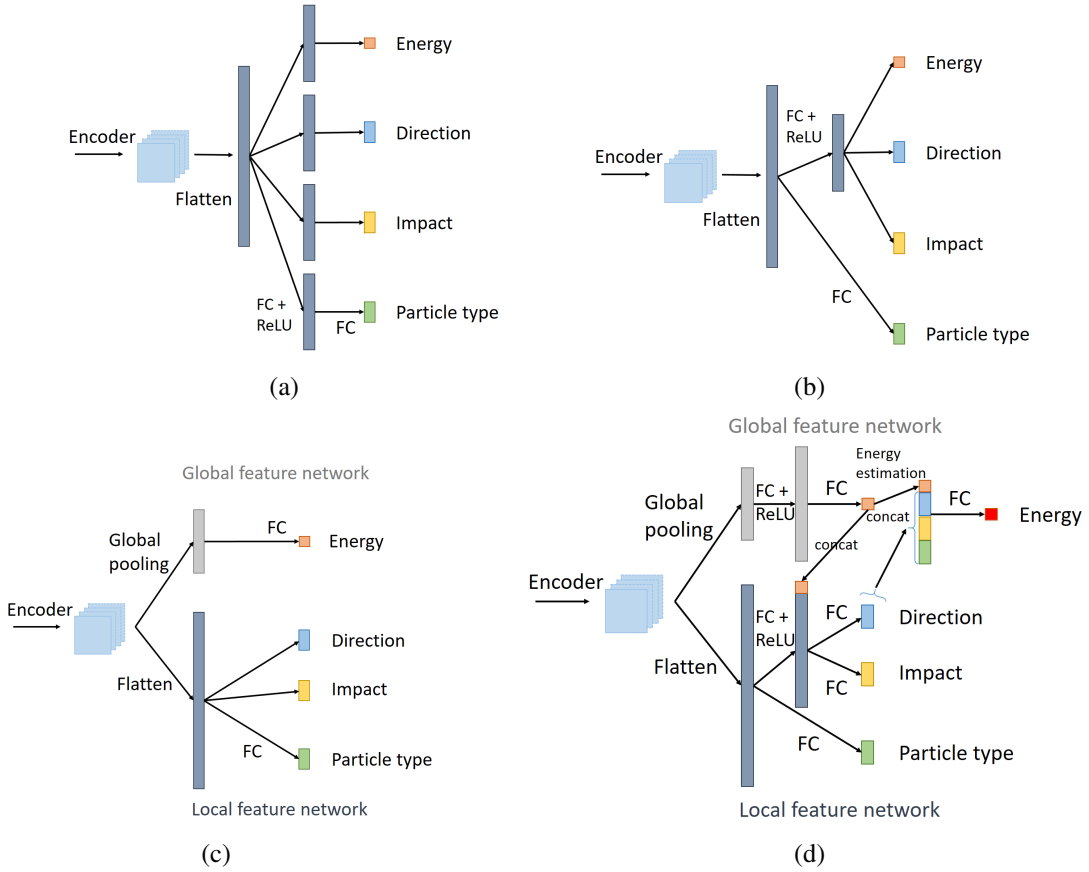


Figure 4.3 – γ-PhysNet multi-task block alternatives. In block a, all the task-related networks are similar, composed of two fully connected layers. In block b, all regression tasks have a similar network, while the classification is performed with a single fully connected layer from the flatten output of the encoder (as in the original implementation of the ResNet). Block c is a simplified version of γ-PhysNet multi-task block where the intermediate fully connected layers have been removed. Block d is more complex. It relies on an intermediate estimation of the energy that is injected in the regression of the direction and the impact point. The final value of the energy is reconstructed from this estimation and the inferred direction, impact point and particle type.

the attention modules after every residual stage instead of every residual block to benefit from the attention for each feature size scale with a reasonable computational cost.

In the following of this chapter, I focus on self-attention (SA), Squeeze-and-Excitation (SE) and dual attention (DA) to probe three different types of attention. Self-attention is a rather complex spatial-attention method modeling long-range dependencies, while Squeeze-and-Excitation is a less costly channel-wise attention method and dual attention combines Squeeze-and-Excitation with a simple pixel-wise attention mechanism to perform both channel-wise and spatial attention. These three attention methods, detailed in Section 2.3, have a reduction ratio to control the strength of their bottleneck, and so their computational cost and their generalization capability.

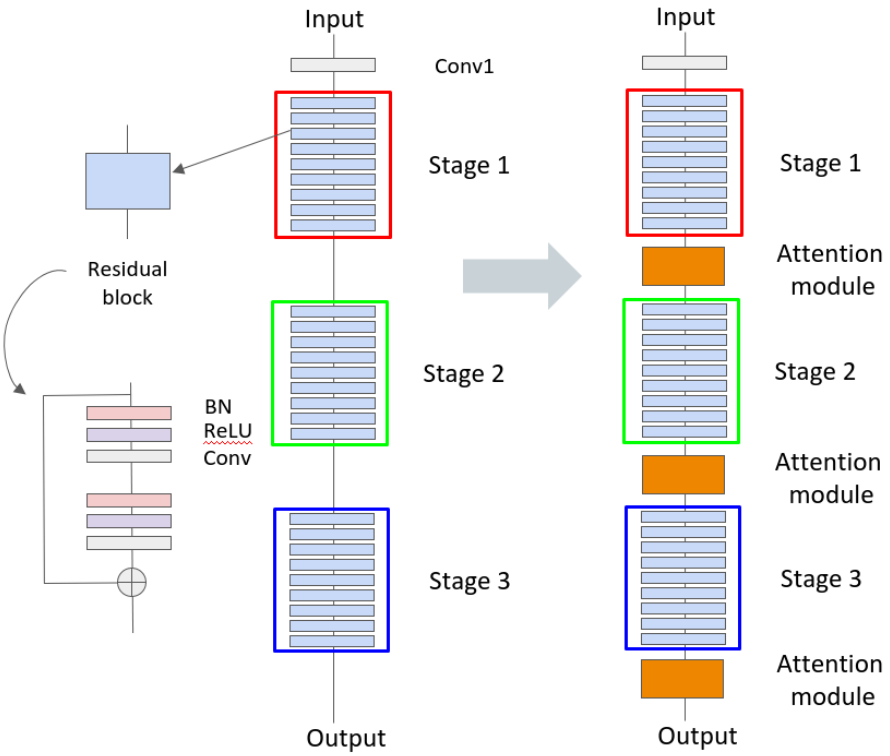


Figure 4.4 – Adding attention to γ-PhysNet backbone. The attention modules are inserted after every stage of the ResNet-56.

4.2 Experiments on the LST4 Monotrigger Dataset

In this section I first demonstrate the benefits of γ-PhysNet for LST data analysis. More specifically, I probe the performance improvement brought by its physically inspired MT block. Experiments are conducted on the LST4 monotrigger dataset described in 1.3.4. I compare its performance with single task networks and the widespread Hillas + RF method on the gamma/proton classification and energy and direction regression tasks. I also demonstrate the interest of adding the impact point regression as an auxiliary task experimentally. Then, I study attention mechanisms for γ-PhysNet backbone in three different configurations to address different analysis use cases.

4.2.1 Training

For the following experiments, all the models evaluated (RF and NN) are trained using the data from the four telescopes of the LST4 monotrigger dataset to provide a more accurate overview of the data variability. The models are trained on gamma diffuse events, so as to reconstruct events coming from any directions within the field of view, and proton events.

Training Parameters. All the neural networks are trained with the same hyperparameters detailed in Appendix B, Table B.7. We must take into account the fact that a single experiment typically requires between 4 and 85 hours (depending on the selection cuts) on a V100 GPU hardware in an optimized computational center. Consequently, an advanced optimization study of my architecture and the networks I compare to is not feasible at the step of the project. However, starting from the default optimized hyperparameters of ResNet, extensive preliminary experiments allowed a common and well-performing hyperparameter set to be identified and to be used in the proposed experiments.

Loss Functions. I use the standard cross-entropy loss for the classification task and the $L1$ loss for regression tasks, as it performed better than the MSE in my preliminary experiments. In addition, as I consider multi-task regression for gamma rays but also gamma and protons classification, I should prevent proton events from disturbing the learning of energy and direction task for gamma events. To solve this, I rely on a masked loss method. More precisely, the loss of the regression parameters (energy, arrival direction and impact point) is set to zero when particles are protons

$$L_t = \frac{1}{M} \sum_{N} \begin{cases} l_i, & \text{if particle is a gamma} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

where L_t is the loss of task t for a batch of data with length N, M the number of gamma events in the batch and l_i the loss of example i.

Task Balancing. As explained in Section 2.2.1, the way of combining the loss functions in multi-task learning is crucial, in particular for hard parameter sharing architectures. Several adaptive methods exist to balance the tasks and avoid that the easiest one takes over the learning process, as detailed in Section 2.2.2. To choose the method to balance the task in γ-PhysNet, I have carried out extensive preliminary experiments with:

- GradNorm [Chen et al., 2018] ; on the contrary to the statements of the paper, my experiments with CTA data showed that α, a parameter controlling the asymmetry of the tasks, is challenging to tweak. Although the authors claim their method performs better than uncertainty estimation, it was not the case so far for my experiments.
- Multi-objective optimization [Sener and Koltun, 2018] ; although the authors claim their method performs better than uncertainty estimation, it was not the case so far for CTA data analysis with multi-task learning.
- Uncertainty estimation method [Kendall et al., 2018] ; it is efficient as it only requires to train an additional parameter that is not part of the network's computational graph. My preliminary experiments show that this method works the best for CTA data analysis with a multi-task network.

To balance the different tasks, I then use the uncertainty estimation method presented in [Kendall et al., 2018] and detailed in Section 2.2.2.1. The task log-variances are learned with the hyperparameters detailed in Appendix B, Table B.8.

Robustness. For reproducibility, and to better understand the robustness and the variability of the proposed architecture, I repeat the experiments for all deep neural network configurations with six different random seeds for parameter initialization.

4.2.2 Evaluation metrics

Although the proposed architecture is trained on gamma diffuse events to deal with a rich variability of event configurations, it is evaluated on gamma point-like events, to comply with gamma-ray astronomy standardized practice and most common scientific use cases. For the gamma/proton classification task, the overall performance of the network is given by the area under the ROC curve (AUC), the precision and the recall. To measure the performance of the proposed architecture on energy and direction reconstruction tasks I compute their resolution curves and the associated relative biases (see Section 1.4.4 for more details), and present them in the LST energy range.

As all neural network models have been trained with six different seeds for parameter initialization, I illustrate the variability of these different runs by drawing the resolution curves as surfaces. The envelope of the surface represents the min/max per bin, and the dots represent the average resolution per bin of the five seeds. This "average" resolution is not related to any physics reality as resolution is a statistical measure of the error of a particular model. However, it gives a trend of the model performance and is useful for readability and a more reliable model analysis.

When comparing only multi-task models, I also evaluate their global performance through their sensitivity curves (see Section 1.4.4 for more details). These curves are preliminary as the library used to compute them doesn't include yet all the three standard conditions in CTA. More than measuring the absolute performance of the models, these sensitivity curves allow for an overall model comparison. Again, the variability due to parameter initialization is represented as a surface while the line represents the average sensitivity par bin.

4.2.3 Multi-Task Learning Performance

In this section I evaluate the interest of multi-task learning for IACT data analysis, *i.e.*, gamma/proton classification, energy and direction regression. I conduct this analysis in the single-telescope context, the most difficult case. I compare the performance on these tasks of the following models:

- γ-PhysNet. The proposed architecture.
- γ-PhysNet w/o impact. To evaluate the importance of the impact point regression as an auxiliary task, I train γ-PhysNet without the impact point task.
- ResNet-56. To demonstrate the contribution of multi-task learning to the performance of the proposed architecture, I train three ResNet-56 (the backbone of γ-PhysNet) to solve the different tasks. The main difference with the architecture

described in [He et al., 2016b] is that for the direction reconstruction and gamma/proton separation tasks I replace the final global average pooling by a flatten operation to keep spatial information before the fully connected layer.

- Hillas + RF. This is a widespread analysis method for IACTs event reconstruction (see Section 1.4.3.1 for more details) that serves as a baseline.

I cannot compare with [Shilon et al., 2019] and [Mangano et al., 2018] as the architectures presented are designed for stereo analysis while my architecture is designed for single-telescope analysis. I compare neither with [Nieto et al., 2017] as this work is related to a different telescope, is focused on classification and does not take into account the temporal information.

A series of selection cuts (see Section 1.4.2 for more details) on image intensity, shower size and truncated showers is applied to the data in order to keep good quality events. These cuts are standard in gamma astronomy and necessary for the comparison with Hillas + RF method that discards the bad quality events. More specifically, we discard the following events:

- with a total intensity less than 300 photoelectrons,
- that do not pass the tail-cut cleaning (picture threshold: 6, boundary threshold: 3, minimum picture pixel neighbors: 2),
- with a leakage more than 0.2 (border width: 2, intensity).

Thus, γ-PhysNet is trained on 388k gamma diffuse events, so as to reconstruct events coming from any directions within the field of view, and 236k proton events. The effect of the selection on the data distributions is illustrated in Figure 4.5.

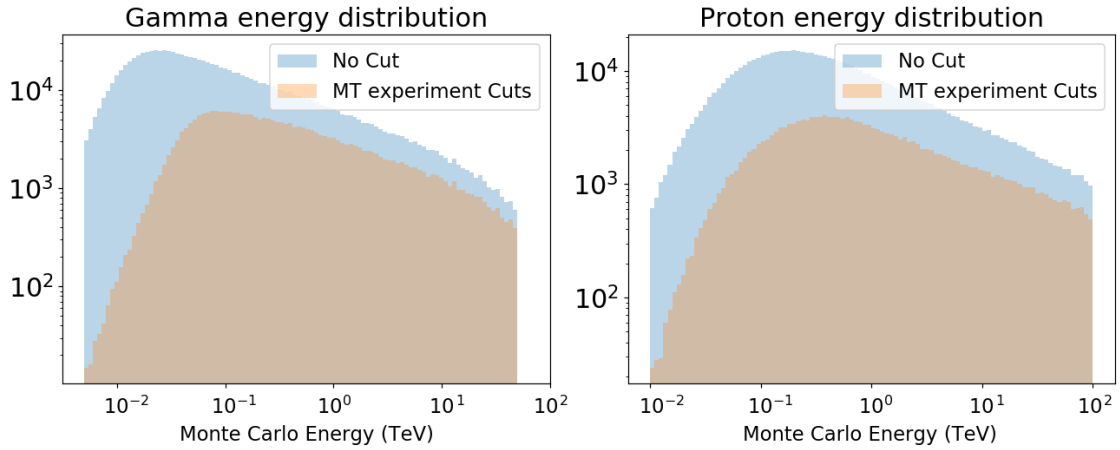


Figure 4.5 – Energy distributions of remaining gamma (*on the left side*) and proton (*on the right side*) events before and after the selection cuts for the multi-task experiment.

4.2.3.1 Gamma / Proton Classification

The results summarized in Table 4.1 clearly show that deep neural networks outperform the widespread Hillas + RF analysis method in both AUC and recall. More specifically,

the proposed architecture improves the AUC by 6.9% and the recall by 60.1% compared to Hillas + RF, while they have a similar precision. The improvement of the recall will lead to a better sensitivity as much more gamma events are kept. The contribution of multi-task in γ-PhysNet architecture is also significant compared to the single task approach relying on the ResNet architecture, in particular for the recall. However, there is no benefit to add the impact point regression as an auxiliary task for gamma/proton classification.

Table 4.1 – AUC, precision and recall of the gamma/proton classification task for the different models. The precision and recall are computed for a default gammanes threshold of 0.5.

Model	AUC	Precision	Recall
Hillas + RF	0.898	0.956	0.593
ResNet-56	0.954±0.001	0.956±0.001	0.942±0.001
γ-PhysNet	**0.960±0.002**	0.957±0.003	**0.956±0.006**
γ-PhysNet w/o Impact	**0.961±0.002**	**0.960±0.001**	0.949±0.003

4.2.3.2 Energy Reconstruction

Figure 4.6 shows that all the evaluated deep neural networks outperform the widespread Hillas + RF method for the energy reconstruction task. γ-PhysNet decreases the relative error on the energy task by up to 0.08 at high energies and up to 1.1 at 31 GeV (the point of the Hillas + RF curve is out of the plot). The resolution curves of γ-PhysNet and the ResNet are very close almost everywhere. However γ-PhysNet has slightly better results below 200 GeV. At energies above 400 GeV, multi-task learning seems to degrade the performance, in particular without the regression of the impact point. This can be explained physically as the particle energy is strongly correlated with the total amount of light emitted by the atmospheric shower, which is correlated with the observed intensity in the camera and the distance from the telescope to the shower impact point. Besides, multi-task learning models have a higher dispersion than the single task model, yet adding the impact point regression task reduces the sensitivity to parameter initialization. This task helps regularize the model. The single task model is also the less biased, while γ-PhysNet have similar or smaller biases than the Hillas + RF method depending on the energy bin.

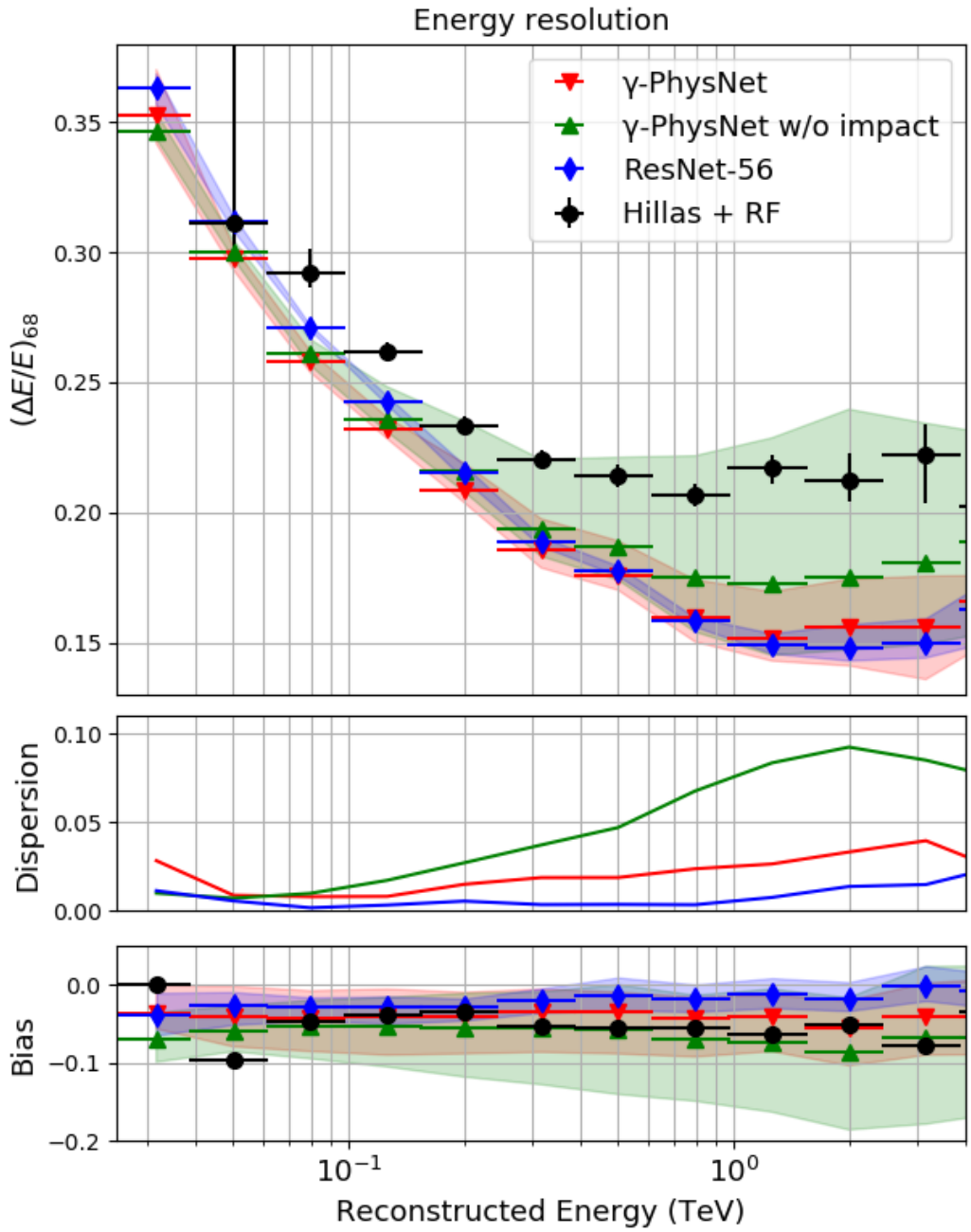


Figure 4.6 – Energy resolution as a function of the energy in the LST energy range (lower is better). Comparison of the performance on the energy regression task between the probed models. For all neural networks, the dots represent the average resolution of all the seeds per bin and the surface the min/max envelope, per bin too. For the Hillas + RF method, I have only carried out one run. Therefore, the error bars represent the 95% confidence interval. The dispersion represents the width of the surfaces per energy bin. The bias plot corresponds to the median relative error per energy bin.

4.2.3.3 Direction Reconstruction

As for the gamma/proton classification and the energy regression tasks, Figure 4.7 shows that deep neural networks outperform the widespread Hillas + RF analysis method for the direction reconstruction task. In particular, γ-PhysNet improves the performance by 0.03 to 0.3° compared to Hillas + RF. Moreover, for this task the contribution of multi-task learning is significant, improving the results by up to 0.08° compared to the single task network. The proposed architecture has also slightly better results with the impact point reconstruction as an auxiliary task, especially at higher energies ($> 1\,TeV$). Both multi-task learning models have a lower variability than the single task ResNet-56. All neural networks are globally less biased than the Hillas + RF method on the altitude reconstruction while all the methods have similar biases for the azimuth reconstruction.

4.2.3.4 Conclusion

My experiments show that the proposed architecture outperforms the widespread Hillas + RF analysis method on all the three tasks to address. In addition, multi-task learning improves the performance, especially for the direction reconstruction task. In particular, γ-PhysNet outperforms both the single task ResNet-56 and the Hillas + RF method. Moreover, it has a much lower dispersion than the ResNet, although it is not the case for the energy reconstruction task. Finally, adding the impact point reconstruction task allows reducing the sensitivity to parameter initialization for all tasks.

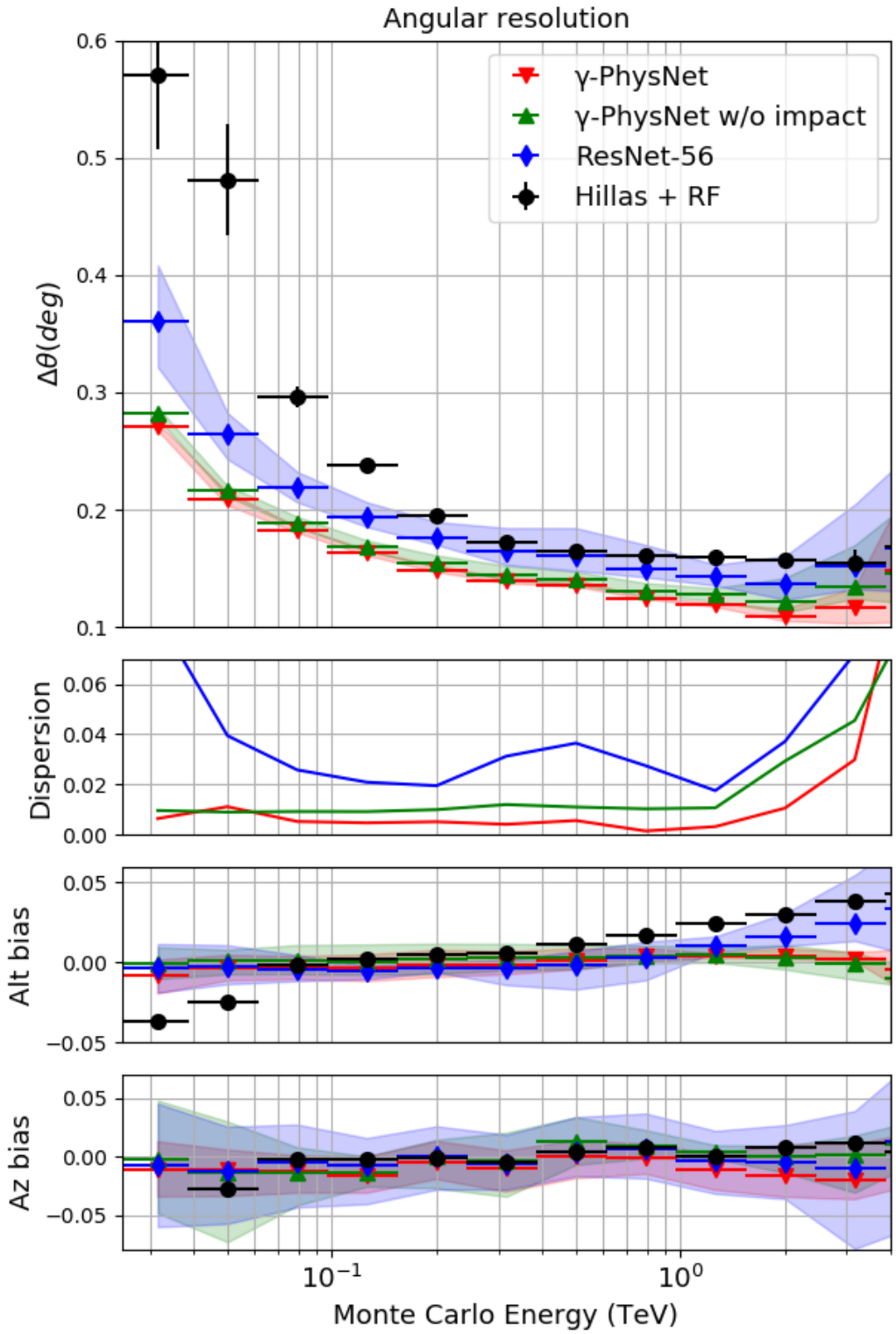


Figure 4.7 – Angular resolution as a function of the energy in the LST energy range (lower is better). Comparison of the performance on the arrival direction regression task between the probed models. The dispersion represents the width of the surfaces per energy bin. The bias plots correspond to the median relative error per energy bin for relatively the altitude and the azimuth reconstructions.

4.2.4 Study of the Impact of Attention Methods

In the previous sections I have introduced γ-PhysNet, a deep multi-task architecture for full event reconstruction from IACT images single-telescope analysis. I have demonstrated the benefits of its physically inspired multi-task block and its superiority over the widespread Hillas + RF method. In this section I focus on the backbone of γ-PhysNet, still in a multi-task learning context. Because of computing resource constraints, two types of experiments are carried out simultaneously.

First, the aim is to study the impact of adding attention modules on model performance. The three methods detailed in Section 2.3 are evaluated:

- Squeeze-and-Excitation,
- self-attention,
- dual attention.

Besides, I investigate three levels of intensity cuts to explore the capabilities of γ-PhysNet and of the attention methods. In particular, I try to answer the following questions. Does the improvement brought by attention, if any, vary with different data selections ? Is γ-PhysNet able to process events that are discarded by the standard analysis ? What spatial resolution can we achieve with highly selected events ?

Seeking for Sensitivity. In order to extend the energy range of interest compared to the experiments presented in Section 4.2.3, I relax the selection cuts. In particular, a lower cut on the image intensity will keep more low-energy events and allow processing events that are discarded by the standard analysis. It will help improve the sensitivity of the model in order to address relevant use cases for gamma astronomy. A more sensitive model below 100 GeV would indeed enhance the study of extragalactic objects and gamma-ray bursts. As a second use case, we can observe the temporal variability of the flux of well-known sources. Finally, we can also realize sky surveys to discover new sources. I then define the **low cuts** (LC) as:

- total intensity > 50 photoelectrons,
- tail-cut cleaning (picture threshold: 6, boundary threshold: 3, minimum picture pixel neighbors: 2),
- leakage (border width: 2, intensity, threshold: < 0.2).

It is worth noticing that it is much less selective than standard cuts applied for standard methods in single-telescope analysis context. This set of cuts keeps $874\,k$ gamma events compared to $388\,k$ for the cuts used in Section 4.2.3.

Experimenting with the Intensity Cut. I also explore a much higher cut on the intensity (> 1000 photoelectrons) with the other cuts being the same (cleaning and leakage), called **high cuts** (HC). It is highly selective and discards 92.2% of the training images, mainly at the lowest energies. The high-quality remaining events contains well defined and bright showers. In the context of single-telescope analysis, their parameters are easier to reconstruct, in particular their arrival direction. With these very strong cuts, I look for

the best angular resolution possible with γ-PhysNet. Although the high cuts are too selective, knowing the performance range of the model will help us define, in a future work, the optimal cut for sensitivity and spatial resolution.

Finally, I carry out a set of more limited experiments in the search of a trade-off between angular resolution and sensitivity. The **mid cuts** (MC), in addition to the cleaning and the leakage filters, select events with an intensity > 200 photoelectrons.

Effects of the Cuts on the Data. Figure 4.8 illustrates the distributions of energy of the remaining events for every set of cuts. Table 4.2 shows the composition of the training sets resulting of the three sets of selection cuts.

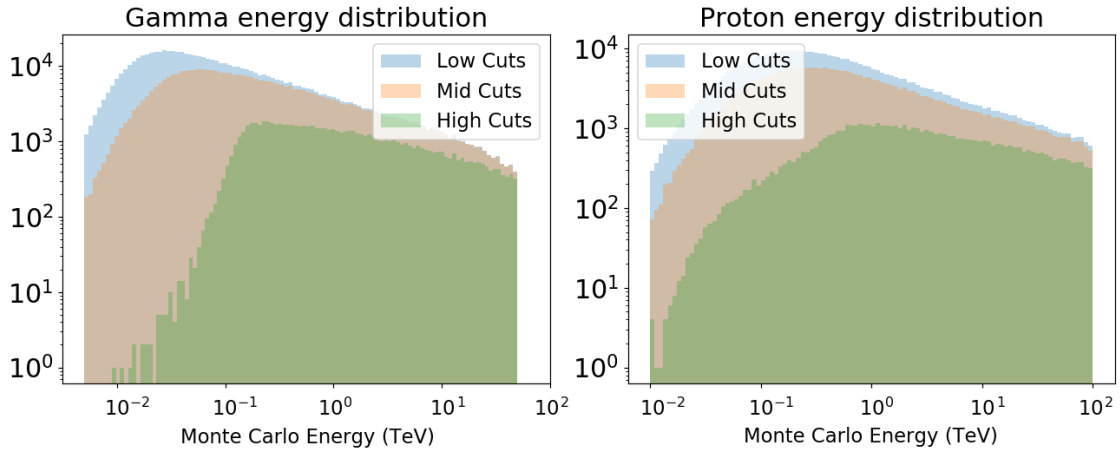


Figure 4.8 – Energy distributions of remaining gamma (*on the left side*) and proton (*on the right side*) events after the three selection cuts.

Table 4.2 – Training set composition for HC, MC and LC experiments.

	diffuse gammas	protons
HC	$121\,k$	$75\,k$
MC	$550\,k$	$328\,k$
LC	$874\,k$	$506\,k$

Attention Reduction Ratio. As detailed in Section 2.3, the probed attention methods have a hyperparameter to control their bottleneck, denoted reduction ratio. Relying on the extensive ablation study presented in Appendix C, I use the reduction ratio presented in Table 4.3 for the three attention mechanisms and the three selection cuts. Noteworthy, depending on the selection cuts, the selected reduction ratio per attention mechanism varies.

Table 4.3 – Selected ratio for the three attention methods and the three selection cuts.

Attention	HC	MC	LC
Squeeze-Excitation	2	4	4
Self-Attention	4	12	12
Dual Attention	8	16	16

4.2.4.1 Performance with the Low Cuts

Sensitivity. Figure 4.9 shows that all the models probed have similar sensitivity curves and dispersion at low energies. Remarkably, below 200 GeV, they outperform the MAGIC observatory (see Section 1.3 for details) although the performance of the latter is obtained in stereo mode. These very good results in mono mode are especially interesting for the study of extragalactic sources, such as gamma-ray bursts. In particular, γ-PhysNet may have a good sensitivity down to 20 GeV, the limit of the LST. Moreover, in the range [50, 80] GeV, relying on a single LST, the model performs 2 to 5 times better than MAGIC with 2 telescopes. However, the telescope properties are different and the improvement may not only due to the model itself. Besides, these results have to be confirmed with a future updated version of the sensitivity computation library.

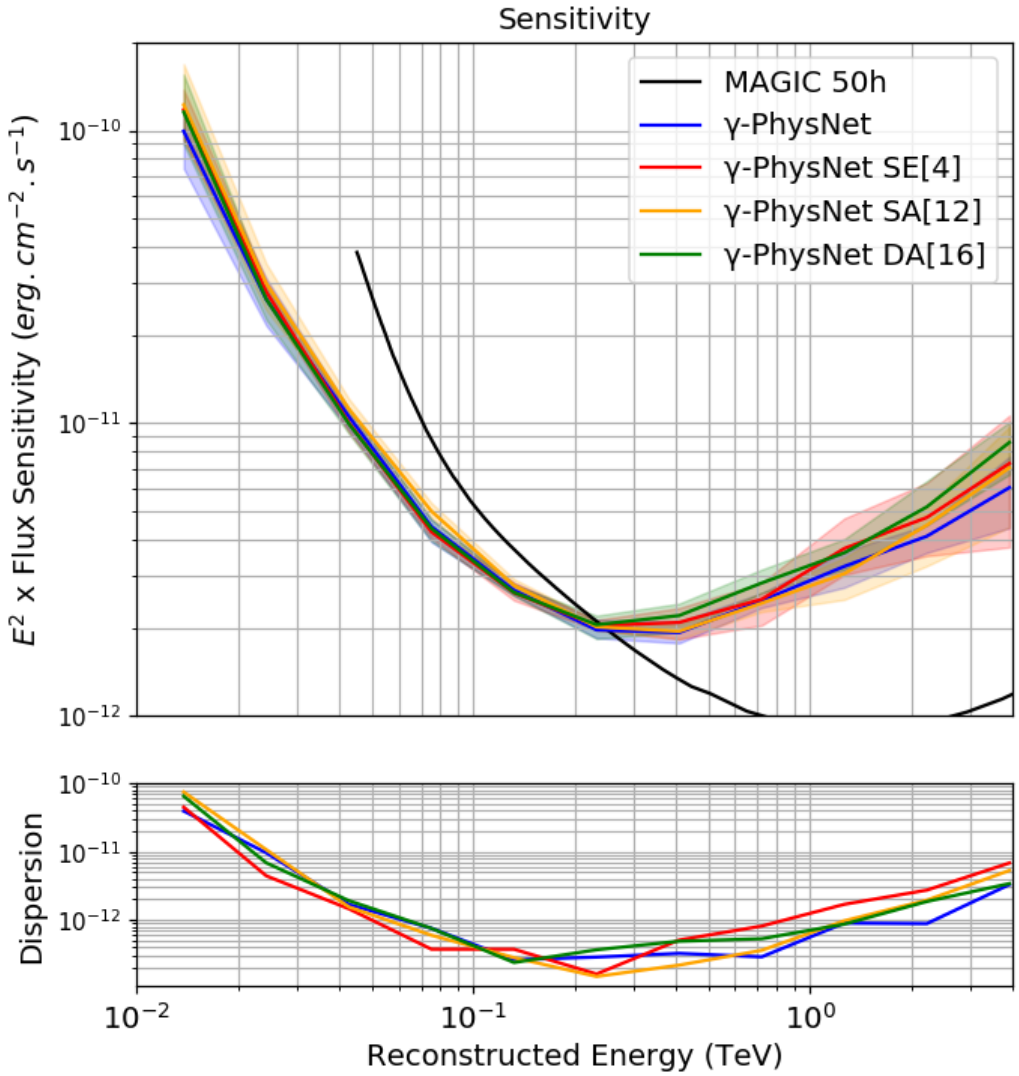


Figure 4.9 – Low cuts. Sensitivity. Comparison of the different attention mechanisms for γ-PhysNet. The surface represents the min/max envelope per bin, and the dots represent the average sensitivity per bin of the six seeds. The dispersion represents the width of the surfaces per energy bin. For reference, the black line corresponds to the performance of the MAGIC observatory in stereo mode. The value between brackets is the reduction ratio for the attention module.

Classification. With the low cuts, γ-PhysNet with the self-attention method performs slightly worse on the classification task, as shown in Table 4.4. This method learns long-range dependencies between pixels. Self-attention might give too much importance to pixels that are far from the shower. In order to verify this hypothesis, I could analyze in a future work the learned attention maps. The other models have comparable results within the standard deviation range.

Table 4.4 – Low Cuts. AUC, precision and recall of the gamma/proton classification task for the different models.

Model	AUC	Precision	Recall
γ-PhysNet	0.882±0.001	0.929±0.001	0.935±0.007
γ-PhysNet SE[4]	0.883±0.002	0.930±0.001	0.932±0.005
γ-PhysNet SA[12]	0.879±0.003	0.928±0.002	0.932±0.003
γ-PhysNet DA[16]	0.882±0.001	0.929±0.001	0.935±0.005

Energy Reconstruction. The performance on the energy reconstruction task is shown in Figure 4.10. The models with Squeeze-and-Excitation and dual attention perform better above 500 GeV. They improve the results up to 0.03, in particular at higher energies. Again, the self-attention method does not improve the average performance compared to γ-PhysNet without attention. However, all the networks with attention have significantly less spread results, showing that attention improves the robustness of the model. Besides, all the probed architectures have similar biases.

Direction Reconstruction. On the direction reconstruction task, as illustrated in Figure 4.11, again the models with Squeeze-and-Excitation and dual attention have slightly better results above 500 GeV, improving the resolution up to 0.02°. However, all the models have a competitive angular resolution in the context of single-telescope analysis, as we will see in Section 4.3. All the models have similar dispersion in their results, and comparable altitude and azimuth biases.

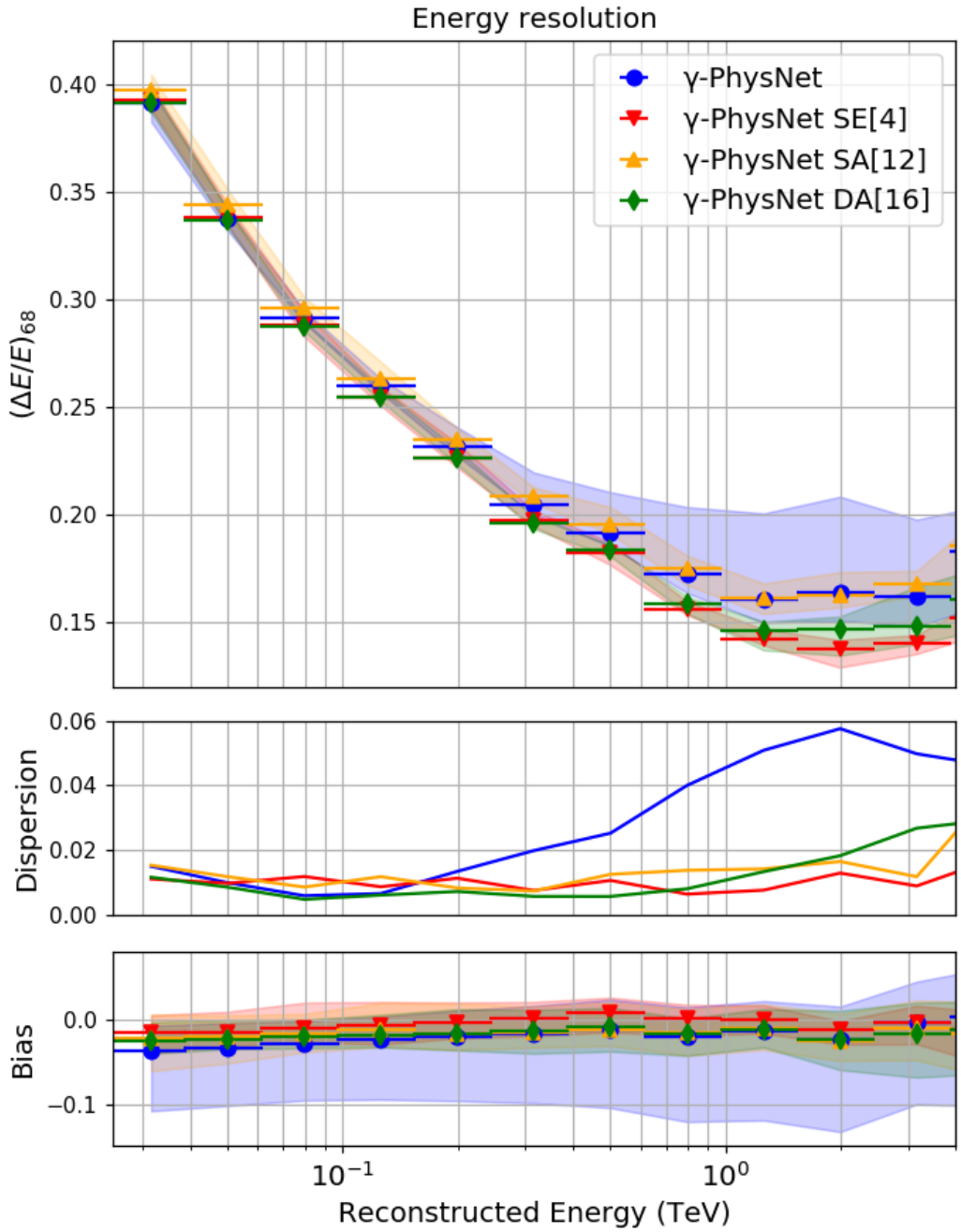


Figure 4.10 – Low cuts. Energy resolution curves of the different attention mechanisms for γ-PhysNet. The surface represents the min/max envelope per bin, and the dots represent the average resolution per bin of the six seeds. The dispersion represents the width of the surfaces per energy bin. The bias plot corresponds to the median relative error per energy bin.

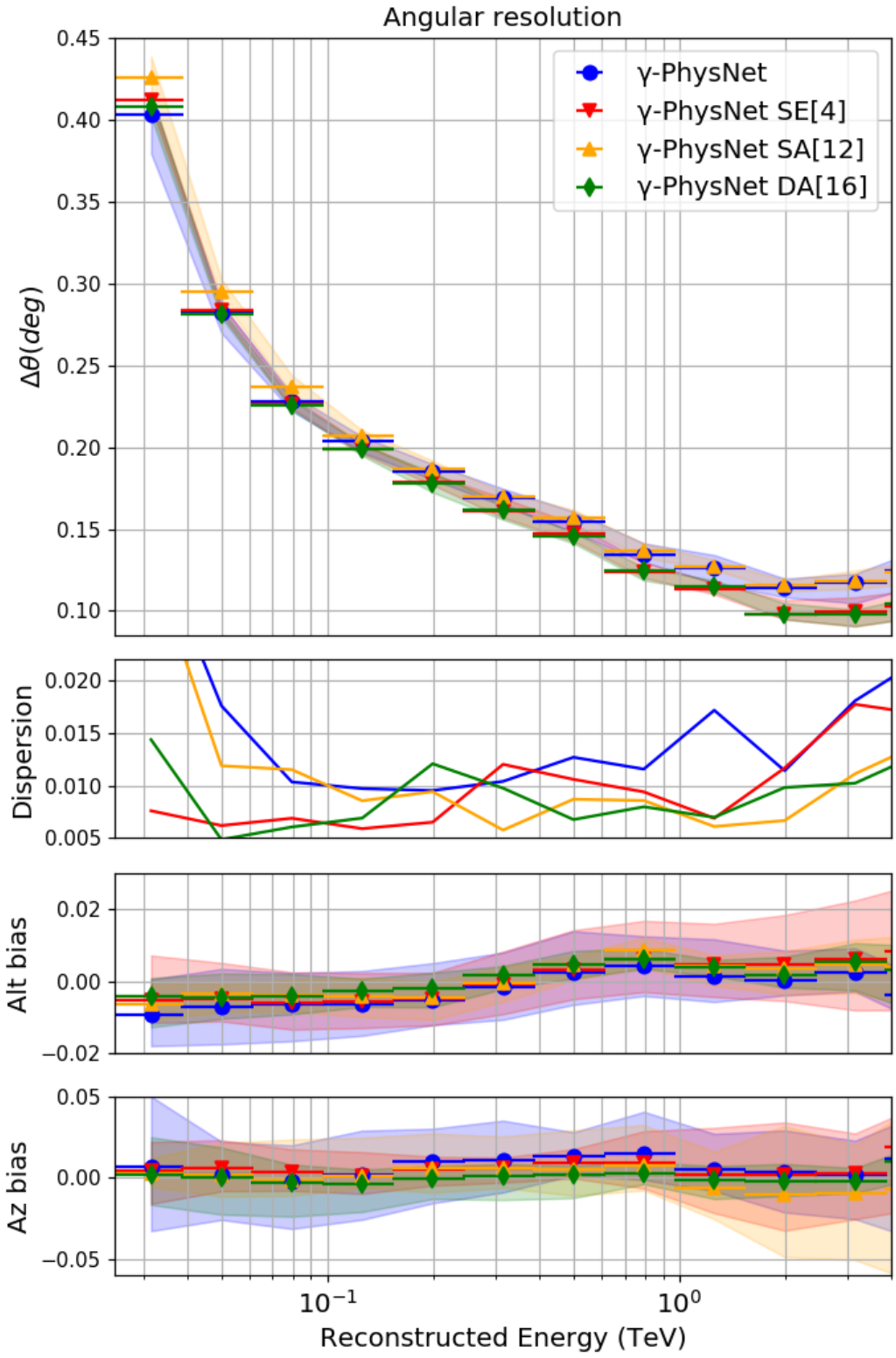


Figure 4.11 – Low cuts. Angular resolution curves of the different attention mechanisms for γ-PhysNet. The surface represents the min/max envelope per bin, and the dots represent the average resolution per bin of the six seeds. The dispersion represents the width of the surfaces per energy bin. The bias plots correspond to the median relative error per energy bin for relatively the altitude and the azimuth reconstructions.

Conclusion. With the low cuts, I aim to keep more events at low energies than the standard analysis. The first result of this experiment with the low cuts is that the sensitivity of all the models, with or without attention, is very good at low energies. Although γ-PhysNet performs single-telescope analysis, it has a better sensitivity than the MAGIC observatory below 200 GeV. In particular, it has competitive results in the range [50, 80] GeV and even below, down to 20 GeV. If confirmed with the future version of the sensitivity library and on the real data, this is particularly interesting for the study of extragalactic sources and gamma-ray bursts. Although CTA is not expected to work in single-telescope mode, these results show that it could be worth considering.

Noteworthy, the models with Squeeze-and-Excitation and dual attention have slightly better results on the energy and direction task at energies above 500 GeV, but it is not the case for the sensitivity. Better spatial and spectral resolutions with similar classification results should lead to a better sensitivity. The computation of the sensitivity involves a cut optimization per energy bin. The analysis of the background rate after this optimization shows a worse background rejection for the models with Squeeze-and-Excitation and dual attention. To investigate this issue, I will, in a future work, compute the sensitivity without the cut optimization.

Finally, on all the tasks to address, all the models have obtained close results, showing the robustness of the γ-PhysNet architecture. In addition, at higher energies, the Squeeze-and-Excitation and the dual attention allow fine-tuning the angular and energy resolutions that are already better than the one of the Hillas + RF method. Besides, all the attention methods probed reduce in a very significant way the variability of the weight initialization. This experiment demonstrates the stability of γ-PhysNet on simulated data. As Squeeze-and-Excitation and dual attention obtain similar results, I would choose the dual attention for γ-PhysNet. Although it is slightly slower, its spatial attention component may help improve the explainability of the model.

4.2.4.2 Performance with the High Cuts

For the energy and direction regression, Figure 4.13 and Figure 4.12 present the results of the different methods in the range 100 GeV to 3 TeV as the selection filters discard most events below 100 GeV.

Direction Reconstruction. On the direction reconstruction task, γ-PhysNet with the Squeeze-and-Excitation and the dual attention mechanisms outperform the other models, in average performance and in dispersion. In particular, they improve the resolution by $0.02°$ on most of the energy range of interest, achieving a resolution of $0.1°$ with a dispersion of $0.01°$. Besides, the model with self-attention has a comparable average resolution with the model without attention, although less spread. Squeeze-and-Excitation and dual attention focus both on feature channel importance. It seems to better benefit the direction reconstruction than pixel-wise attention. Besides, all models have similar altitude and azimuth biases.

Energy Reconstruction. On the energy reconstruction task, all the attention methods probed have a better average performance than the model without attention. Their results are also less spread. In particular, the dual attention mechanism performs clearly better on average, improving the resolution up to 27%. Its dispersion is four times smaller than the one of self-attention. The model without attention spreads ten times more. In addition, all models with attention are less biased than the original γ-PhysNet, especially the model with dual attention.

On the contrary to the direction reconstruction, the energy reconstruction seems to benefit from the spatial attention as the model with self-attention performs better than the model without attention, and the model with dual attention performs slightly better than the one with Squeeze-and-Excitation.

A hypothesis is that the spatial attention helps the model focus on the shower by highlighting the relevant features corresponding to the signal. The HC are highly selective and keep events with large and bright showers that may contain more details than low energy ones. As the total intensity of the shower is related to the primary particle energy, bringing out the shower from the noise may lead to a better reconstruction by the global feature network of γ-PhysNet.

Another hypothesis is that spatial attention helps the model handle better truncated showers. Indeed, in proportion, there are more truncated showers in the dataset filtered with the HC than with the LC. They thus have a greater influence on the model performance with the HC.

In a future work, I will verify these hypotheses by observing the output of the spatial attention path of the dual attention.

Classification. Table 4.5 shows that all three attention methods and the model without attention have similar results on the classification task.

Sensitivity. As illustrated in Figure 4.14, all the models probed have similar sensitivity curves and dispersion. However, the intensity cut is too high and the models have no sensitivity below 150 GeV. As it is, this configuration has a limited interest for the physics.

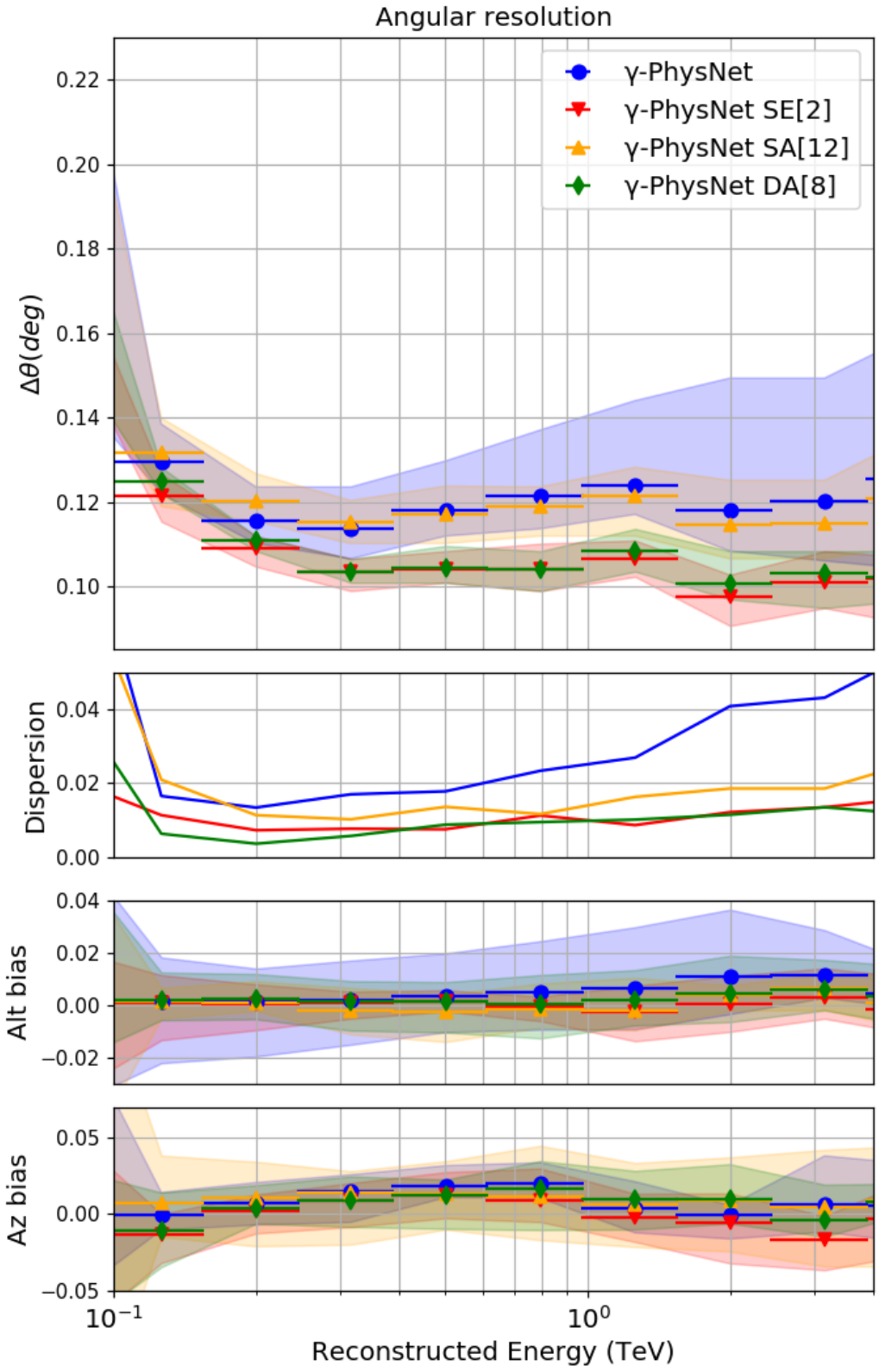


Figure 4.12 – High cuts. Angular resolution as a function of reconstructed energy in the LST energy range updated by the cuts. Comparison of the different attention mechanisms for γ-PhysNet. The surface represents the min/max envelope per bin, and the dots represent the average resolution per bin of the six seeds. The dispersion represents the width of the surfaces per energy bin. The bias plots correspond to the median relative error per energy bin for relatively the altitude and the azimuth reconstructions.

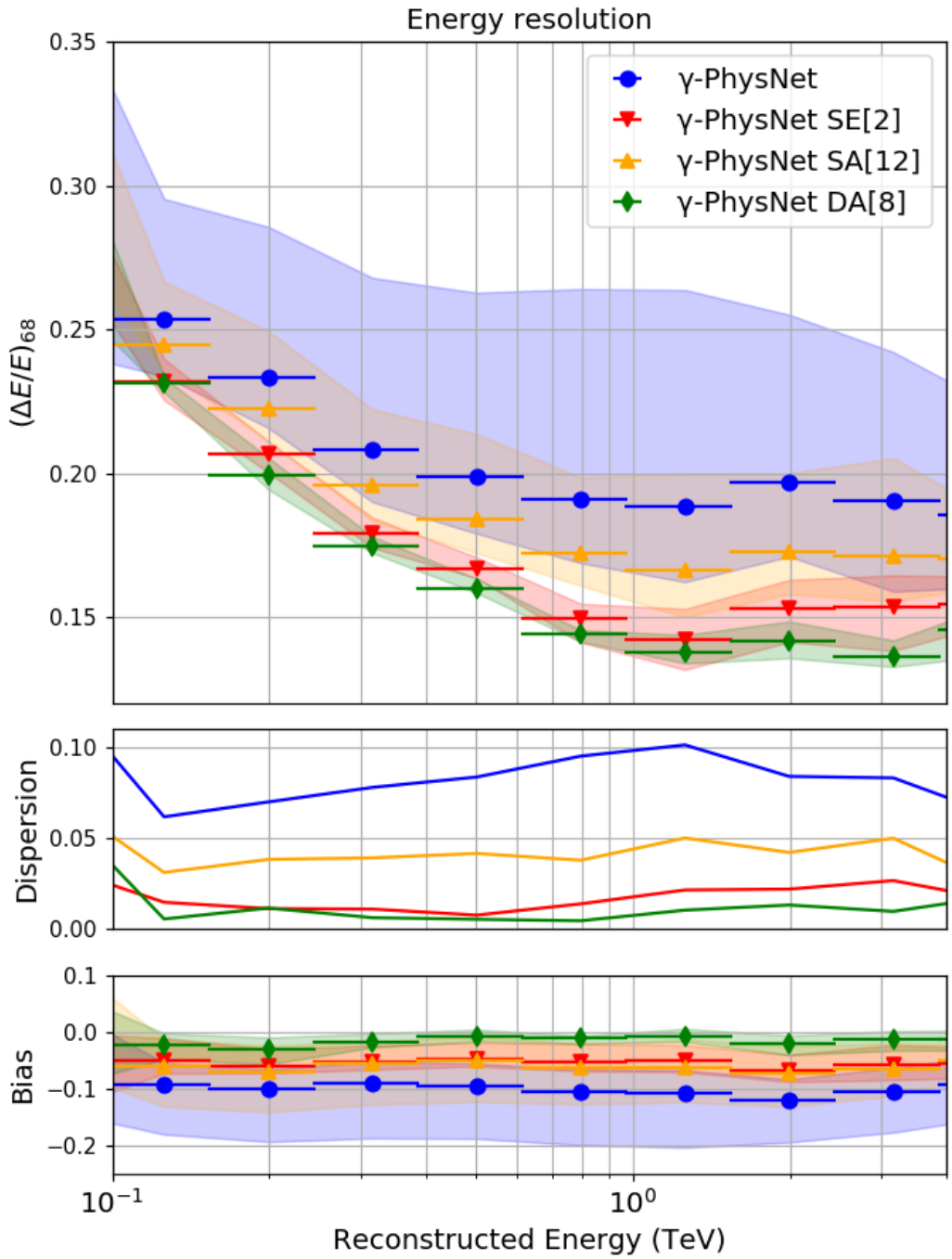


Figure 4.13 – High cuts. Energy resolution as a function of reconstructed energy in the LST energy range updated by the cuts. Comparison of the different attention mechanisms for γ-PhysNet. The surface represents the min/max envelope per bin, and the dots represent the average resolution per bin of the six seeds. The dispersion represents the width of the surfaces per energy bin. The bias plot corresponds to the median relative error per energy bin.

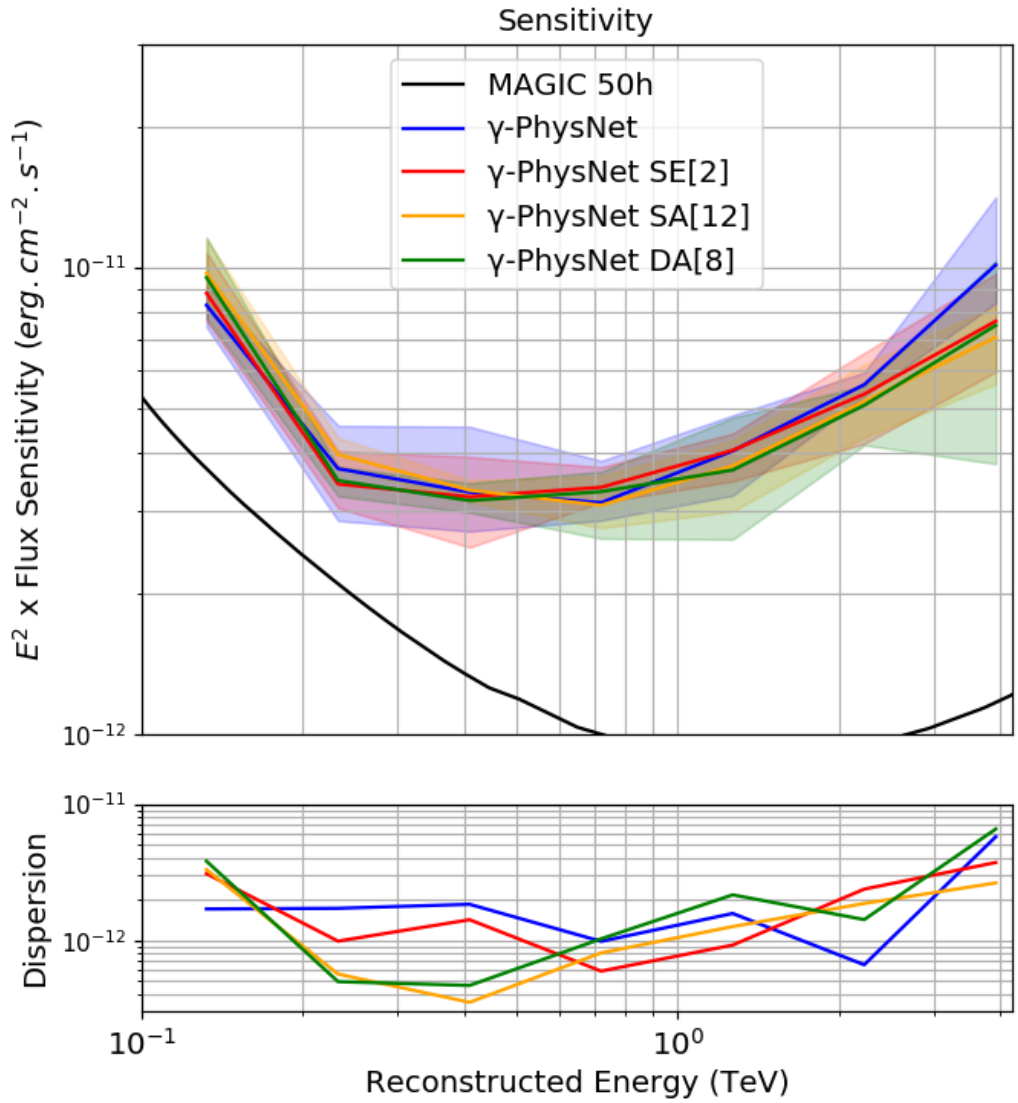


Figure 4.14 – High cuts. Sensitivity curves, the lower the better. Comparison of the different attention mechanisms for γ-PhysNet. The surface represents the min/max envelope per bin, and the dots represent the average sensitivity per bin of the six seeds. The dispersion represents the width of the surfaces per energy bin. For reference, the black line corresponds to the performance of the MAGIC observatory in stereo mode.

Table 4.5 – High Cuts. AUC, precision and recall of the gamma/proton classification task for the different models.

Model	AUC	Precision	Recall
γ-PhysNet	0.990±0.001	0.980±0.002	0.982±0.002
γ-PhysNet SE[2]	0.991±0.001	0.980±0.001	0.983±0.002
γ-PhysNet SA[12]	0.989±0.001	0.980±0.001	0.981±0.002
γ-PhysNet DA[8]	0.991±0.001	0.980±0.002	0.984±0.002

Conclusion. As with the low cuts, all the models exhibit similar performance on the classification task and have comparable sensitivity. Again, the models with Squeeze-and-Excitation and dual attention perform better on the energy and direction reconstruction tasks. In particular they achieve a very good angular resolution of 0.1° for almost all the energy bins between 100 GeV and 4 TeV. However, the interest of single-telescope analysis is low in this energy range as stereo analysis will achieve better results with sufficient sensitivity. Moreover, the cut on the intensity is too strong and sacrifices the sensitivity for the angular resolution. This configuration has then a limited interest for the physics, but allows to know the best possible angular resolution with the model in single-telescope analysis.

Noteworthy, although the amount of data has dramatically decreased with the HC, adding more parameters to the network with attention does not lead to overfitting. Moreover, the model is still able to generalize, even without early stopping, *i.e.*, the duration of the experiments being the same as for the LC.

4.2.4.3 Performance with the Mid Cuts

The experiment with the mid cuts is detailed in Appendix D. As for the other selection cuts, all models have a similar sensitivity curve while the Squeeze-and-Excitation and the dual attention methods perform slightly better on the energy and direction reconstructions at high energies (above 1 TeV). I observe the same contradiction between the improvement of energy and angular resolutions and the absence of benefit for the sensitivity curve.

4.3 Discussions

4.3.1 Computational Cost

In the context of CTA data analysis, the computation cost is crucial as all the data will be reprocessed every year to benefit from the improvements of the analysis models. For LST images, the whole γ-PhysNet, implemented with the PyTorch machine learning library and indexed convolutions, has 2.6×10^6 parameters. Although it has not yet been optimized for production, γ-PhysNet has an inference rate of the order of the telescope trigger rate. The results presented in Table 4.6 only take into account the inference time of the model. The cost of the multi-task block is negligible compared to the one of the backbone as γ-PhysNet has an inference rate only 0.01 kHz lower than the ResNet. The multi-task approach allows reducing the full event reconstruction by a factor of 3. The Squeeze-and-Excitation and the dual attention methods slightly lower the efficiency of the model by respectively about 0.5% and 1.4%. Noteworthy, for these methods the reduction ratio has no impact on the inference rate. The self-attention method is way more costly and divides the performance by 1.8.

The Hillas + RF method, that is neither optimized, has an inference rate 1.3 times higher than γPhysNet with dual attention. However, methods exist to optimize the neural network efficiency for production. With the help of pruning and quantization, also named network compression, we can expect to reduce the computation time by a factor ranging from 3 to 4 [Han et al., 2016, Luo et al., 2017, Zhao et al., 2019]. Although similar approach can be used for the Hillas + RF method, it may be more complicated. Indeed, on the contrary to a neural network that is composed of a single algorithm, the Hillas + RF is a toolchain for which every algorithm has to be optimized with a specific strategy.

Table 4.6 – Inference rate of the different models compared. Neural network rates are measured on an NVIDIA V100 GPU and Hillas + RF on one core of an Intel Xeon.

Model	Inference rate
Hillas + RF	5.64 kHz
ResNet-56	4.49 kHz per task
γ-PhysNet	4.48 kHz
γ-PhysNet SE	4.46 kHz
γ-PhysNet SA	2.52 kHz
γ-PhysNet DA	4.42 kHz

4.3.2 Contribution of γ-PhysNet to Gamma Astronomy

The comparison between γ-PhysNet and the widespread Hillas + RF method presented in Section 4.2.3 shows that deep multi-task learning augmented with attention dramatically improves the performance of IACT data single-telescope analysis, especially for the direction reconstruction task.

Improvements in energy resolution will allow the production of more detailed spectrum, bringing more constraints on sources modeling. Improving the angular resolution and the classification will both improve the signal-to-noise ratio, thus allowing the detection of fainter sources in a significant way. Studies of extended sources at very high energies are quite recent. However, the studies made by H.E.S.S. show extended emissions corresponding to angular separation going from 0.05° (of the order of the H.E.S.S. angular resolution at energies > 5 TeV) [Aharonian et al., 2019] to 0.3° [Hoppe et al., 2009]. These values show that the gains obtained in angular resolution, even compared to the single task ResNet-56 (up to 0.08°), could make the difference between observing a point source and an extended source. This then allows for morphological studies, bringing important insights on the physics of these sources. Moreover, the gain in performance below 1 TeV is relevant to the search for dark matter [Bergström, 2013] and the study of gamma-ray bursts [Inoue et al., 2013].

4.3.3 Attention Mechanisms

Globally, the models with attention have similar performance compared to the bare γ-PhysNet on all the tasks. Attention allows slightly improving the results, acting as a fine-tuning strategy. This stresses the stability of this architecture.

Besides, for all the configurations of selection cuts explored, augmenting γ-PhysNet with Squeeze-and-Excitation or dual attention slightly improves both energy and direction reconstruction at higher energies. Noteworthy, the self-attention mechanism, although the more complex, constantly underperforms compared to the Squeeze-and-Excitation and the dual attention methods. γ-PhysNet benefits more from learning channel dependencies than learning long-range spatial ones. The useful information is rather localized in the image and the pixels that are far from the shower are generally noise (the farness depends on the size of the shower). Learning long-range dependencies between pixels on the whole image, as does self-attention, might not help the model better reconstruct the event. To verify this hypothesis, I could explore local self-attention [Parmar et al., 2019] in a future work.

Then, all the attention methods help reduce the variability of the results and thus improving the robustness of the models.

Finally, as we will see later in this chapter, interpretability is crucial to understand and trust the decision of the model. Observing the output of attention layers could be an interesting complement to the methods described in Section 4.4. It could also be useful to monitor the model response to real data or to changing data due to sensor deviation or different sky conditions.

4.3.4 Comparison of the Selection Cuts

To compare the effect of the selection cuts, I select the "best" model per selection configuration, as a trade-off between all the performance indicators. The sensitivity curves, the angular and energy resolution are shown in Figure 4.15.

With the low cuts, I aim to optimize the sensitivity of the model, in particular at low energies. Such a model could address the study of extragalactic sources, realize the spectral analysis of well-known sources or search for new sources. γ-PhysNet obtains very good results in single-telescope analysis. Its sensitivity threshold is close to the

limit of the LST and its sensitivity below 200 GeV is competitive with MAGIC, a stereo observatory.

On the other hand, with the high cuts, I aim to optimize the spatial resolution of the model. γ-PhysNet has a very good angular resolution (0.1°) in a single-telescope context. The classification task also benefits from these cuts. It was expected as the gamma/proton separation and the direction reconstruction rely on local and spatial parameters, and the high cuts retain larger and brighter showers. However, the intensity cut for this configuration is too strong and sacrifices the sensitivity. It is then of limited interest for the physics.

The mid cuts are an interesting trade-off. The sensitivity of γ-PhysNet is also better than the one of MAGIC below 200 GeV while its angular resolution improves compared to the low cuts. The mid cuts are a good starting point to search for an intensity cut that optimizes the direction reconstruction without tearing down the sensitivity.

Noteworthy, the performance on the energy reconstruction is similar for the three sets of cuts, although I would expect a better resolution with a stronger selection. This emphasizes the robustness of the γ-PhysNet architecture on the energy reconstruction task, but also its limit.

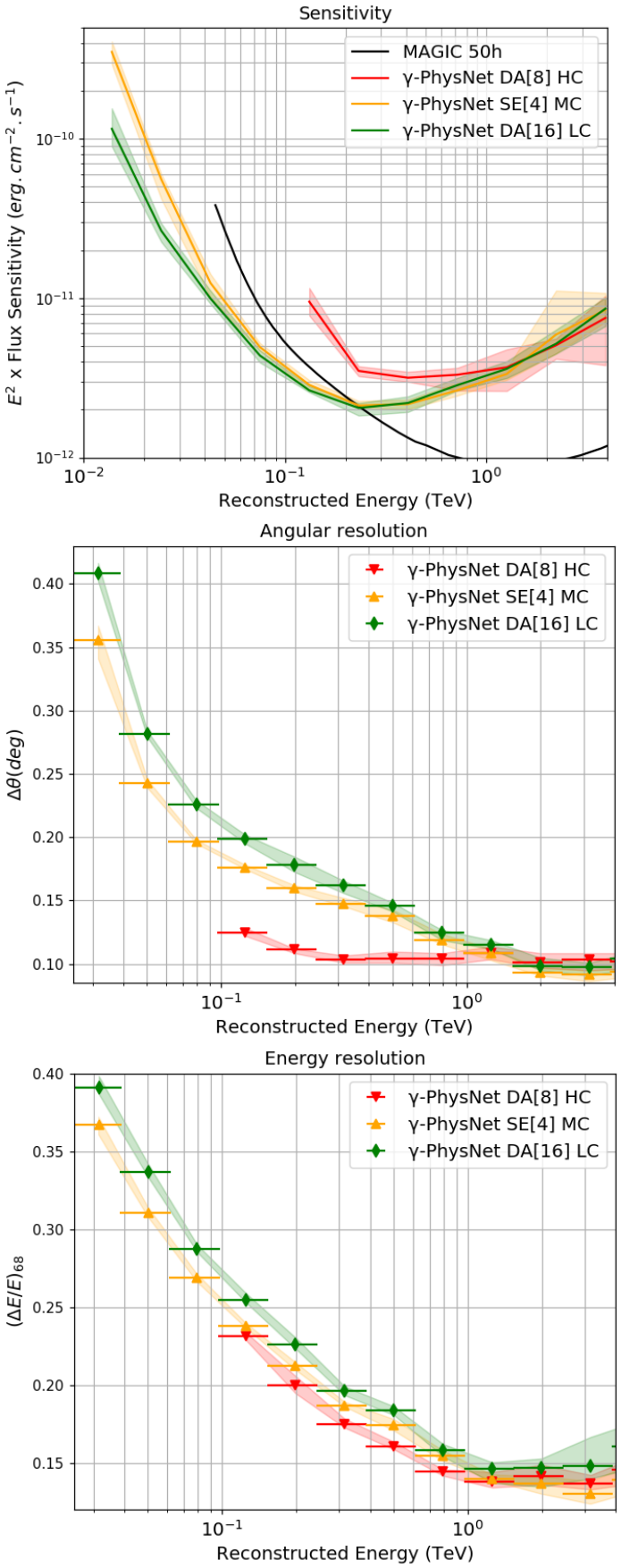


Figure 4.15 – Comparison between the selection cuts.

4.3.5 Comparison to Standard Methods

We have seen in Section 4.2.3 that γ-PhysNet outperforms the widespread Hillas + RF method using a common set of cuts. On another hand, the experiments presented in Section 4.2.4 show that γ-PhysNet augmented with attention obtains a very interesting performance with less selective cuts (the low cuts), especially at energies below 200 GeV. To further evaluate the interest of the proposed model, I compare the performance of γ-PhysNet DA[16] and the Hillas + RF method, both trained with the low cuts. Figure 4.16 shows that γ-PhysNet clearly outperforms the standard method below 600 GeV. Its sensitivity is more than twice better below 100 GeV. In this energy range, my model also improves the angular resolution by 0.15° to 0.3° and the energy resolution by 0.1. The improvement in sensitivity is especially interesting as it corresponds to the LST energy range of interest in the context of single-telescope analysis.

For reference, I also compare in Figure 4.17 the performance of γ-PhysNet DA[16] trained with the low cuts to the state-of-the-art observatory H.E.S.S. II for IACT data mono-telescope analysis. Noteworthy, γ-PhysNet's results are consistent with ImPACT, a template-based method, for both loose and safe cuts. Although not optimized, γ-PhysNet is more than **800 times** faster than ImPACT [Parsons et al., 2016], processing 4,420 events per second. It is then more applicable for the yearly reprocessing of CTA data, and could be a candidate for a real-time analysis with optimization.

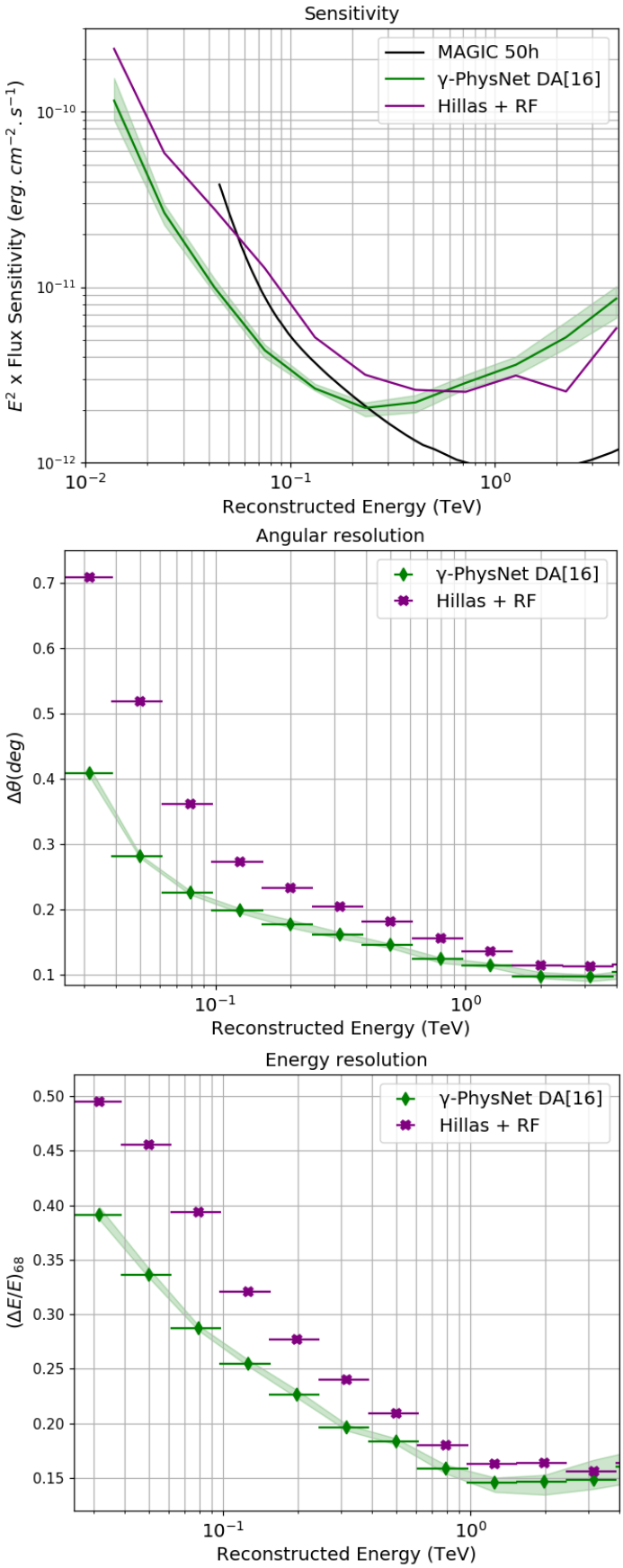


Figure 4.16 – γ-PhysNet with the LC, comparison with Hillas + RF.

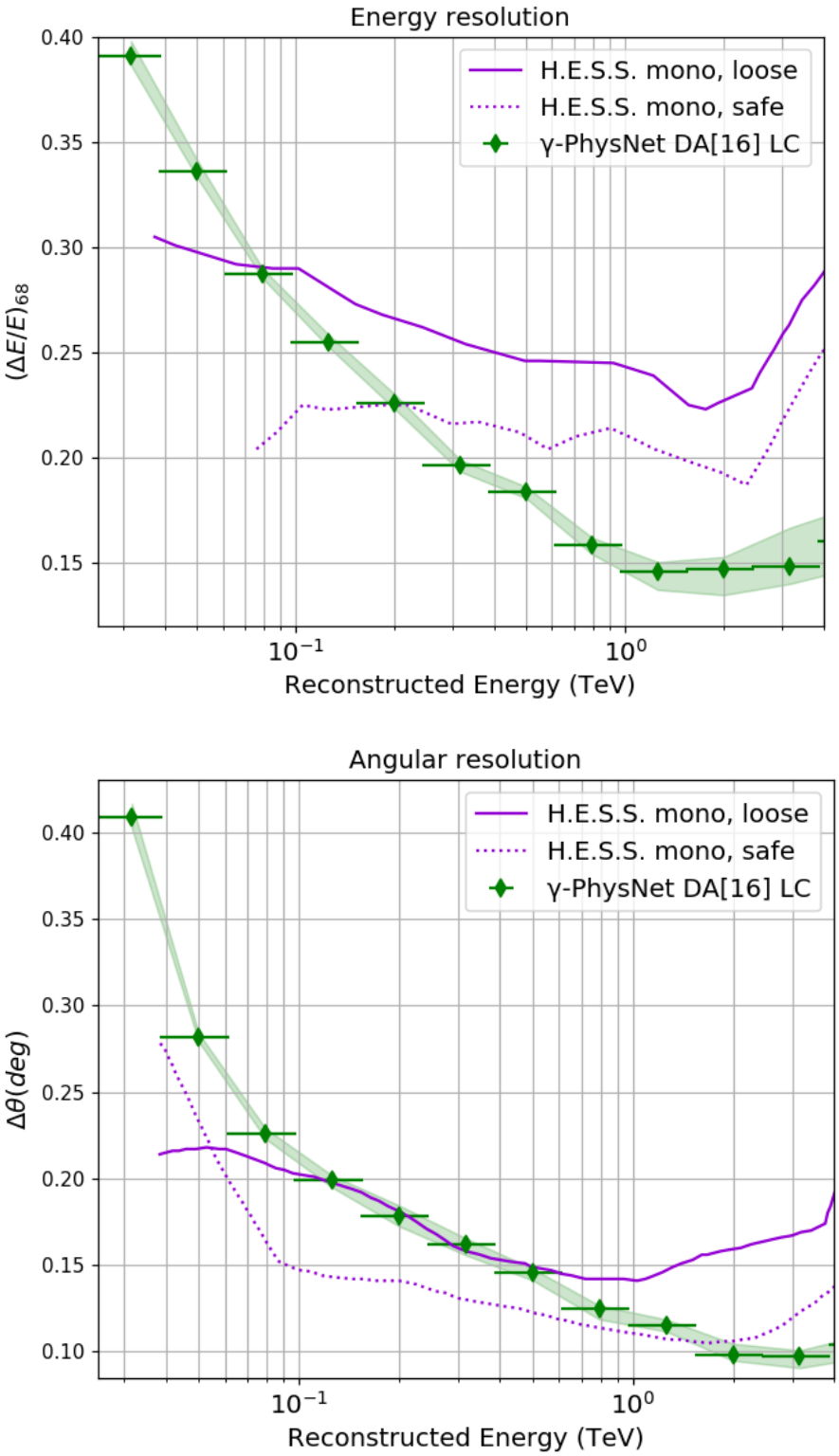


Figure 4.17 – γ-PhysNet results are consistent with the state-of-the-art IACT H.E.S.S. II mono (ImPACT analysis, source [Parsons et al., 2016]). Safe and loose represent two sets of cuts.

4.4 Opening the Black Box of γ-PhysNet

Since the first successes of deep learning algorithms, explainability and interpretability of model's decisions have been of great concern. Progressing to more transparency and accountability is crucial as deep learning techniques spread more and more in domains such as healthcare, insurance or autonomous vehicles.

Interpretability is related to the ability to predict how a change in the input of the model will affect its output. While traditional expert systems are highly interpretable, deep neural networks rely on a greater abstraction to achieve better results, giving them their sense of *black box*. However, they can be constrained into a more interpretable functional form [Frosst and Hinton, 2017]. Frosst and Hinton propose to train a soft decision tree in order to learn the function discovered by the neural network. The predictions of the neural network are used to label training data and unlabeled data to build the training set of the soft decision tree. On the other hand, explainability is related to the comprehension of the internal mechanics of neural networks. Recently the deep learning community has put an increasing effort on opening the black box. Some methods explore the role of individual neurons or linear combination of units through ablation [Zhou et al., 2018, Morcos et al., 2018a] or optimization [Olah et al., 2017]. Others estimate the importance of input features for a particular output activation. They produce saliency maps [Springenberg et al., 2015, Cao et al., 2015, Srinivas and Fleuret, 2019] or heatmaps [Selvaraju et al., 2017] for network visualization. As a complementary method, we can also compute the receptive field (or field of view) [Luo et al., 2016, Le and Borji, 2017] of the network or of a particular unit. It represents the region in the input taken into account to produce the output of the model (or the unit).

A better knowledge of the working of the model could help find insights to improve its performance. In this section, I first compute the receptive field of γ-PhysNet and compare it to LST image size. Then I apply visual explanation methods to well and badly reconstructed events in order to understand how my architecture produces a decision and to verify the consistency of its behavior with the physical phenomenon.

4.4.1 Receptive Field

To understand the behavior of the model, we are interested in mapping the region in the input that produces a particular output or intermediate feature. We define the size of this region as the *receptive field* of the feature. This concept is important for a variety of applications, to ensure that all the relevant information is used to produce the model's decision. In object detection for example, if the receptive field is too small, the model might not be able to detect large objects.

In fully connected neural networks, the value of each feature depends on the entire input. Thus, the receptive field is the input size. On the contrary, in convolutional neural networks each feature depends on a local neighborhood of the input (see Section 3.1.1 for details).

Receptive Field Computation. To describe the computation of the receptive field, we consider 1D input data and a single path fully convolutional network with L layers, $l = 1, 2, \ldots, L$. The feature map f_l is the output of the l-th layer, the input image being f_0 and the output of the network f_L. Each layer l has three parameters:

- k_l: the kernel size,
- s_l: the stride,
- p_l: the padding.

We define r_l as the receptive field of the output feature map f_L with respect to the feature map f_l. r_l represents the size of the region in f_l that contributes to one feature in the output f_L. Knowing r_l, we obtain r_{l-1} with:

$$r_{l-1} = s_l \cdot r_l + (k_l - s_l) \tag{4.2}$$

Reference [Araujo et al., 2019] solved this recurrent equation and obtained the following solution:

$$r_0 = \sum_{l=1}^{L} \left((k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1 \tag{4.3}$$

For multiple-dimension data, we have to compute the receptive field along every dimension.

Most of the state-of-the-art models, such as the ResNet-56 backbone of γ-PhysNet, are multiple-path networks. Equation 4.3 can be used to compute each path. When the network is aligned [Araujo et al., 2019], *i.e.*, the center of the receptive field of every path is the same, as is the case for the ResNet-56, the receptive field of the whole model is the largest among all paths.

γ-PhysNet Receptive Field. In the context of hexagonal pixel images and Indexed Convolutions, hexagonal convolution kernels can be represented as circles as illustrated in Figure 4.18.

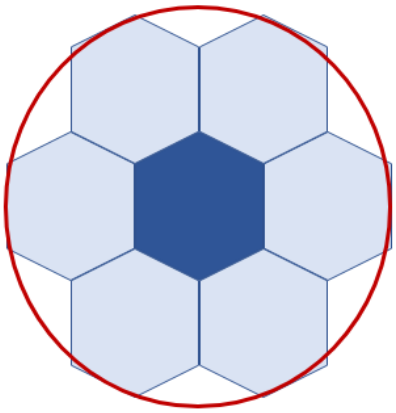


Figure 4.18 – Hexagonal convolution kernels can be represented as circles. In the above example, the hexagonal blue pixels represent a kernel of the first nearest neighbors, corresponding to $k_l = 3$. It can be expressed as the red circle of diameter three pixels.

To compute the receptive field of the convolutional part of γ-PhysNet, we can express the kernel size k_l in Equation 4.3 as the diameter of the hexagonal kernel of layer l. We need to compute the receptive field only once, and the latter is a disc. The result for the original version of the architecture (without attention) is $r_0 = 249$ pixels. The details of

the computation are presented in Appendix E. As illustrated in Figure 4.19, it is much larger than the input images. To design γ-PhysNet, I chose the ResNet-56 CIFAR-10 version as a backbone because LST images are rather small. They are composed of 1855 pixels and can be bounded by a box of 46×54 pixels, comparable to CIFAR image size (32×32). From the receptive field point of view, the model is much deeper than necessary. On the other hand, the great depth of the network allows for a fast decomposition of the input with fewer parameters (smaller kernel size and less neurons) to produce the relevant latent representation of the data. In a future work, I will explore depth reduction for γ-PhysNet, seeking for a faster network without loss of performance. The objective is to comply with CTA computation time requirements for both real-time analysis and yearly data reprocessing. Preliminary experiments will be presented in Section 6.2.2.

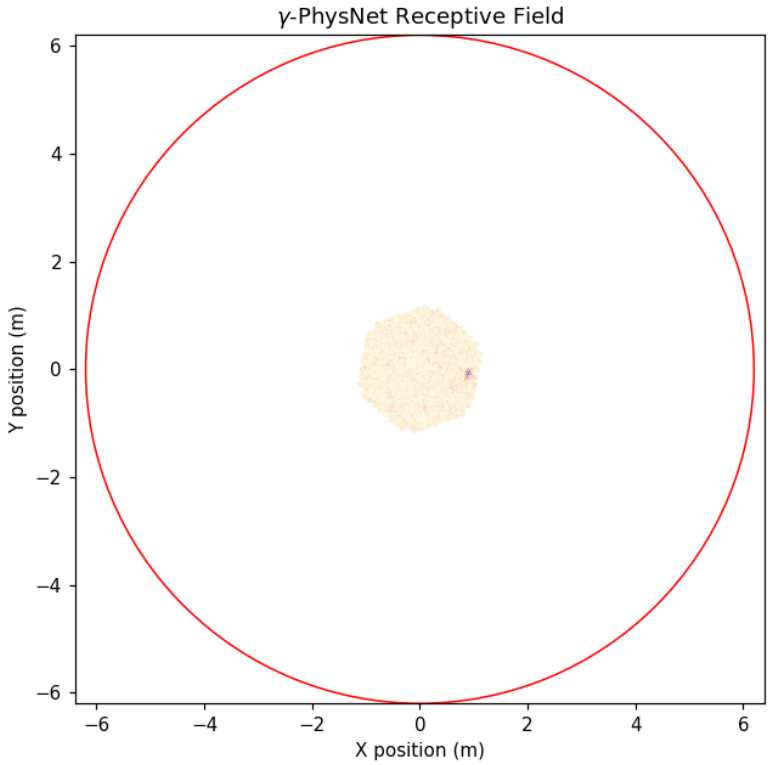


Figure 4.19 – Receptive field of γ-PhysNet. The red circle represents the receptive field in the camera coordinate system. It is much larger than the input image (in the center).

4.4.2 Visual Explanation with Grad-CAM

To place trust in deep neural networks, we must understand their internal process. Visual explanation methods help analyze network decision by highlighting important pixels in the input image. Grad-CAM [Selvaraju et al., 2017] is a highly class-discriminative localization method that produces a heatmap of the region that is the most relevant for the model's decision. It is applicable to any network structure and to analyze any output task (*e.g.*, classification, image captioning, visual question answering). The heatmaps are computed based on the last convolutional layer. In the context of multi-task learning, Grad-CAM is then especially adapted to hard parameter sharing networks that share a convolutional encoder between all tasks. In the case of γ-PhysNet, the last convolutional

layer is also the last common one. In this section, I present first steps to apply Grad-CAM to γ-PhysNet.

Grad-CAM procedure. As illustrated in Figure 4.20, it combines the last feature maps produced by the convolutional part of the network using the gradients of the task output we are interested in. The last convolutional layers represent the best trade-off between high-level semantics, that is crucial for a meaningful explanation, and spatial information. The gradient signal, with respect to the retained feature maps, captures the neuron importance for the task we want to explain.

Consider a neural network composed of a convolutional backbone and a multi-task block made of fully connected layers. The last convolutional layer (including the ReLU activation) produces A, comprising k feature maps. We first compute the neuron importance weights α_k^t corresponding to task t:

$$\alpha_k^t = \frac{1}{N} \sum_{i=0}^{N} \frac{\partial y^t}{\partial A_i^k} \tag{4.4}$$

with y^t the output of task t, N the number of pixels in feature map A^k. We then combine the feature maps of A through a sum weighted by the α_k^t:

$$L_{Grad-CAM}^t = ReLU\left(\sum_k \alpha_k^t A^k\right) \tag{4.5}$$

The *ReLU* operation allows keeping only features that have a positive influence on the task of interest. $L_{Grad-CAM}^t$ is a coarse heatmap, with the same size of A, and needs to be scaled up to the input size for a better readability. In the case of γ-PhysNet analysis, the scaling up is achieved with a custom method similar to the bilinear interpolation but adapted to the specific pixel grid and image shape of the data.

Grad-CAM is very efficient as it only requires a forward and a partial backward pass.

Grad-Cam for γ-PhysNet. To analyze our architecture, I have implemented Grad-CAM with the support of Indexed Convolution and hexagonal pixel images. Because of the limited time of the PhD thesis, the analysis is restricted to the behavior of γ-PhysNet DA[16] trained with the low cuts. Besides, I expect a similar behavior with the other cuts. As an illustration, I first select good and bad events among the data, based on the prediction of the model. The good events selected for this analysis are:

- a gamma point-like event with a classification score of 0.999
- a proton event with a classification score of 1.0
- a gamma point-like event with an energy resolution of 0.013
- a gamma point-like event with an angular resolution of 0.02°

while the bad events selected are:

- a gamma point-like event with a classification score of 0.375
- a proton event with a classification score of 0.12

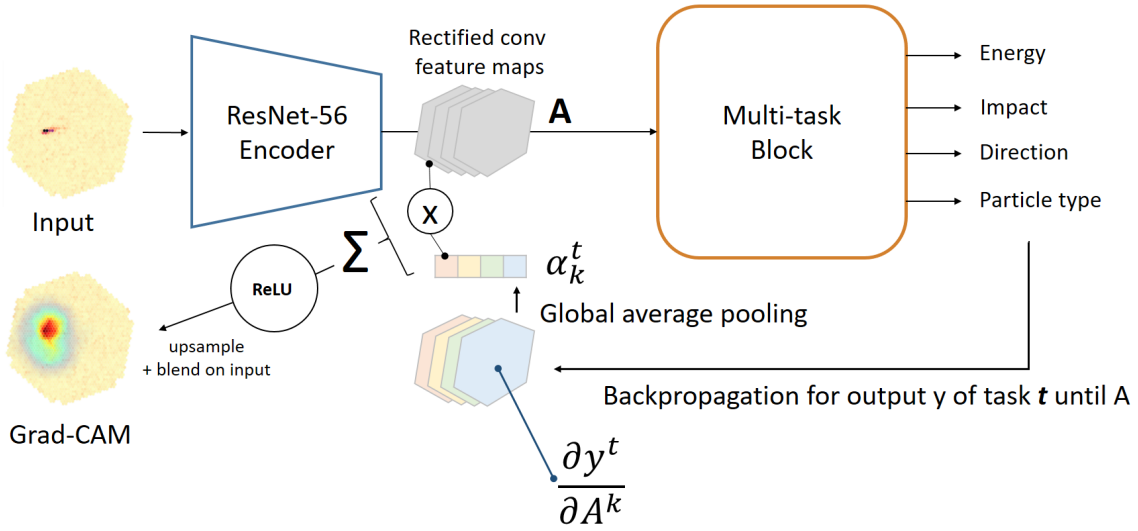


Figure 4.20 – Grad-CAM method applied to γ-PhysNet. To produce the Grad-CAM heatmap of the input for output y of task t, we first compute the gradients of y^t with respect to the feature maps A produced by the backbone encoder. These gradients are averaged and multiplied to the corresponding feature maps. The so weighted feature maps are summed pixel-wise and passed through a ReLU to produce the heatmap.

- a gamma point-like event with an energy resolution of 2.335
- a gamma point-like event with an angular resolution of 0.433°

I then compute the heatmaps of the model for these events. They are presented in Figures 4.21 to 4.28. The first columns represent the charge (in photoelectrons) and the temporal information (the position of the charge peak in the video sequence) of the events (see Chapter 1 for details). The other columns represent the normalized Grad-CAM heatmaps of the outputs considered blended over the input data.

Classification of a Gamma Event. Figure 4.21 represents the Grad-CAM heatmaps of a well-classified gamma event and Figure 4.22 of a badly classified gamma event. For both events, the important pixels highlighted by Grad-CAM are located in the area of the shower. For the well-classified gamma event, the model seems to focus on one extremity of the shower while for the badly classified one the maximum of the heatmap seems outside the shower, although the significant pixels of the heatmap seem to surround the shower area. From a physical point of view, focusing on the contour of the shower is interesting to separate gamma and proton events. However, this might not be sufficient as the charge pattern of the shower also provides information on the particle type.

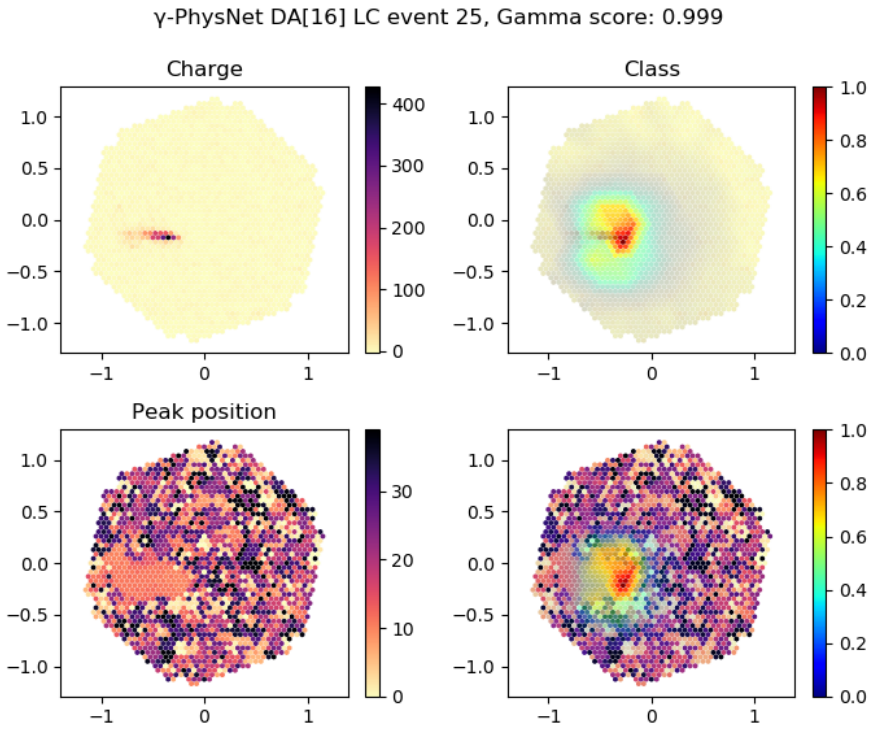


Figure 4.21 – Grad-CAM applied to γ-PhysNet for the task *classification as a gamma*, well-reconstructed event.

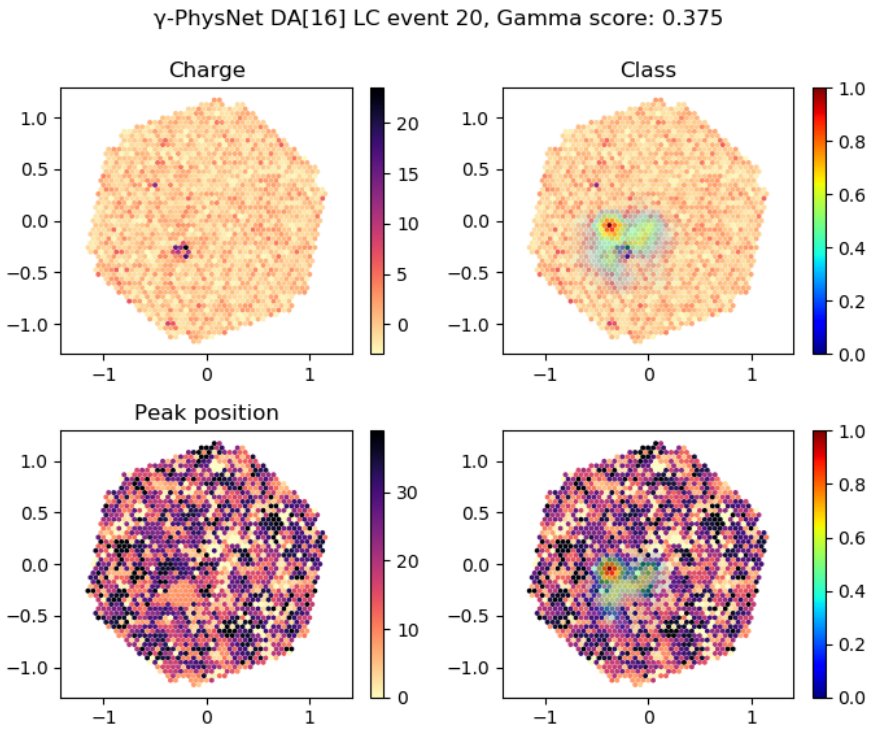


Figure 4.22 – Grad-CAM applied to γ-PhysNet for the task *classification as a gamma*, badly reconstructed event.

Classification of a Proton Event. Figure 4.23 represents the Grad-CAM heatmaps of a well-classified proton event and Figure 4.24 of a badly classified proton event. As for the gamma classification, the important pixels highlighted by Grad-CAM are located in the area of the shower. However, for the well-classified proton event, the model seems to focus only on a part of the shower while for the badly classified one the heatmap seems to be well centered on the shower. The misclassified proton is a faint event and corresponds to a hard case. From a human expert point of view, it could look like a gamma.

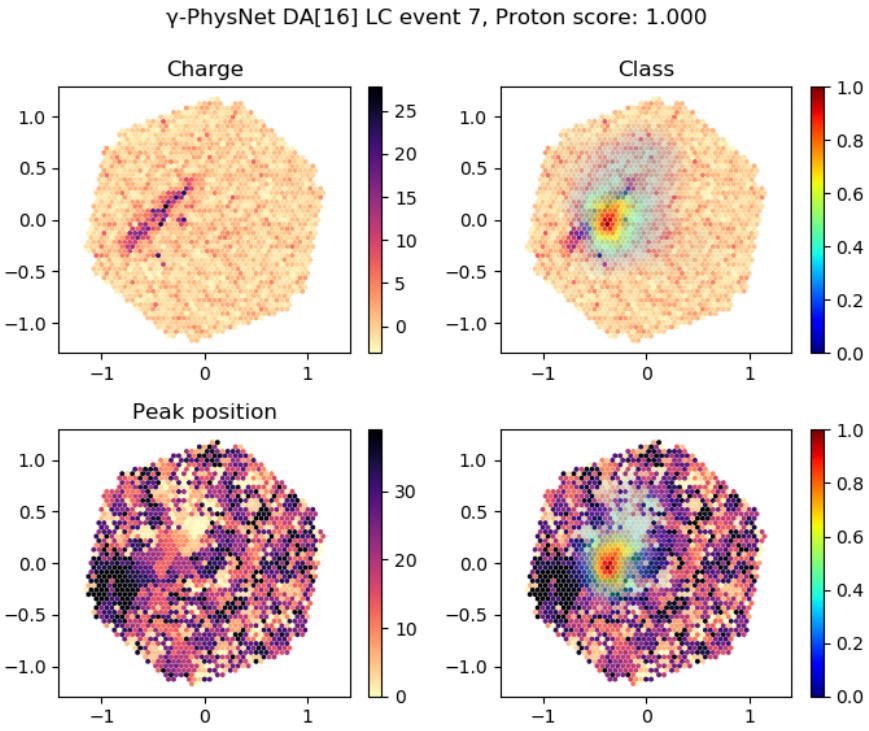


Figure 4.23 – Grad-CAM applied to γ-PhysNet for the task *classification as a proton*, well-reconstructed event.

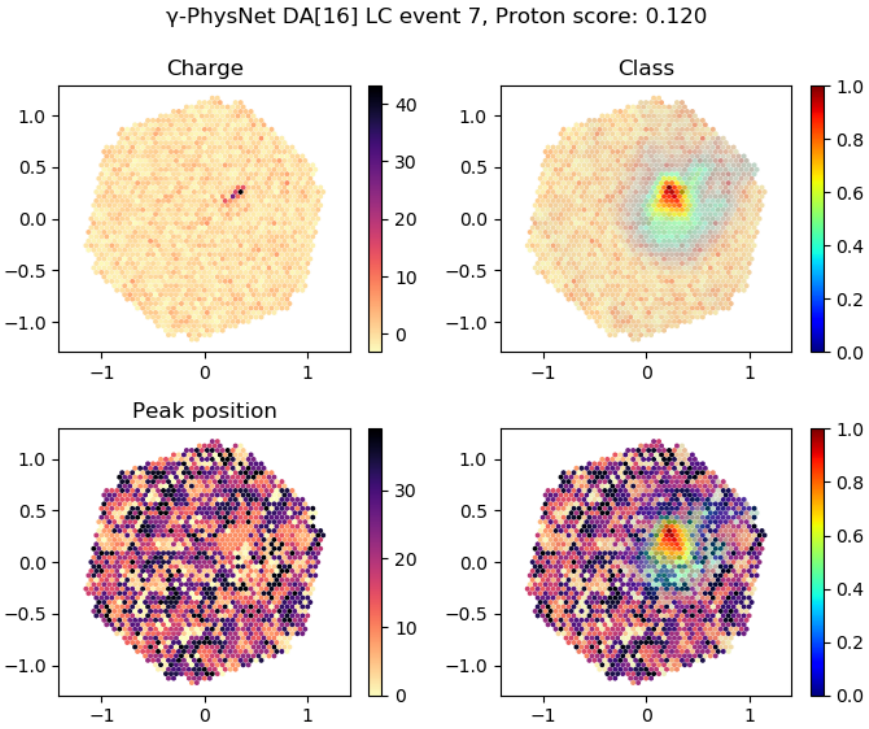


Figure 4.24 – Grad-CAM applied to γ-PhysNet for the task *classification as a proton*, badly reconstructed event.

Energy Reconstruction. Figure 4.25 represents the heatmaps of a gamma event with well-reconstructed energy and Figure 4.26 of a gamma event with badly reconstructed energy. As for the classification task, the highlighted pixels are located in the area of the shower. For the well-reconstructed gamma, the observation of the temporal information shows that the model seems to focus on the tail of the shower. Surprisingly, the maximum of the heatmap is not situated on the brightest pixels of the shower. For the badly reconstructed gamma, the model seems to focus on pixels outside the shower. However, this area might contain signal pixels that are invisible to the human eye. Information about true signal pixels could help refine this analysis. Besides, this is a faint event and it is expected to be a hard case.

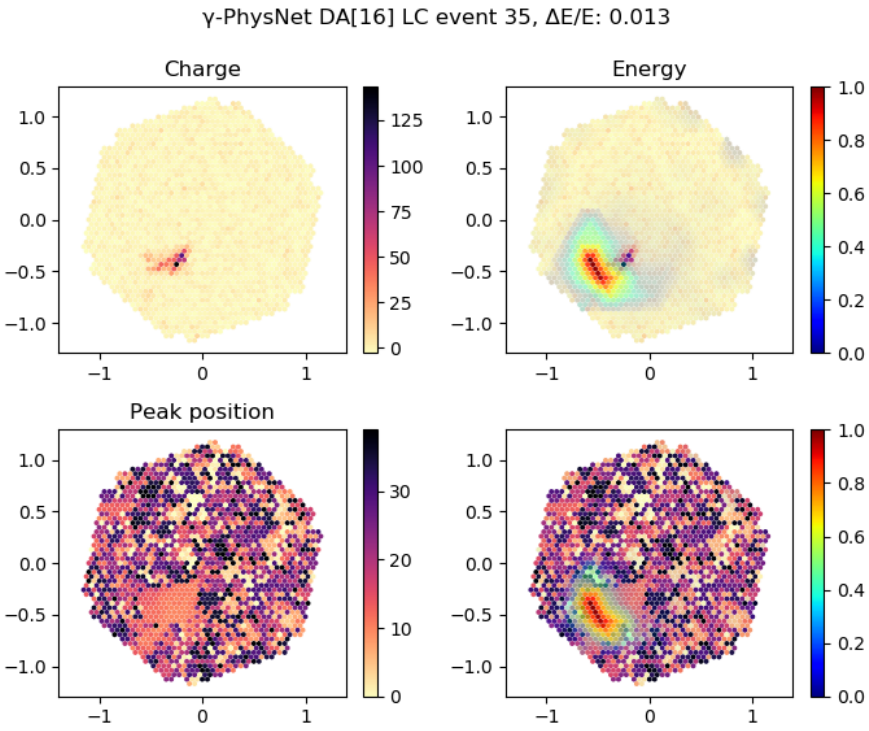


Figure 4.25 – Grad-CAM applied to γ-PhysNet for the task *energy reconstruction*, well-reconstructed event.

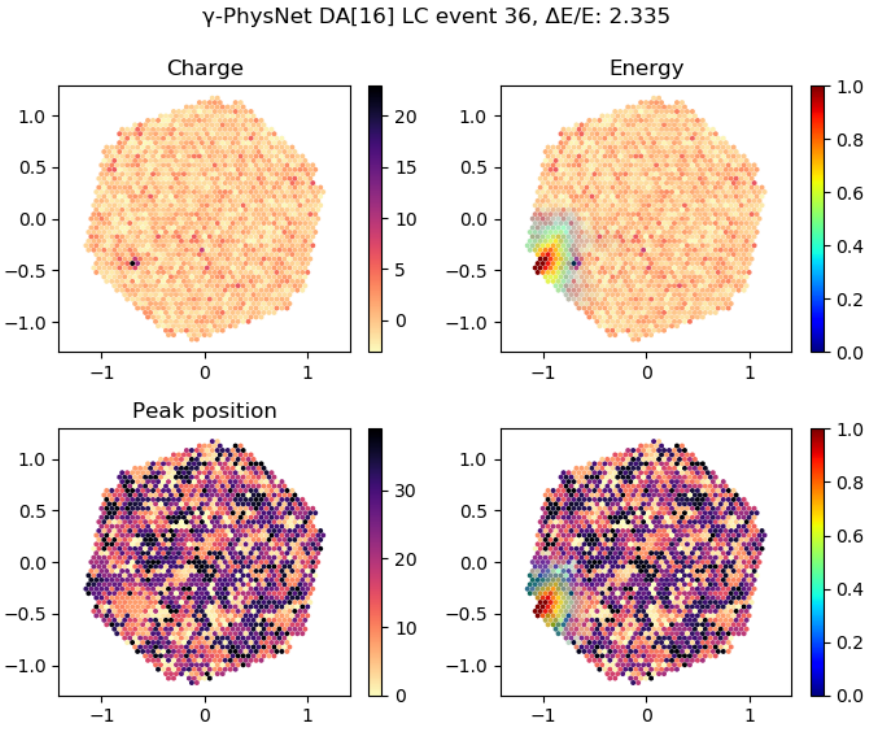


Figure 4.26 – Grad-CAM applied to γ-PhysNet for the task *energy reconstruction*, badly reconstructed event.

Direction Reconstruction. Figure 4.27 shows the heatmaps of a gamma event with well-reconstructed direction and Figure 4.28 of a gamma event with badly reconstructed direction, in terms of angular separation. In both cases, the highlighted pixels are located in the region of the shower. As for the energy reconstruction, the brightest pixels of the heatmaps seem to be located on the tail of the shower for both altitude and azimuth. The badly reconstructed event is surprising as it is a bright and relatively large shower. The network seems to focus on pixels besides the shower, especially for the azimuth reconstruction. However, there might also be some signal in this area that we cannot see. This stresses the need of knowing which pixels contain true signal. To force the network focus on the signal and possibly improve the reconstruction I could add the detection of the signal or its semantic segmentation as another task. However, I expect the attention modules to learn the position of the shower in the image. In a future work, to verify this I will analyze the spatial attention maps of the dual attention modules.

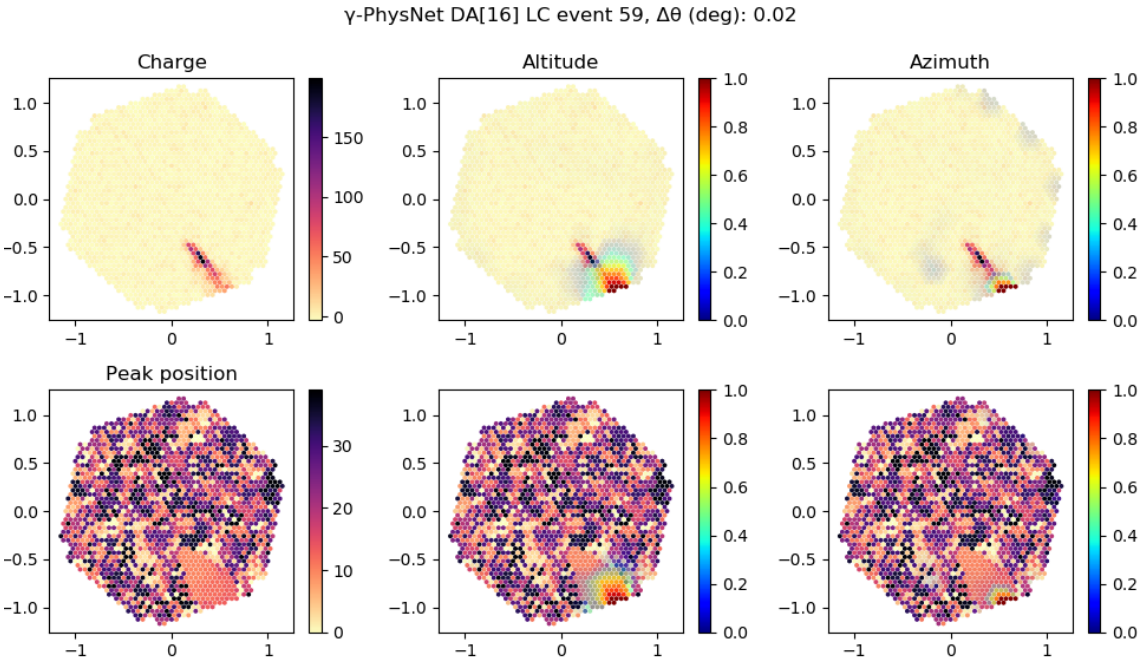


Figure 4.27 – Grad-CAM applied to γ-PhysNet for the task *direction reconstruction*, well reconstructed event.

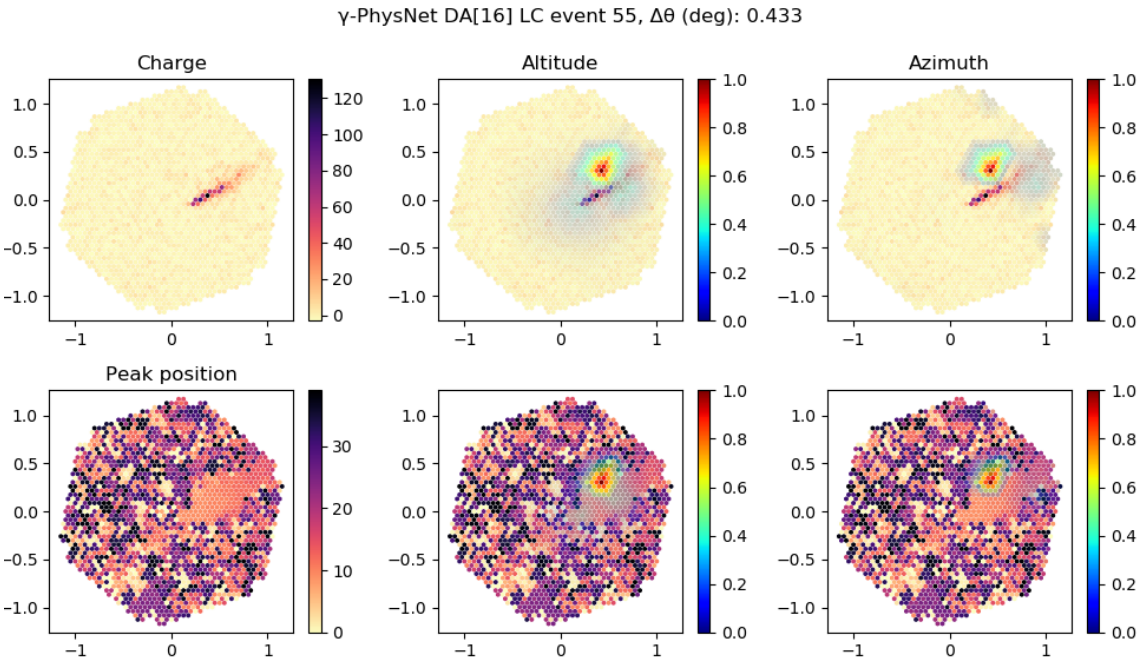


Figure 4.28 – Grad-CAM applied to γ-PhysNet for the task *direction reconstruction*, badly reconstructed event.

Conclusion. With Grad-CAM, I have realized the first steps towards visual explanation of γ-PhysNet. In addition to the examples presented in this section, I have analyzed a limited number of events that seem to exhibit the same behavior. For all the analyzed events, well or badly reconstructed, the important pixels highlighted by the method are located in the area of the shower. This is reassuring as the input data are mainly composed of noise. This first analysis tends to show that the model focuses on the region containing the signal. Moreover, it seems to focus on the tail of the shower for gamma events with well-reconstructed energy and direction. For badly reconstructed gamma events, the model seems to miss the signal, and I could consider to add another task to the architecture to help it focus on the whole shower. However, to fully understand how the model makes its decisions, I need a deeper analysis. Indeed, although Grad-CAM is highly class discriminative, it is low resolution and does not capture fine-grained details. Moreover, it can be hard for the human eye to distinguish in the input images the signal from the noise. In a future work, I will use Guided-Grad-CAM, that is both highly class discriminative and high resolution. I will also use simulated data containing true signal information to better compare highlighted pixels and true shower by removing the noise from the images. In a complementary approach, I could analyze the attention maps produced by the dual attention modules, especially the spatial ones, and compare them to Grad-CAM heatmaps. However, the method to merge the attention maps of the three attention modules of the model has to be defined. This will be the topic of a future work. Finally, I need to analyze more events to determine the consistency of the behaviors observed. As the methods presented in this section are visual methods, this may require to elaborate reliable statistical indicators.

4.5 Summary

In this chapter I have presented γ-PhysNet, a deep multi-task architecture for single-telescope IACT full event reconstruction. My model exploits the multi-task nature of IACT events to perform gamma/proton classification, energy and arrival direction reconstruction. Multi-task learning allows the degeneracy of the particle parameters induced by the atmospheric detection of the showers to be significantly avoided. The addition of the impact point reconstruction helps regularize the model and reduce the initialization variability. γ-PhysNet is composed of a physically inspired multi-task block and a standard very deep convolutional neural network as a backbone, implemented with Indexed Convolution to process the hexagonal pixel images of the LST.

γ-PhysNet Performance. My experiments show that my architecture outperforms the widespread Hillas + Random Forest analysis on Monte Carlo simulated data. The improvement brought by γ-PhysNet is significant for the domain, in particular below 100 GeV. My extensive experiments also show that multi-task learning in the context of CTA data analysis achieves better performance than single task networks.

Besides, I have explored a variety of selection cuts for the experiments on multitasking to explore the limits of the model. With the low cuts, I have relaxed the intensity cut to seek for sensitivity at low energies. γ-PhysNet demonstrates a very competitive sensitivity in single-telescope analysis below 200 GeV. It also outperforms the MAGIC observatory that benefits from stereoscopy, even if the telescope properties are different and the improvement may not only due to the model itself. This good performance allows studying extragalactic sources, such as gamma-ray bursts, and searching for dark matter. To search for the best possible spatial resolution of the model, I have strengthened the intensity cut with the high cuts. Although γ-PhysNet achieves an angular resolution of 0.1° between $200\,GeV$ and $3\,TeV$, the sensitivity is sacrificed. In a search to a trade-off, the mid cuts still have a competitive sensitivity below 200 GeV while slightly improving the angular resolution compared to the low cuts. They are a good starting point to investigate data selection that would allow both a good spatial resolution and a sufficient sensitivity.

Although CTA is not expected to realize single-telescope analysis, the good results obtained with the low cuts, if confirmed on real data, show that it could be worth considering it. It could be done as a supplementary online analysis to search for new sources and generate alerts to other observatories.

Augmenting γ-PhysNet with Attention. As highlighted in Section 1.4, gamma event reconstruction from Cherenkov images is challenging. The difference in shape of showers produced in the telescope camera between a high-energy gamma and a low energy proton can be faint and lie in subtle details. To exploit these subtleties I have probed attention mechanisms for the backbone of γ-PhysNet. Originating from natural language processing, attention helps the network learn to focus on specific parts of the feature maps, spatially, channel-wise or both. I have conducted an extensive ablation study of three attention methods. My experiments demonstrate the stability of the γ-PhysNet architecture as the addition of attention modules does not completely change the performance on the different tasks. Besides, they show that attention helps reduce the variability of the model due to parameter initialization, enforcing the robustness of the model. In addition, the Squeeze-and-Excitation and the dual attention methods allow fine-tuning the model for direction and energy reconstruction, in particular at higher energies.

γ-PhysNet Efficiency. The contribution of my multi-task architecture also lies in its speed as we can expect a substantial gain by using a single network instead of one for each of the three tasks. The convolutional backbone of such models has a much higher computational cost than the model head composed of fully connected layers. Relying on a unified backbone shared between all tasks significantly enhance the global efficiency. Furthermore, although not optimized, γ-PhysNet is about 800 times faster than a state-of-the-art template-based analysis method while showing consistent performance. Speed is actually a strong requirement to enable real-time source and transient event detection as well as alert broadcasting to other observatories. It is also essential for offline analysis considering the amount of data that CTA will produce and that must be reanalyzed every year to benefit from the improvement of the analysis models.

First Steps towards γ-PhysNet Explainability. Finally, I have carried out a preliminary analysis of our architecture. Computing its receptive field, we have seen that it might be possible to lighten its ResNet backbone without loss of performance, thus reducing its computational cost and increasing its inference speed. I have also applied Grad-CAM, a visual explanation method, to γ-PhysNet to analyze well and badly reconstructed events. We have seen that the proposed architecture focuses on the region of the images containing signal to make its decision. This is reassuring as the data are mainly composed of noise. Besides, it seems that, for well reconstructed events, the model focuses on a part of the shower while for badly reconstructed events, the most important pixels for the decision are outside the shower. But I cannot draw conclusions as Grad-CAM is a visual explanation method, it is low resolution, I only observed few examples, and I do not have the signal ground truth for the moment. To better understand the model's behavior, I need to deepen the analysis with a high-resolution method and the information of the pixels containing the true signal. Besides, for the dual attention version of the model, the observation of the spatial attention maps will help complete this investigation. However, this preliminary analysis gives a lead that could be explored to improve the performance of the model.

$\sim \gamma \sim$

In the next chapter, I will analyze the first real data produced by the LST1 with γ-PhysNet augmented with attention.

5

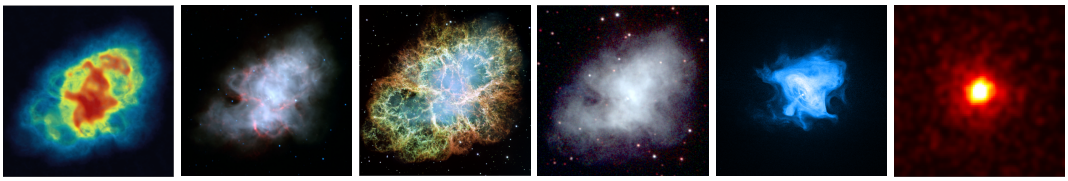
Preliminary Analysis of the Crab Nebula with γ-PhysNet

During the writing of this thesis, the LST1 prototype in La Palma acquired first exploitable data of four already known gamma-ray sources, including the Crab Nebula. In astronomy, this well-studied source is often chosen as a standard candle. In this chapter, I realize a very preliminary analysis of two observation runs of the Crab Nebula with the γ-PhysNet DA[16] described in 4. As there are still important differences between the simulation data and the real ones, the aim of this analysis is to check that γ-PhysNet DA[16] is able to detect the source, rather than evaluate precisely its performance. As a reference, the analysis with the robust Hillas + RF method is also carried out.

5.1 The Crab Nebula

The Crab Nebula is a supernova remnant (see Section 1.1.2.2 for details) situated in the constellation of Taurus. 6500 light years away from the Earth, it results from a bright supernova recorded by Chinese astronomers in 1054, and the nebula was first observed in 1731. The nebula, powered by the most energetic pulsar known, and the first discovered, emits radiation in a broad spectral range, from the radio to the gamma wavelengths, as illustrated in Figure 5.1. This is one of the most studied source of cosmic rays, and the first one detected by an IACT [Weekes et al., 1989].

Recently, Nigro et al. presented the Crab Nebula spectrum obtained with a multi-instrument analysis [Nigro et al., 2019], as shown in Figure 5.2. Besides, stereoscopic observations from H.E.S.S. allowed measuring the extension of the Crab Nebula with unprecedented precision [Abdalla et al., 2019a], as shown in Figure 5.3. Evaluated to 52" (0.0145°), it is significantly larger than the extension seen in X-rays. However, the angular resolution of γ-PhysNet and Hillas + RF for the LST1 is 10 times higher. Therefore, the observations of the Crab Nebula with the LST1 allow for a detection of the source, but not for its morphological study.



(a) Radio (b) Infrared (c) Visible light (d) Ultra violet (e) X-rays (f) Gamma rays

Figure 5.1 – Crab Nebula observed at different wavelengths. Credits: Radio: NRAO/AUI and M. Bietenholz, NRAO/AUI and J.M. Uson, T.J. Cornwell; Infrared: NASA/JPL-Caltech/R. Gehrz (University of Minnesota) ; Visible: NASA, ESA, J. Hester and A. Loll (Arizona State University) ; Ultraviolet: NASA/Swift/E. Hoversten, PSU ; X-ray: NASA/CXC/SAO/F.Seward et al. ; Gamma: NASA/DOE/Fermi LAT/R. Buehler.

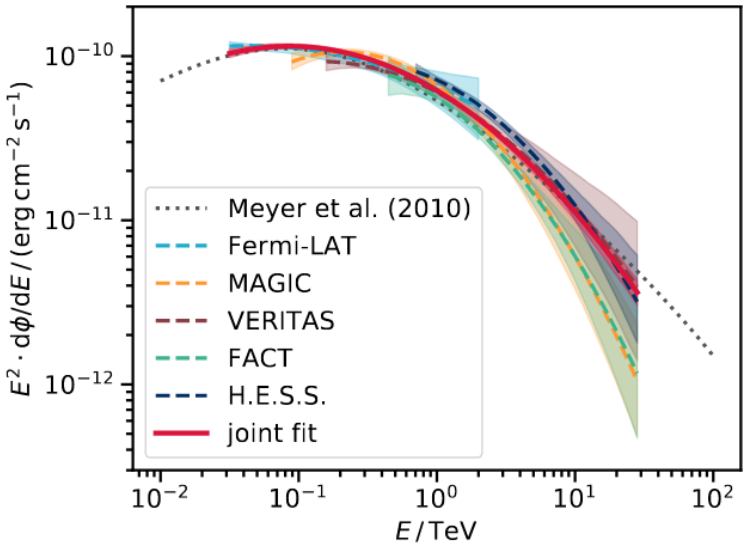


Figure 5.2 – Spectral energy distribution of the Crab Nebula obtained with a multi-instrument analysis (red line). The shaded areas represent the uncertainty of the methods. Source [Nigro et al., 2019].

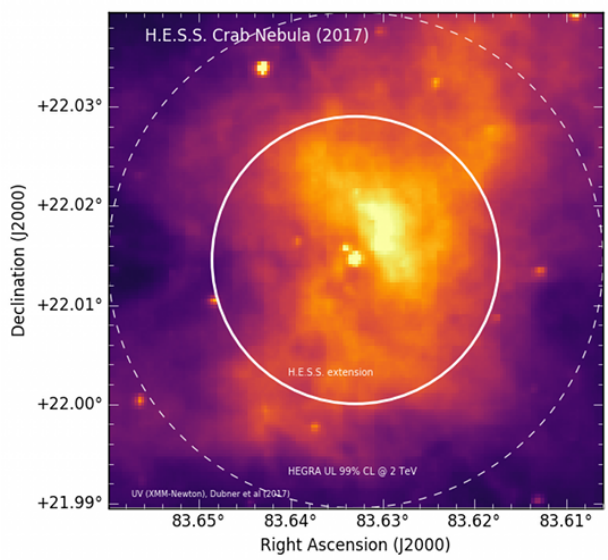


Figure 5.3 – Crab Nebula extension in the gamma spectrum measured by H.E.S.S. (white circle). Source [Abdalla et al., 2019a].

5.2 LST1 Observations and Discrepancies with Simulations

5.2.1 Observation Runs

The data analyzed in this chapter were acquired the 18^{th} of February, 2020. They consist of two observation runs:

- an ON run (no. 2011), with the telescope pointing at the source direction (71.3° in altitude and 253.4° in azimuth at the beginning of the run). It lasted 20 minutes and 55 seconds, and recorded 9 982 457 events.
- an OFF run (no. 2012), with the telescope pointing outside the source direction (68.6° in altitude and 256.7° in azimuth at the beginning of the run). It is a control run used to measure the background emission. It lasted 21 minutes and 30 seconds, and recorded 9 850 595 events.

It is worth noticing that the telescope moves during the runs to follow its initial pointing direction. This way, the region of the sky observed remains the same.

These runs have been selected because the initial pointing direction of the telescope is close to the one of the simulations used to train the models.

5.2.2 Data Discrepancies

Although we benefit from high-quality simulations, there are several discrepancies between the simulated data used to develop γ-PhysNet and the runs 2011 and 2012. The origins of these discrepancies are threefold: the intrument response, the data preprocessing and the observation conditions.

Instrument Response. The LST1 is still in commissioning, and the knowledge of the instrument will be improved during this process. In particular, we already know from the internal analysis of muon real data realized in the LST collaboration that the optical efficiency of the telescope is different from the one defined in the model used to produce the simulations by $\sim 20\%$, resulting in real data images with less intensity. As deep learning networks are sensitive to changes in the data distribution, this certainly affects the performance of the model, especially for the energy reconstruction task. To overcome this kind of issue, data are generally normalized before feeding neural networks. However, the charge value of the signal pixels is crucial to estimate the energy of the primary particle. Preliminary experiments have shown that normalizing the input data worsens the model performance.

Data Preprocessing. As mentioned in Section 1.3.3, the data produced by the LST1 are integrated with the local peak method, while the LST4 monotrigger dataset benefits from the neighbor peak method. Moreover, in the cta-lstchain version used to process runs 2011 and 2012, the implementation of the local peak method is different between real data and simulations, resulting in another $\sim 20\%$ intensity reduction for the real data.

Besides, this difference in implementation also affects the pixel peak time computation. This results in pixel peak times in range $[\sim 0; \sim 40]$ for the simulations, and range

$[\sim -20; \sim 20]$ for the real data. As the temporal information is crucial for single-telescope analysis, this discrepancy strongly affects the model performance. Noteworthy, in both cases values outside the range come from noise pixels, and cannot be trusted. The time peaks are then clipped for the analysis.

Observation Conditions. Analysis models are prepared for defined pointing directions. In particular, γ-PhysNet DA[16] has been trained with the LST4 monotrigger dataset (see Section 1.5) simulated with a telescope pointing direction of 70° in altitude and 180° in azimuth. A different telescope direction corresponds to a different atmosphere thickness and to a different magnetic field, and so to a different shower development. As a result, for a given particle, the image recorded by the telescope camera is different. Although the chosen runs have telescope altitudes close to the one of the LST4 monotrigger dataset, their azimuths are rather different. This could affect the analysis performance.

Furthermore, the level of night sky background and the atmospheric conditions (temperature, pressure, humidity ...) defined in the training dataset are probably different from the ones of the runs 2011 and 2012. This could also affect the analysis performance. Noteworthy, once the commissioning of the telescope is finished, it is considered to train several run-wise models with observation conditions as close as possible to the real ones.

5.2.3 Reducing the Discrepancies

In order to obtain reliable results, it is necessary to solve as much as possible the differences between simulations and runs 2011 and 2012. However, due to time constraints, it is not feasible to generate new simulations. Still, we can take actions on three aspects of the discrepancies: the intensity of the images, the integration method and the temporal information distribution.

Intensity of the Images. To account for the differences in optical efficiency and data integration detailed in Section 5.2.2, the LST collaboration applies a rescaling factor of 0.66 to the pixel charge of the simulated data before training the Hillas + RF model. This factor derives from the real muon analysis mentioned in Section 5.2.2. As illustrated in Figure 5.4, it allows decreasing the intensity distribution difference between the real data and the simulations, denoted MC (for Monte Carlo). It is worth noticing that the intensity is computed after a cleaning operation, and so represents the total intensity of **signal** pixels.

However, this factor is not adapted for γ-PhysNet. As illustrated in Figure 5.5, it underestimates the noise pixel charge during the model training, and I observe that it dramatically disturbs the reconstruction of real data, especially the gamma / proton separation. On the contrary to the Hillas + RF method that reconstructs events from cleaned images, deep neural networks rely on all the pixels, including the noise ones. In this case, computing the rescaling factor so that the charge distribution of all the pixels matches between simulated and real data seems better adapted. As shown in Figure 5.5, the factor 0.84 corresponding to the ratio of the medians of both distributions achieves this goal. However, this leads to slightly overestimating the shower intensity at training time, and may affect the reconstruction of the real data parameters, especially it may result in underestimating their energy.

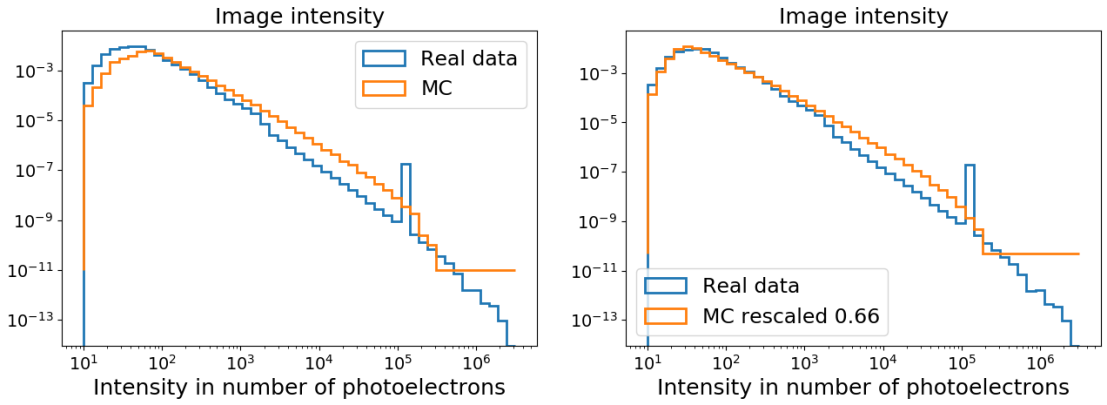


Figure 5.4 – Intensity discrepancy between real data and simulations. *Left*: distribution of the intensities of both real data (runs 2011 and 2012) and simulations. The intensity represents the total pixel charge of the images after a cleaning operation. *Right*: the pixel charge of the simulation data is rescaled by a factor of 0.66. Their intensity distribution is then closer to the one of the real data.

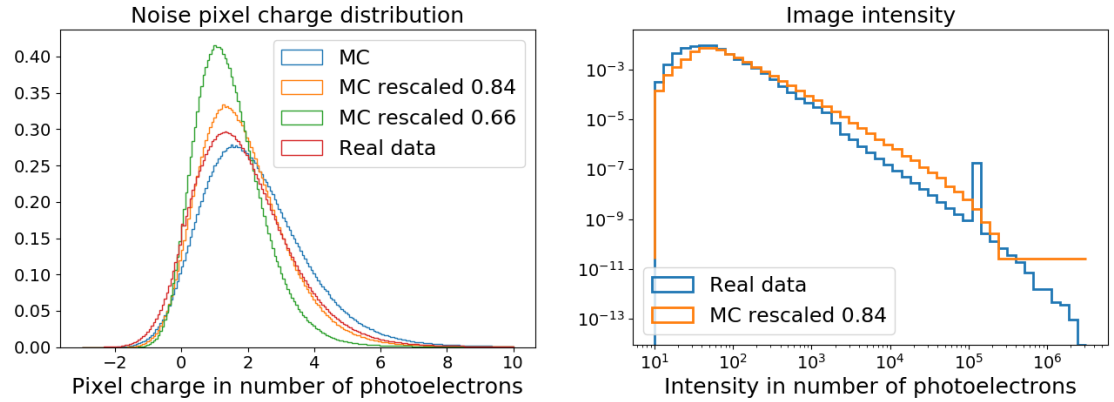


Figure 5.5 – Rescaling the charge with a factor of 0.84 that takes into account all the pixels. *Left*: the noise pixel charge distribution discrepancy is significantly reduced, while a factor of 0.66 would underestimate the noise pixel charge of the simulations. *Right*: the image intensity (after a cleaning operation) of the simulations is still slightly overestimated.

Integration Method. Beside the neighbor peak method, the LST4 monotrigger dataset has also been integrated with the local peak method. γ-PhysNet DA[16] can thus be retrained with this version of the dataset. However, the discrepancy in the temporal information distribution still exists.

Temporal Information Distribution. To reduce the remaining discrepancy of the temporal information, I shift the time peaks of the real data by 20 ns so that their distribution lies in the same range as the simulated data one. However, as illustrated in Figure 5.6, both distributions still have a slightly different shape. More importantly, the distributions of the time peak of the pixel with maximum intensity are rather different, modifying the temporal information of the shower development. This may significantly affect the event reconstruction with γ-PhysNet, but it seems a better solution than having in the real data negative time peaks unseen during training.

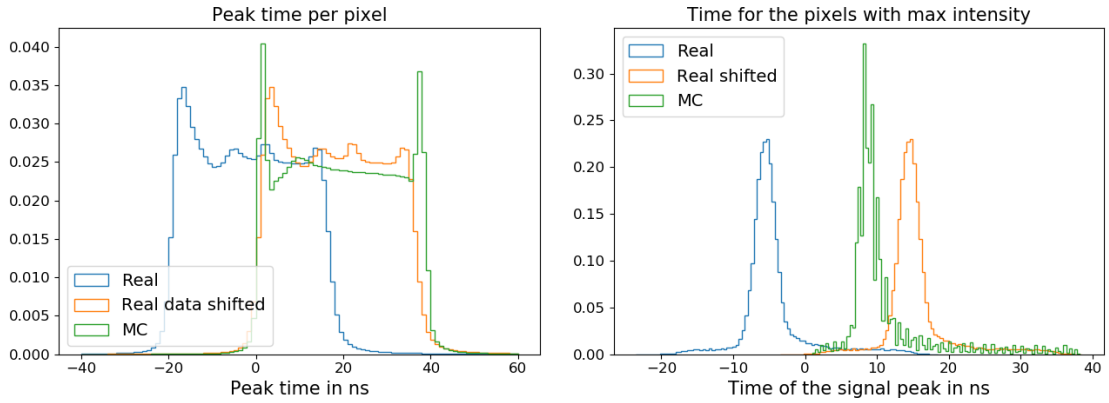


Figure 5.6 – Timing discrepancies between real data and simulations. *Left*: distribution of the peak time for all the pixels. *Right*: distribution of the peak time for the pixel of maximum intensity of each image.

5.3 Data Analysis with γ-PhysNet

The preliminary analysis of the real data presented in this section consists in determining whether the model is able to detect the source in the ON run (2011). It relies on the following procedure:

1. reconstruction of the event parameters;
2. data selection;
3. gamma event detection;
4. source detection.

For reference, I compare the results of γ-PhysNet DA[16] and Hillas + RF.

5.3.1 Full Event Reconstruction

All the data of the runs 2011 and 2012 are processed with γ-PhysNet DA[16] trained in the conditions described in Section 5.2.3. More precisely, the network is trained with the LST4 monotrigger dataset integrated with the local peak method. In addition, the charge of each pixel is scaled by 0.84. Besides, for the reconstruction of the real data, their temporal information is shifted by 20 ns. Although this is suitable for the preliminary analysis presented in this thesis, a production of updated simulations may be necessary to carry out a deeper analysis.

The reconstruction consists then in predicting the energy, altitude, azimuth and gammaness (confidence in the gamma nature) of the primary particle that produced the image.

5.3.2 Data Selection

We have seen in Chapter 4 that γ-PhysNet DA[16] obtains very interesting performance with the low cuts (see Section 4.2.4 for details) on simulations in terms of sensitivity, especially below 200 GeV. However, because of the discrepancies between simulations and real data detailed in Section 5.2.2, I apply stronger selection cuts for the analysis of runs 2011 and 2012. Derived from the mid cuts detailed in Section 4.2.4, they consist of:

- intensity > 200, the intensity being computed after a tail-cut cleaning (picture threshold: 6, boundary threshold: 3, minimum picture pixel neighbors: 1),
- leakage (border width: 2, intensity, threshold: < 0.2).

Compared to the mid cuts applied in Chapter 4, the selection is a bit stronger as the pixel charge distribution of real data is lower.

Besides, we can see in Figure 5.4 that the intensity distribution of real data exhibits an excess of events between 116 000, 119 000 photoelectrons. As illustrated in Figure 5.7, these events probably correspond to an artificial source of light. Indeed, almost all the pixels received more than 70 photoelectrons. Therefore, I also select events with intensity lower than 10^5 in order to discard these odd events. As a side note, we can see in Figure 5.7 that some pixels seem to be dead. This may affect the reconstruction performance. To limit the effect of dead pixels, in a future work, I could add dropout (see Section 2.1.2 for details) during training. Besides, dropout helps the model generalize.

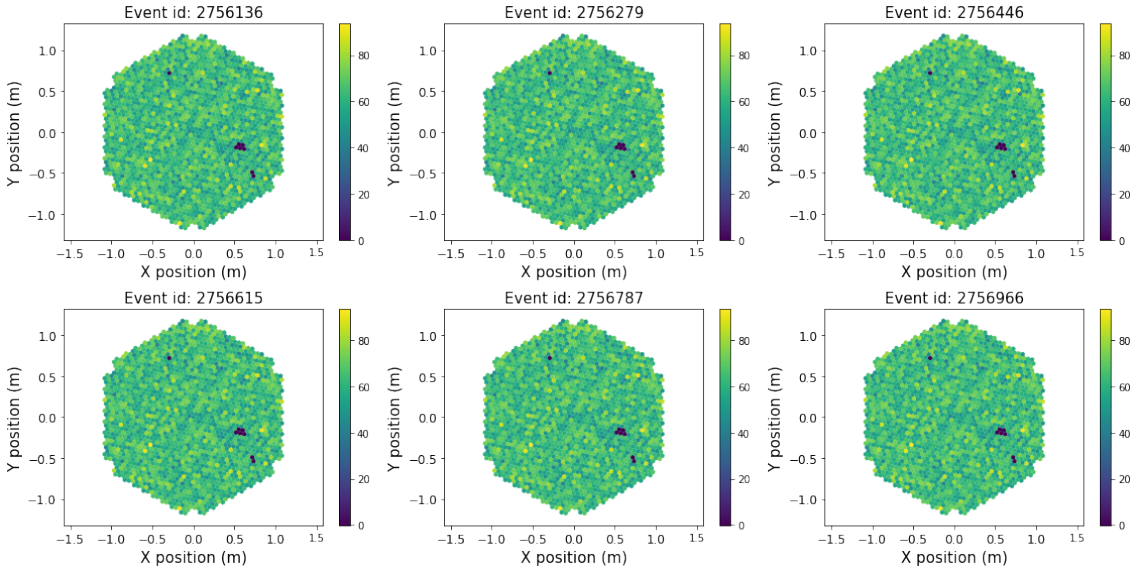
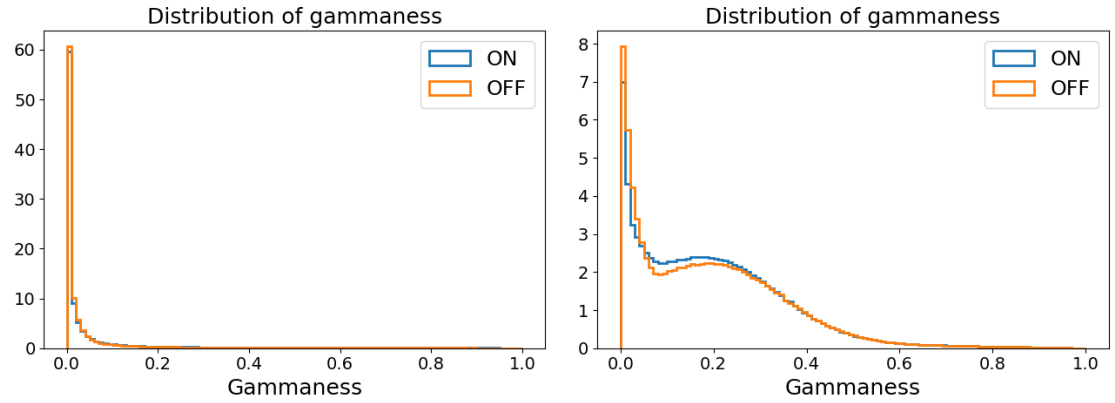


Figure 5.7 – Odd events in the real data corresponding to the excess of data with intensity in range $[116\,000, 119\,000]$ photoelectrons.

These selection cuts discard $\sim 86\%$ of the data. Once applied, we observe in Figure 5.8, that the gammaness distribution is rather different. However, they are not comparable as the models rely on different principles.



(a) γ-PhysNet DA[16]

(b) Hillas + RF

Figure 5.8 – Comparison of the gammaness distribution of reconstructed events between γ-PhysNet DA[16] (on the *left*) and Hillas + RF (on the *right*).

5.3.3 Gamma Event Detection

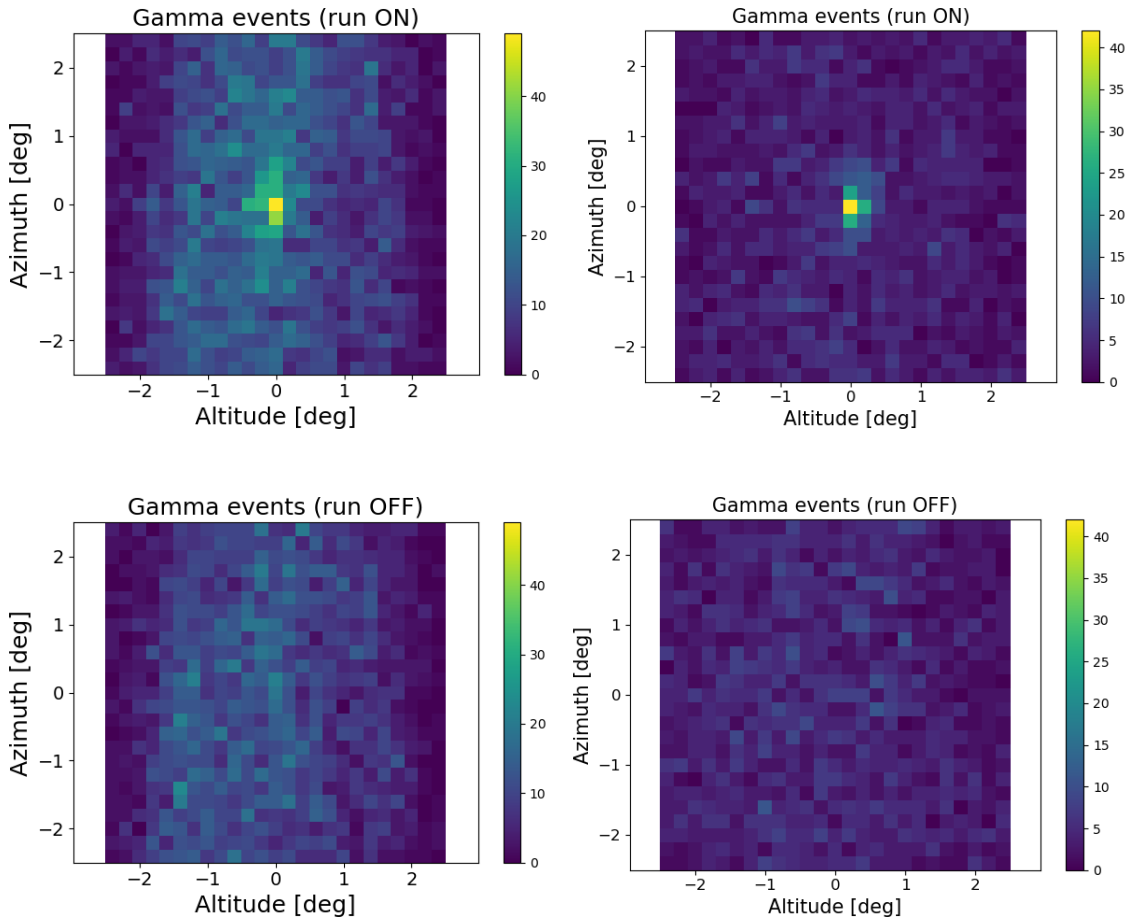
The next step of the analysis consists in defining the gammaness threshold above which events will be considered as gammas. The gammaness represents the confidence level that an event has been produced by a gamma ray. Practically, in the case of γ-PhysNet, this is the output of the network for the class "gamma" after the softmax operation. The definition of this threshold relies on the model performance on the simulation data. In particular, to produce the sensitivity curves presented in Chapter 4, the threshold that maximizes the performance is found through an optimization process for each energy bin (11 bins in total). However, because of the discrepancies between simulations and real data, we cannot rely on such a fine-tuned threshold for γ-PhysNet, neither for the Hillas + RF method. Therefore, I choose a safe gammaness threshold of 0.8 for both models. This threshold corresponds to a quite high level of confidence in the gamma nature of the particle. It is worth noticing that this threshold probably degrades the sensitivity performance presented in Chapter 4.

Table 5.1 presents the number of gamma events detected in the runs ON and OFF once the gammaness threshold has been applied to the data selected in Section 5.3.2. As the gammaness distribution of γ-PhysNet DA[16] and Hillas + RF are different, these numbers are not comparable. However, we can see that they are of the same order.

Table 5.1 – Number of gamma events detected in the real data by γ-PhysNet DA[16] and the Hillas + RF method.

Run	ON	OFF
Duration	20 min 55 s	21 min 30 s
γ-PhysNet	12 038	9 751
Hillas + RF	5 752	6 432

The reconstructed direction, relatively to the telescope pointing direction, of the gamma events detected this way can be represented as distributions in the altitude-azimuth 2D space. As we know the position of the source in the ON run, the observation of these altitude-azimuth maps for the ON and OFF runs provides a visual hint of the reconstruction quality. In the case of a good model, the maximum of the distribution for the ON run should be located at the source direction, and the distribution should be narrow. For the OFF run, the distribution should be smoother, following the acceptance of the telescope. Figure 5.9 shows the altitude-azimuth maps for the ON and OFF runs reconstructed by γ-PhysNet and Hillas + RF. We can see that the maps of γ-PhysNet are noisier, and its distribution for the ON run is wider than the one of Hillas + RF. This is not surprising as neural networks are more sensitive to changes in the data distribution, such as the discrepancies described in Section 5.2.2.



(a) γ-PhysNet DA[16]

(b) Hillas + RF

Figure 5.9 – Comparison of altitude-azimuth maps of ON and OFF runs between γ-PhysNet DA[16] (on the *left*) and Hillas + RF (on the *right*).

5.3.4 Source Detection

To verify if the source is detected by the model, it is necessary to remove the expected background from the gamma events detected in the source region of the ON run. This background is measured in the OFF run after a normalization process. Indeed, the observation time, the telescope trigger rate and the observed region of the sky are different between runs 2011 and 2012. Once the event distributions are normalized, we can compute the significance of the source detection.

The ON-OFF method usually consists in recording an ON and an OFF run with same duration and trigger rate, pointing to close directions in the sky in order to benefit from similar background. Then, the excess distribution of gamma events in the ON run is computed by subtracting the distribution of events detected in the OFF run. However, the ON and the OFF runs analyzed in this thesis have different observation duration and telescope trigger rates. It is then necessary to normalize the distribution of events detected in the OFF before computing the excess. The normalization consists in weighting the distribution of gamma events detected in the OFF run. These events correspond to the background noise wrongly detected as gamma events. To compute the normalization weight α, we first define a region outside the location of the source in the ON data. The events detected in this region correspond to background for both runs. As the telescope points to the direction of the source, I choose the region comprised between 1° and 2° from the pointing direction, as illustrated in Figure 5.10.

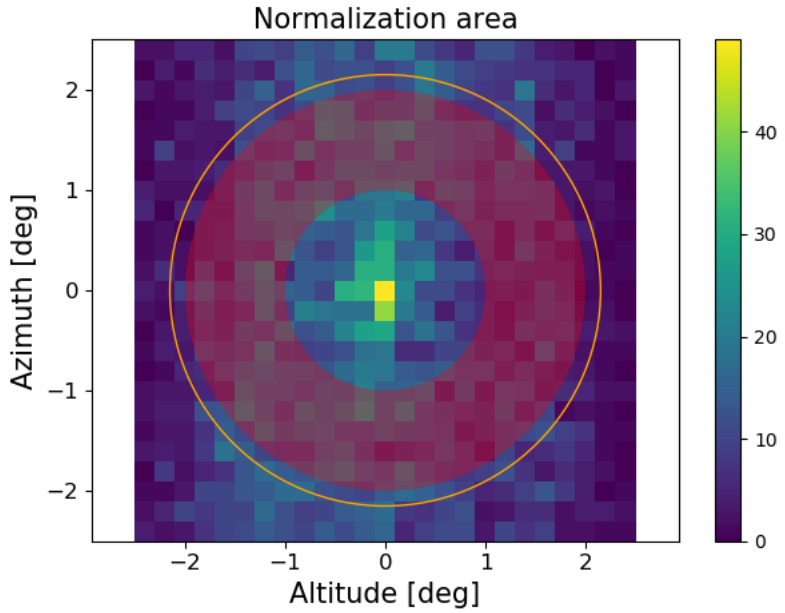


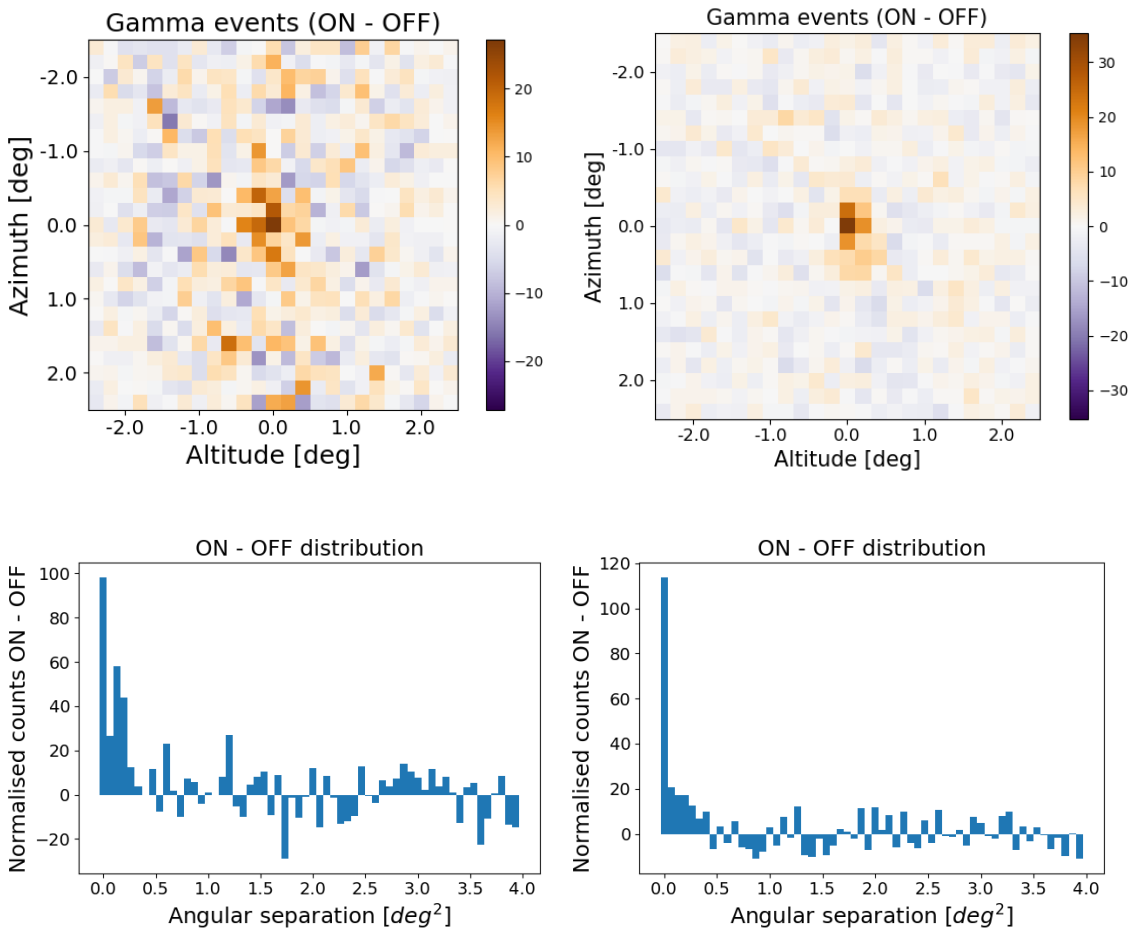
Figure 5.10 – ON-OFF normalization region represented in the camera view. The red area corresponds to the region used for the ON-OFF normalization. The orange circle represents the LST1 field of view.

Next, we compute the ratio of events detected in this region in the run ON and in the run OFF:

$$\alpha = \frac{N_{ON}^{bkg}}{N_{OFF}^{bkg}} \tag{5.1}$$

We compute then the excess distribution of gamma events detected in the ON run by

subtracting the distribution of events detected in the OFF run weighted by α. Figure 5.11 represents the signal excess as an altitude-azimuth map and as the distribution of squared angular separations between the source direction and the reconstructed one. Again, we can see that the excess of Hillas + RF is less noisy and more concentrated in the source direction than the one of γ-PhysNet.



(a) γ-PhysNet DA[16]

(b) Hillas + RF

Figure 5.11 – Comparison of the excess of gamma detected as altitude-azimuth maps and angular separation distributions between γ-PhysNet DA[16] and Hillas + RF.

Finally, to compute the significance of the detection with the Li & Ma formula detailed in Section 1.4.4.4, we apply a last cut to the data on the angular separation between the reconstruction direction of the detected events and the source direction. As for the gammaness threshold, the computation of the optimal angular separation relies on the performance of the model on the simulation data. Again, because of the discrepancies between the simulations and the real data, we cannot apply the angular separation found during the computation of the sensitivity. Therefore, I choose a safe angular separation of 0.5°, corresponding to the angular resolution upper bound of the Hillas + RF method between 50 GeV and 5 TeV. The resulting significance of the Crab Nebula detection is shown in Table 5.2. Although γ-PhysNet obtains a lower significance than Hillas + RF, its detection of the source is clear, stronger than the five-sigma discovery level used in astrophysics.

Table 5.2 – Significance of the Crab Nebula detection by γ-PhysNet DA[16] and the Hillas + RF method.

	γ-PhysNet	Hillas + RF
Significance	7.8 σ	9.9 σ
Excess	230	170

5.4 Discussion

The very preliminary analysis of real data presented in this chapter shows that γ-PhysNet is able to detect the Crab Nebula. Despite the discrepancies between the simulations used to train the model and the real data, it performs a seven-sigma detection from one ON run. This is encouraging as neural networks are particularly sensitive to changes in the data distribution, and so, these discrepancies probably affect the reconstruction performance.

Besides, as expected, the Hillas + RF method obtains better results. It is indeed more robust to changes in the data as it uses geometrically extracted parameters as input data. These parameters are less likely to be affected by slight changes in the pixel charge level. Moreover, the temporal information, that is crucial for the single-telescope analysis, is compressed into a single gradient value per event in the case of Hillas + RF. This reduces the possibility of the model to be affected by modification of the temporal information computation during the image integration process. However, we have seen in Chapter 4 that γ-PhysNet, exploiting richer temporal information, obtains better results than Hillas + RF on simulated data. With simulations corresponding better to real data, as the next production should, I expect a significant improvement of the performance of γ-PhysNet on real data. It is important to notice that the correspondence between simulations and real data is a crucial issue for IACT data analysis. However, it is out of the scope of this thesis whose goal is to demonstrate that deep learning techniques are applicable to CTA data analysis.

Besides, the use of dropout during training should help reduce the dependence to the training data, and allow the model to handle better dead pixels.

Then, this preliminary experiment stresses the need to better control the data preprocessing for both the simulated and real data. In particular, the image calibration and integration pipeline should be exactly the same. The next version of cta-lstchain solves this issue. In addition, the simulation conditions must be as close as possible to the real ones. New simulated data updated with the knowledge of the telescope are currently in production. They will certainly help improve the model performance, as the use of closer simulated observation conditions will.

Finally, the greater sensitivity of neural networks to the data constitutes their strength, but is also a potential cause of performance degradation if the conditions met during training are too much altered. We need to find a trade-off between sensitivity and robustness.

6 Conclusions and Perspectives

Ground-based gamma astronomy aims to answer fundamental questions about the universe by studying the gamma rays produced in its most violent phenomena. CTA, the next generation gamma-ray observatory, proposes challenges unseen so far due to the better sensitivity of its telescopes, their number and the tremendous amount of data it will generate every year. To achieve the best science possible, we need high-performance and efficient analysis methods. Following the recent advances of deep neural networks, especially in the computer vision field, this thesis addresses these challenges using deep learning for the analysis of IACT data from CTA in the single-telescope context.

6.1 Contributions

The main contribution of this thesis is a deep multi-task architecture (denoted γ-PhysNet) augmented with attention that outperforms a widespread method for the analysis of LST simulated data. γ-PhysNet exploits the multi-task nature of IACT events reconstruction to avoid the degeneracies introduced by their atmospheric detection. The improvement brought by the approach proposed in this thesis on Monte Carlo simulated data is significant. In particular, the sensitivity of γ-PhysNet below 200 GeV obtained with the low cuts could enhance the study of extragalactic sources, such as gamma-ray bursts and transient phenomena. Besides, extensive experiments have highlighted the robustness of the proposed architecture to weight initialization variability, and to slight changes in the model, such as the attention bottleneck ratio. Furthermore, although not optimized for production, using modern GPUS, γ-PhysNet is about 800 times faster than a state-of-the-art template-based analysis method. Its efficiency is consistent with online and offline analysis use cases. The interesting results obtained in a single-telescope context constitute a good basis to build a model for stereoscopic analysis. Besides, the preliminary analysis of real data shows that γ-PhysNet achieves a seven-sigma detection of the Crab Nebula. However, this analysis also shows that the proposed architecture is sensitive to the discrepancies observed between the simulations used for training and the real data.

In addition, this thesis proposes a preliminary analysis of γ-PhysNet in order to understand its decision-making process. The computation of its receptive field indicates that the depth of its ResNet encoder might be reduced. This could help increase the model inference speed that is crucial for CTA data analysis. Besides, using a visual explanation method with the simulated data, we have observed that the proposed model focuses on the area of the images containing signal to produce its decision. We have also seen that, for well reconstructed events, it seems to focus on the tail of the shower. However, this analysis has to be deepened.

This thesis also presents an original method to apply deep learning algorithms to images with any pixel organization. Specifically, the Indexed Operations have been successfully applied to the hexagonal pixel images of the LST. In the context of solving gamma full event reconstruction with very deep neural network, this strategy has proven to perform better than grid resampling through bilinear interpolation. Thus, γ-PhysNet makes use of this method for the convolutions and pooling operations of its backbone encoder. The main advantages of Indexed Operations are that they do not introduce any distortion in the data, they respect better the neighborhood of the pixels, they avoid preprocessing before training the model, and they do not add useless pixels. However, the current implementation of this method adds a computational overhead to the model.

Finally, in parallel to this scientific work, a framework has been developed to ease the process of training and testing deep learning models with CTA data. It provides all the tools to load the data, define the model (including Indexed Convolution), train the model, store the training data and compare performance across runs. This framework then ensures the robustness and the reproducibility of the results presented in this thesis.

$\sim \gamma \sim$

With the commissioning of its first on-site prototype, CTA has started a journey in the observation of the gamma-ray sky that is expected to last decades. Building on the interesting results shown by this thesis, we still need to tackle many challenges. First, the approach presented addresses single-telescope analysis while CTA has been designed for stereoscopy. Then, although γ-PhysNet is much faster than the state-of-the-art analysis method, its efficiency has to be improved to comply with real-time stereo analysis. Next, the good performance obtained on Monte Carlo simulated data has to be confirmed on real data. Although γ-PhysNet is able to detect the source in the real data analyzed, its performance is degraded. In addition to simulations corresponding better to real data, it is also necessary to explore other leads to improve the generalization over real data. Finally, the preliminary analysis of the model behavior has to be expanded.

6.2 Future Research

6.2.1 Stereo Analysis

This thesis has addressed the analysis of CTA data in the single-telescope context. In particular, I proposed an architecture to perform full event reconstruction from LST1 data, as the LST1 is the first prototype of CTA on site. However, CTA has been designed as an array to benefit from the stereoscopic analysis of detected events. In this configuration,

combining the information from the different telescopes that triggered to a cosmic particle is crucial and challenging.

In the case of deep multi-task learning models, a straightforward solution consists in sharing the architecture encoder between all the telescopes. The latent feature maps are then concatenated before feeding the multi-task block. Preliminary experiments have shown interesting results. However, this configuration has three main drawbacks. First, if sharing the encoder may make sense for telescopes of the same type, it is probably not appropriate for different telescope types. Then, concatenating the feature maps results in a rapidly growing representation with the number of telescopes. If the multi-task block is composed of fully connected layers, as is the case in γ-PhysNet, it may lead to a severe increase of the computational cost for the 19 telescopes of the CTA northern site, and even more for the 99 of the CTA southern site. Finally, the number of telescopes in the input data has to be fixed. As only a part of the telescopes triggers to an event, the data of the rest can be set to zero, resulting in sparse input data. Thus, this solution to combine telescope data is only suitable for small sub-arrays of the same telescope type, such as the LST sub-array, composed of four telescopes.

A more elegant solution consists in considering the features of the different telescopes as a sequence and combining them with a recurrent cell (a GRU [Cho et al., 2014] or an LSTM [Hochreiter and Schmidhuber, 1997]), as proposed by Shilon et al. [Shilon et al., 2019]. In this case, the main issue to solve is how to order the telescope features in the sequence. Shilon et al. use the trigger time as the sorting key. However, training recurrent cells can be challenging.

Besides, the Indexed Convolution approach presented in Chapter 3 is not limited to the hexagonal lattice. It is even not restricted to images. With the help of Indexed Convolution and the Delaunay tessellation, we could apply convolution to the whole array at the layout level to fuse the encoded features of every telescope before feeding the Multi-task block of γ-PhysNet.

As an alternative, we could process each telescope data with γ-PhysNet, and fuse the resulting predictions with a dedicated strategy that could be a network composed of convolutions at the layout level.

6.2.2 γ-PhysNet Depth and Real-Time Analysis

Two kinds of analyses will be applied on CTA data, the offline (on-site and off-site) and the online analysis. The offline analysis requires a high-performance model as it should provide the highest quality data for scientific analysis. Moreover, it will be reapplied every year on all the data acquired so far to benefit from model performance improvements. The online analysis or real-time analysis requires a robust and efficient model able to process data at the telescope trigger rate ($\sim 10\,\text{kHz}$ for the LST). Its purpose is to monitor sources in order to broadcast alerts to other observatories in case of transient phenomena, or to adapt the observation strategy in real time depending on the source detection. It can also be used to reduce the volume of data to transfer by suppressing background events while keeping as many gamma events as possible.

γ-PhysNet efficiency is suitable for the high-quality analysis case (offline analysis), but its inference speed must be optimized for real-time analysis. Methods exist to reduce the computation time of neural networks for production, such as network pruning and quantization [Han et al., 2016, Luo et al., 2017, Zhao et al., 2019]. These methods are generally employed after an extensive hyperparameter optimization process that has not

yet been realized for the model proposed in this thesis.

On the other hand, as we have seen in Section 4.4, from the receptive field point of view, γ-PhysNet seems to be over-parametrized. In such a multi-task architecture, the costliest part is the convolutional backbone, as illustrated in Section 4.3.1. This encoder is a ResNet composed of 55 convolutional layers. As a first step towards computational cost reduction, we can try to find a compromise between decreasing this number of convolutions and keeping a good performance level.

As an alternative, we could replace the backbone by a more efficient architecture. Although a reference, the ResNet is quite old, and recent efforts have been made to design networks with a less important computational cost. Some dedicated directions have already been identified, and are presented in Section 6.2.2.2.

6.2.2.1 Lightening the ResNet

Less convolution layers The most obvious way to reduce the computational cost of γ-PhysNet is to decrease the number of convolutional layers of its ResNet backbone. However, the depth of this encoder is crucial for the network to be able to extract meaningful features with fewer parameters. In a first attempt, I decreased the number of convolutions of γ-PhysNet DA[16][1] from 55 to 19. As illustrated in Figure 6.1, it is achieved by reducing the number of residual layers from 9 to 3 in each stage. I chose to keep the three stages in order to benefit from the three feature scales. This results in a receptive field of 81 pixels, closer to the image size. In this configuration, the model, named γ-PhysNet19 DA[16], achieves an inference rate of **12.25 kHz** on an NVIDIA V100 GPU that is compatible with LST1 real-time analysis. It is worth noticing that this inference rate does not take into account the computational cost of data calibration and integration. Improving the efficiency of these steps is by the way an active field of research at LAPP.

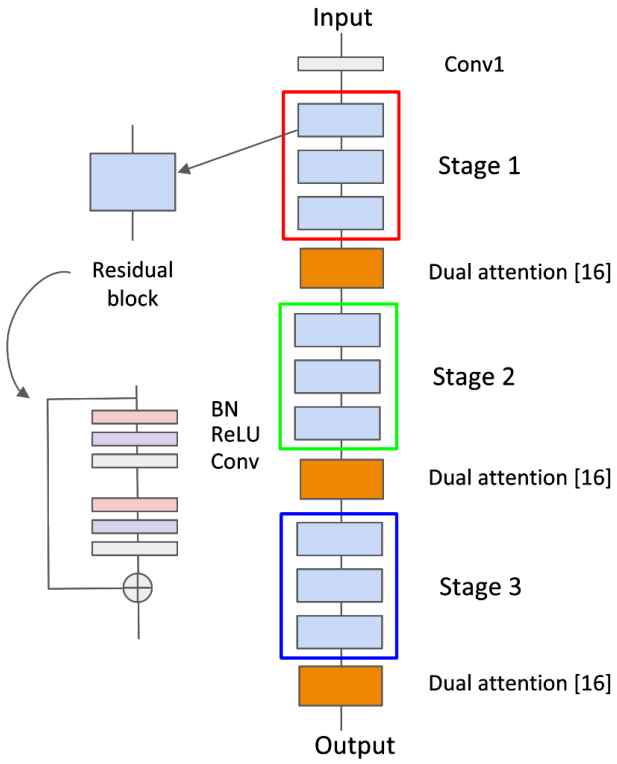
Preliminary Results Figure 6.2 shows the performance of γ-PhysNet19 DA[16] trained on the LST4 monotrigger dataset with the low cuts (see Chapter 4 for details). The comparison of the sensitivity curve, the energy and angular resolutions with the original version of γ-PhysNet DA[16] exhibits no performance loss. These results need to be confirmed as only one training has been run. However, this preliminary experiment shows that this lead is worth investigating to reduce the computational cost of the model.

6.2.2.2 Architecture alternatives

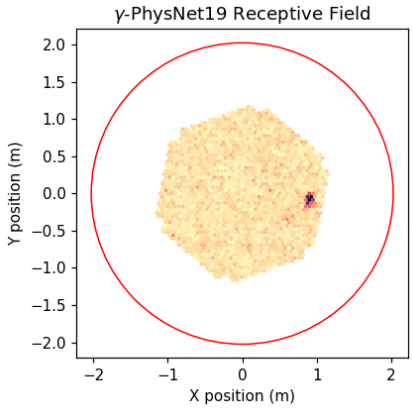
To find a more efficient, and possibly better performing architecture for γ-PhysNet backbone, two leads could be explored.

On one hand, Tan and Le [Tan and Le, 2019] propose a compound scaling method to simultaneously scale all the dimensions of a baseline model (*i.e.*, depth, width and resolution). It consists in maximizing the model accuracy given resource constraints, such as memory or floating point operations per second (FLOPS). It is worth noticing that their method does not modify the nature of the layers to simplify the design problem. Yet, it explores for each layer the dimension space. Starting from the MobileNetV2 architecture [Sandler et al., 2018], the authors propose EfficientNet B0 to B7 using different

[1] With dual attention modules which reduction ratio is 16.



(a) γ-PhysNet19 DA[16] backbone. The number of convolutions is decreased from 55 to 19.



(b) γ-PhysNet19 receptive field.

Figure 6.1 – Lightening the γ-PhysNet backbone to reduce the receptive field.

constraints. For a similar task accuracy, their models are 4 to 19 times more efficient in terms of inference speed than state-of-the-art networks.

On the other hand, Wang et al. [Wang et al., 2020] propose the Cross Stage Partial Network (CSPNet) to reduce the computational cost of neural networks, and increase their robustness against input pattern scales. Their method consists in partitioning the feature maps from the base layer of each block into two parts, with half the feature maps each. For the current ResNet backbone of γ-PhysNet, the base layer corresponds to the first convolutional layer of each of the three stages. Then, the first part of the feature maps passes through the convolutional block whose output is merged with the other part. This allows reducing the number of computations. Moreover, this method makes the gradient propagate through different paths, enhancing the variability of the features learned. The authors show that the CSPNet version of state-of-the-art models achieves similar performance with a computational cost reduction ranging from 2 to 22%.

6.2.3 Adaptation to Real Data

The discrepancy between simulated and real data is a well-known issue in the IACT data analysis field. In [Shilon et al., 2019] Shilon et al. have shown that for H.E.S.S., the angular resolution was significantly degraded when a CNN was applied to real data, with a loss of about 0.04 degrees compared to simulated data. In the same way, although we benefit from high quality simulations to train γ-PhysNet, real LST1 data certainly differ from simulated data. Indeed, the main discrepancy concerns the response of the simulated

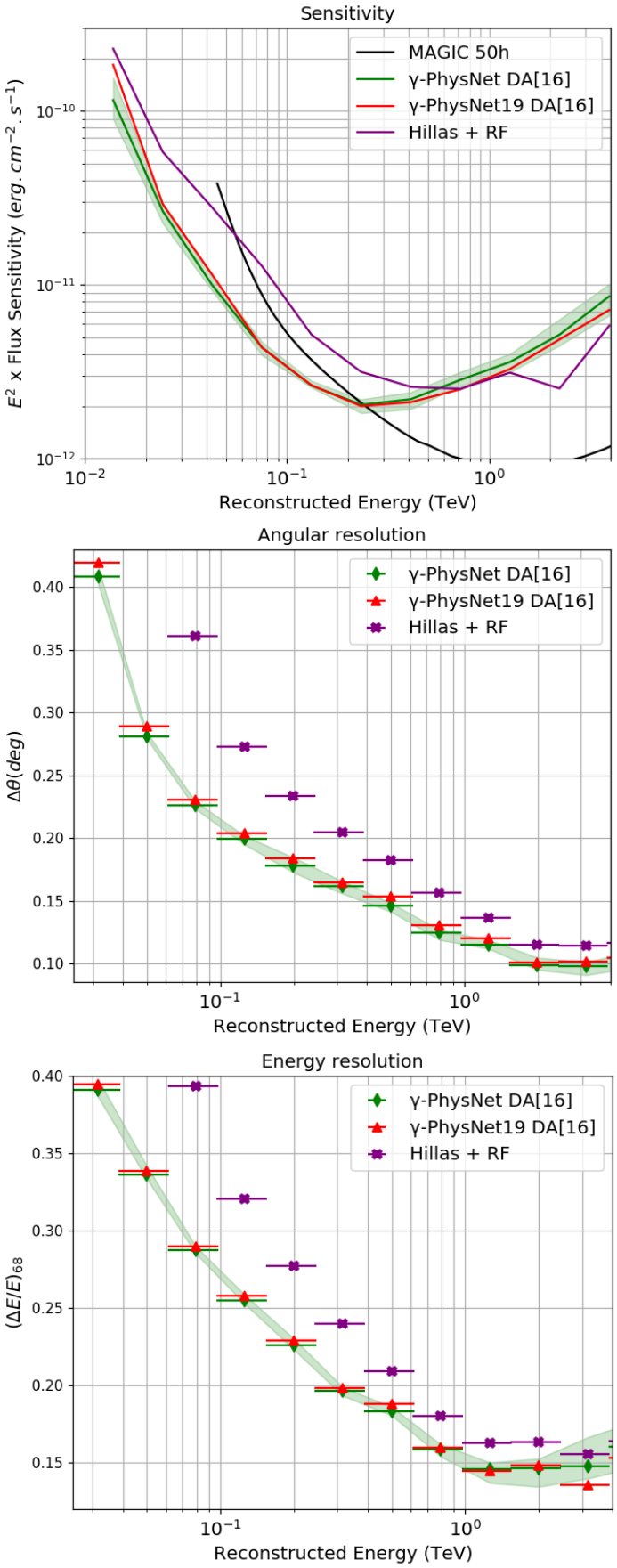


Figure 6.2 – Performance of γ-PhysNet19 with the low cuts.

telescope that relies on approximations of the optical and photomultiplier efficiency, and on the electronic noise of the real one, as we have seen in Chapter 5. Moreover, the physical properties of the telescope will evolve during its lifetime. Besides, although the shower development process is well known, the simulations are produced for particular atmospheric conditions that may be different during the real observations. Furthermore, the simulated night sky background is an estimation of the real one, especially, it does not take into account the presence of stars.

The first lead to improve the performance of γ-PhysNet consists in reducing as much as possible the differences between simulated and real, in terms of simulation conditions and preprocessing pipeline (calibration and integration). A phase of improvement of the simulation, in particular of the telescope response, is currently conducted with real data. Moreover, the next version of cta-lstchain solves the implementation differences mentioned in Chapter 5. Therefore, I expect the model to benefit from these improvements. This context will then permit to carry out advanced model hyperparameter tuning.

Besides, to reduce the sensitivity of γPhysNet to the training data, I will add dropout to the training. This will allow generalizing and handling better dead pixels in the real data. Furthermore, I plan to use real data to improve the performance of γ-PhysNet architecture. Since ground truth is difficult to obtain from real data, generative approaches, such as autoencoders or generative adversarial networks, could help build up relevant features representation of the real data. It has been successfully applied on light curve analysis in [Pasquet et al., 2019]. In the case of γ-PhysNet, I could add another task to the model consisting of an autoencoder. Real data would be then added to the training set, so that the model captures their distribution. However, preliminary experiments with autoencoders on LST1 real data have shown that it is challenging.

6.2.4 Going Deeper with Network Analysis

In Section 4.4 I have taken the first steps towards γ-PhysNet result explainability. With Grad-CAM, a visual explanation method, we have seen that the model focuses on the area of the image containing signal. In the case of well-reconstructed events, it seems to take into account especially the tail of the shower. However, although Grad-CAM is highly discriminative, it is low resolution, and the heatmaps generated have to be scaled up.

On the other hand, Grad-CAM can be combined with another method, guided backpropagation, yielding Guided Grad-CAM, a highly discriminative and high-resolution method. Indeed, guided backpropagation [Springenberg et al., 2015] is a high-resolution explanation method that allows investigating the model behavior at the pixel level. It consists in observing the gradient of the neuron (*e.g.*, the output) of the network we want to analyze with respect to the input pixels. However, only positive gradients are backpropagated through the network, in order to visualize the signal that increases the activation of the neuron we want to analyze.

Besides, it is hard to draw conclusions about the pixels highlighted by visual explanation methods due to the level of noise contained in the images. As γ-PhysNet is trained with simulations, it is possible to retrieve the information about the signal generated by the shower, and to separate it from the noise. Comparing the pixels highlighted and the signal pixels will help better understand the behavior of the model.

Finally, I have augmented γ-PhysNet with attention modules, in particular dual attention that consists of a channel-wise and a spatial attention path. Comparing the spatial attention maps and the heatmaps of Guided Grad-CAM will give insights on the working

of the model. However, as the attention modules are located at three different levels in the network, the observation of the attention maps will require to define a procedure to combine them.

7 Résumé Long en Français

7.1 Astronomie gamma

L'astronomie est un domaine scientifique qui fournit des quantités très importantes de données à partir desquelles les chercheurs désirent caractériser ou modéliser des phénomènes complexes. En cela, l'astronomie requiert des méthodologies performantes de traitement et d'assimilation de données.

7.1.1 Rayonnement gamma et méthode d'observation

En particulier, l'astronomie gamma consiste en l'observation des photons les plus énergétiques produits par des phénomènes astrophysiques violents. Ils sont principalement émis lors de l'interaction de particules accélérées, appelées rayons cosmiques, avec la matière ambiante ou les champs électromagnétiques. Leur étude permet, par exemple, de mieux comprendre les lois qui gouvernent la création des étoiles et l'évolution des galaxies. Elle permet aussi d'explorer une nouvelle physique, telle que la matière noire ou la violation de l'invariance de Lorentz. La détection de ces rayonnements depuis la Terre se fait de manière indirecte, par l'observation de la lumière Cherenkov émise par la gerbe électromagnétique qu'ils génèrent en pénétrant l'atmosphère, comme illustré sur la Figure 7.1. Ces observations sont rendues possibles par des télescopes spécifiques comme les télescopes à effet Cherenkov, ou Imaging Atmospheric Cherenkov Telescopes (IACT), situés sur Terre. La lumière Cherenkov ainsi collectée par des caméras ultra-sensibles laisse une trace caractéristique dont le développement ne dure que quelques nano-secondes. L'analyse du rayonnement gamma consiste alors en sa **séparation du bruit de fond** - les rayons cosmiques, dominés principalement par des protons (tâche de classification entre les types de particules), et la **reconstruction de son énergie et de sa direction incidente** (tâches de régression). Cette analyse est complexe, car les rayons cosmiques peuvent produire des images très similaires à celles des gammas (voir les exemples de la Figure 7.2) et le rapport signal sur bruit est typiquement inférieur à 1/1000. De plus, les images sont en grande partie composées de bruit provenant à la fois de l'électronique des capteurs et du fond lumineux du ciel nocturne. Enfin, la détection atmosphérique des rayons

gamma, en particulier dans le contexte d'un seul télescope, introduit des dégénérescences entre les paramètres à reconstruire. Il existe en effet une forte interdépendance entre l'énergie, la direction et le point d'impact virtuel sur le sol de la particule incidente d'un côté, et les images produites dans la caméra des télescopes de l'autre. Pour une direction et un point d'impact donnés, l'intensité du signal est fortement reliée à l'énergie du rayon gamma [Völk and Bernlöhr, 2009]. La position et la forme du signal produit dans la caméra dépendent quant à elles de la direction et du point d'impact. L'analyse du rayonnement gamma est donc multitâche par nature.

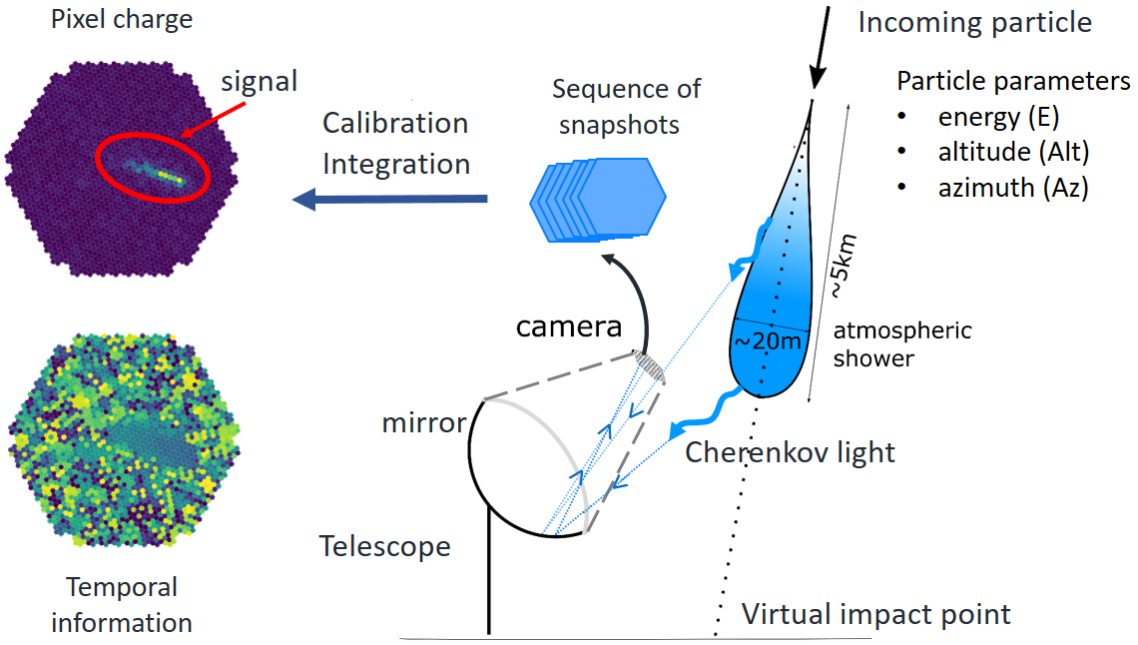


Figure 7.1 – Principe du télescope à effet Cherenkov, ou Imaging Atmospheric Cherenkov Telescope (IACT).

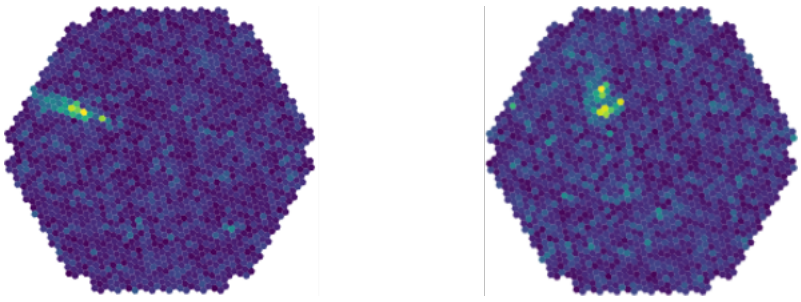


Figure 7.2 – *Gauche:* Évènement gamma d'énergie 0.5 TeV. *Droite:* Évènement proton d'énergie 27 GeV.

L'astronomie gamma au sol est relativement jeune et, depuis la première détection de la nébuleuse du Crabe en 1989 par Whipple, d'autres observatoires ont été construits. Ils sont principalement conçus comme des réseaux de télescopes pour bénéficier de la stéréoscopie. Parmi ceux-ci, le Cherenkov Telescope Array (CTA)[1] constitue la nouvelle génération d'observatoire de l'univers à très haute énergie. Composé de plus de

[1] https://www.cta-observatory.org/

100 télescopes de trois tailles différentes répartis sur 2 sites, CTA aura une sensibilité 10 fois meilleure que les experiences actuelles, telles que le High Energy Stereoscopic System (H.E.S.S.)[2], les télescopes MAGIC[3] ou le Very Energetic Radiation Imaging Telescope Array System (VERITAS)[4], tout en améliorant la précision de la reconstruction. En contrepartie, une fois achevé, CTA produira une énorme quantité de données, plus de 200 Po/an, devant être analysées en temps réel avant d'être réduites et compressées pour transfert et stockage. Les données acquises seront ainsi conservées et traitées à nouveau, chaque année, afin de bénéficier des améliorations des méthodes d'analyse.

Bien que CTA soit en construction, un prototype des plus grands télescopes, les Large-Sized Telescopes (LST), appelé LST1 a été inauguré à l'automne 2018, et a commencé à fournir des données exploitables au printemps 2020. Les LST ont été conçus pour avoir une sensibilité optimisée dans la partie basse du spectre des rayons gamma de très haute énergie. Cette configuration leur permet d'être appropriés pour l'étude des phénomènes transitoires et extragalactiques. Le LST1 est également particulièrement important pour la collaboration CTA car il va permettre aux astrophysiciens de tester leurs outils d'analyse. En effet, la vérité terrain est impossible à obtenir pour les données réelles produites par les télescopes. De ce fait, les modèles d'analyse sont développés à l'aide d'une grande quantité de données de simulation de haute qualité.

7.1.2 Méthodes d'analyse des images Cherenkov

Dans le cas du LST1, les données brutes réelles comme simulées se composent de cubes spatiotemporels de 40 échantillons de 1 ns chacun, appelés *waveforms*. Dans les deux cas, ces waveforms sont calibrées et intégrées à l'aide de la bibliothèque lstchain v0.6.3 [Lopez-Coto et al. for CTA LST project, 2021] dans sa configuration standard. Les données d'entrée des modèles d'analyse sont alors composées de deux canaux, le premier représentant la quantité de photoélectrons reçue par pixel (l'intensité), le second contenant une information temporelle sur le développement de la gerbe, définie également par pixel. Plus précisément, cette information temporelle représente, pour chaque pixel, la position temporelle du pic d'intensité durant le développement de la gerbe.

Actuellement, la méthode d'analyse la plus répandue repose sur l'extraction des paramètres géométriques de la gerbe proposée par A. M. Hillas [Hillas, 1985] et l'analyse de ces paramètres par une méthode multivariée [Bock et al., 2004, Albert et al., 2008b, Ohm et al., 2009, Fiasson et al., 2010]. Cette méthode est appelée *Hillas + RF* dans cette thèse. Comme indiqué sur la Figure 7.3, la méthode Hillas fait l'hypothèse que le signal produit par un rayon gamma dans la caméra du télescope est un ellipsoïde. Après une étape de suppression du bruit de fond, les moments de l'ellipse sont caractérisés. Outre l'intensité totale de la gerbe, sont déterminés la position du centroïde de l'ellipse (sa distance au centre de la caméra ainsi que l'angle azimutal φ), les demi-axes majeur et mineur, et l'angle α formé par le grand axe de l'ellipse et l'axe reliant son centroïde au centre de la caméra. Ces paramètres sont ensuite fournis à un algorithme de machine learning de type Random Forest (RF) pour réaliser la reconstruction de l'évènement. Dans le cas d'une observation stéréoscopique des évènements (par un réseau de télescopes), la direction incidente du rayon gamma détecté peut être déterminée géométriquement : les images produites par

[2] http://www.mpi-hd.mpg.de/hfm/HESS/pages/about/telescopes
[3] http://www.magic.iac.es/
[4] https://veritas.sao.arizona.edu/

chaque télescope sont superposées dans un repère altitude - azimut et la position de la source correspond alors à l'intersection des grands axes des ellipsoïdes.

En réalité, la méthode la plus performante, dite *template*, consiste en la comparaison pixel à pixel, à l'aide d'une fonction de vraisemblance, des images produites par les télescopes avec une très grande banque de modèles [de Naurois and Rolland, 2009, Parsons and Hinton, 2014], la meilleure comparaison fournissant alors les caractéristiques recherchées. Cependant, l'application de ces méthodes à CTA fait face à des limites, en matière soit de sensibilité (Hillas + RF), soit de capacités de calcul (template), le nombre de comparaisons à effectuer pouvant devenir extrêmement important. Il est donc nécessaire d'explorer d'autres pistes.

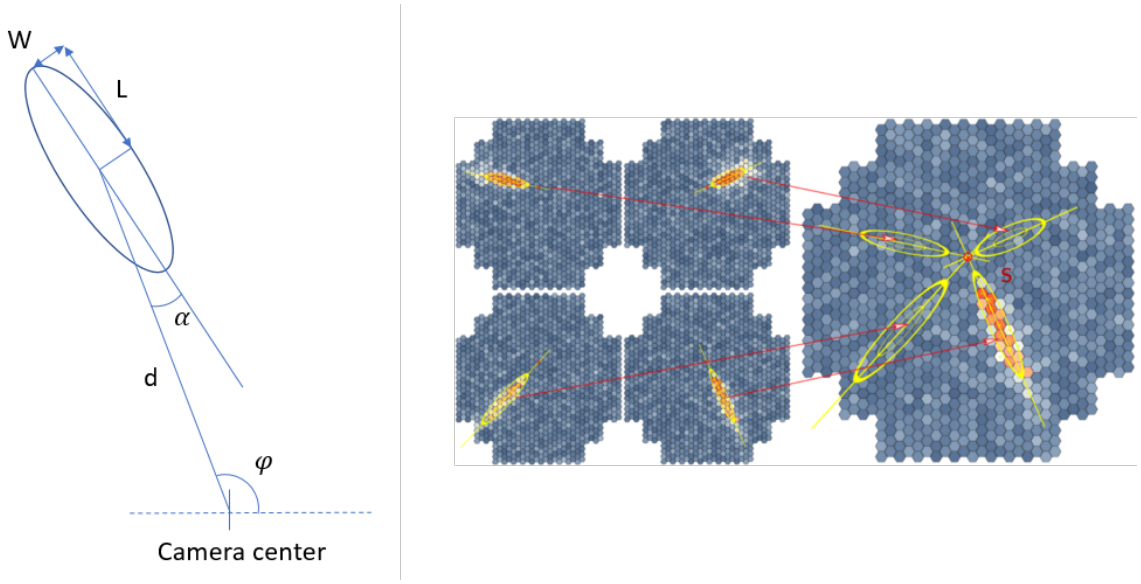


Figure 7.3 – *Gauche :* Extraction des moments de l'ellipse produite par un rayon gamma. *Droite :* La stéréoscopie permet de déterminer géométriquement la direction de la particule incidente. Source : [Völk and Bernlöhr, 2009].

En nous appuyant sur les récentes avancées des réseaux de neurones artificiels, nous proposons dans cette thèse une nouvelle approche d'apprentissage profond pour analyser les données de CTA, en particulier celles du LST1. Nous présentons une architecture multitâche, γ-PhysNet, inspirée de la physique qui réalise la reconstruction complète des évènements gamma. Celle-ci bénéficie d'un mécanisme d'attention la rendant plus robuste aux conditions d'initialisation. Sur les données de simulation utilisées pour préparer les algorithmes d'analyse, γ-PhysNet obtient des résultats significativement meilleurs, tant en sensibilité qu'en résolution spatiale, que la méthode standard Hillas + RF. Le gain de sensibilité à basse énergie pourrait permettre d'améliorer l'étude des phénomènes transitoires. Par ailleurs, nous présentons une analyse préliminaire du réseau proposé à l'aide d'une méthode d'explicabilité visuelle afin de mieux comprendre son comportement.

7.2 Réseaux neuronaux profonds

Depuis le triomphe d'AlexNet [Krizhevsky et al., 2012] lors de la compétition ILSVRC2012 [Russakovsky et al., 2015], l'apprentissage profond a émergé comme l'approche dominante pour résoudre de nombreux problèmes de vision par ordinateur tels que la classi-

fication d'images [Touvron et al., 2019], la segmentation sémantique [Yuan et al., 2019] ou la détection d'objets [Zhang et al., 2020]. Le domaine de l'astrophysique ne fait pas exception [Kim and Brunner, 2016, Brunel et al., 2019]. En particulier, l'apprentissage profond a été exploré pour l'analyse des images produites par des IACT, de l'analyse de muons [Feng et al., 2016] à la reconstruction d'évènements gamma.

7.2.1 Apprentissage profond pour l'astronomie gamma

[Holch et al., 2017] proposent un réseau de neurones convolutif de faible profondeur (3 couches de convolution) pour l'analyse des données stéréoscopiques acquises par les quatre télescopes de l'expérience H.E.S.S.. La méthodologie adoptée consiste en l'optimisation d'un modèle pour chaque tâche à réaliser : classification gamma/proton, régression de l'énergie et régression de la direction de la particule incidente. Pour combiner les informations provenant des quatre télescopes, les images produites sont sommées pixel à pixel pour n'en former qu'une seule, de façon analogue à la reconstruction de la direction dans la méthode Hillas + RF. Bien que leur modèle obtienne de bonnes performances pour la séparation des évènements gamma du bruit de fond, [Holch et al., 2017] ne fournissent que des résultats préliminaires pour les tâches de régression. Dans la continuité de ces travaux, [Shilon et al., 2019] proposent le modèle CRNN pour la classification gamma/proton, intégrant une couche récurrente de type LSTM [Cheng et al., 2016]. Placée après trois couches de convolution similaires au modèle de Holch, cette couche récurrente vise à améliorer le traitement de la stéréoscopie. Avec cette stratégie, au lieu d'être sommées, les images des quatre télescopes sont présentées au réseau comme une séquence, ordonnée selon l'amplitude totale des images. En revanche, pour la reconstruction de la direction, les auteurs ont adopté une approche différente. Les images provenant des quatre télescopes sont combinées en une représentation multicanal, une image par canal. Le modèle proposé est par ailleurs plus profond, composé de cinq ensembles de deux convolutions suivies d'un sous-échantillonnage. Les bonnes performances obtenues sur les données de simulation ne sont cependant pas reproduites pour l'analyse de données réelles. Par ailleurs, ces travaux ne proposent pas de reconstruction de l'énergie. Afin de résoudre la différence de performance observée entre l'analyse des simulations et celle des données réelles, [Parsons and Ohm, 2020] proposent de combiner la représentation latente produite par la partie convolutive du modèle avec les paramètres géométriques extraits par la méthode standard. Cependant, l'amélioration apportée par leur méthode par rapport aux travaux de [Shilon et al., 2019] n'est pas discutée.

Concernant CTA, des travaux préliminaires ont été menés. [Nieto et al., 2017] investiguent la séparation des évènements gamma des protons dans le cas d'un seul télescope grâce à deux réseaux très profonds basés sur des architectures récentes initialement proposées pour des problématiques de reconnaissance d'image multimédia, un ResNet-50 [He et al., 2016b] et un Inception V3 [Szegedy et al., 2015]. Ces deux modèles obtiennent des résultats similaires, cependant nettement moins bons que ceux obtenus avec une chaîne d'analyse standard telle que Hillas + RF. D'un autre côté, [Mangano et al., 2018] testent des réseaux convolutifs beaucoup plus simples pour la classification des particules et la régression de leur énergie et direction, chaque tâche étant effectuée à l'aide d'un modèle dédié. Cette analyse de données stéréoscopiques, reprenant la méthode combinaison des images proposée par [Holch et al., 2017] et réalisée dans un contexte très contraint de forte sélection des données, n'a pas permis d'obtenir de meilleurs résultats que la méthode standard.

Bien que tous ces travaux présentent des résultats prometteurs, en particulier pour la classification gamma/proton, tous traitent les différentes tâches de la reconstruction des évènements gamma comme des problèmes indépendants. Leur approche simple tâche ne prend pas en compte la forte interdépendance des paramètres à reconstruire.

7.2.2 Apprentissage multitâche

L'interdépendance des paramètres à reconstruire rend la reconstruction des évènements gamma à partir d'images provenant d'IACT multitâche par nature. L'apprentissage multitâche [Caruana, 1997] est un paradigme consistant à apprendre toutes les tâches simultanément, à l'aide d'un modèle commun plutôt qu'en dédiant un modèle différent à chaque tâche. L'approche multitâche permet d'augmenter la capacité de généralisation du modèle en l'aidant à se concentrer sur les caractéristiques pertinentes pour toutes les tâches. Cette approche permet ainsi d'éviter la surspécialisation pour une tâche particulière des paramètres partagés [Baxter, 1997].

La topologie de réseau la plus utilisée, nommée *hard parameter sharing*, consiste, ainsi qu'illustrée en Figure 7.4, en la mise en commun entre toutes les tâches d'une partie du modèle, généralement l'encodeur [Kendall et al., 2018, Luvizon et al., 2018, Ren and Jae Lee, 2018] ou les premières couches de celui-ci [Iizuka et al., 2016]. Les parties du réseau spécifiques à chaque tâche sont généralement composées de couches complètement connectées, mais peuvent également l'être de couches de convolution pour certains problèmes de vision par ordinateur tels que la segmentation sémantique [Xu et al., 2018]. L'autre topologie de modèle possible est appelée *soft parameter sharing*. Dans ce cas, chaque tâche est apprise par un réseau différent. L'ajout de couches supplémentaires permet alors de partager l'information entre les différentes tâches [Cao et al., 2018] ou de contraindre les poids de certaines couches spécifiques de chaque réseau vers des distributions similaires, par exemple en régularisant la norme des tenseurs de paramètres concaténés par couche entre les différents réseaux [Yang and Hospedales, 2017]. Un exemple d'architecture *soft parameter sharing* est donnée en Figure 7.5.

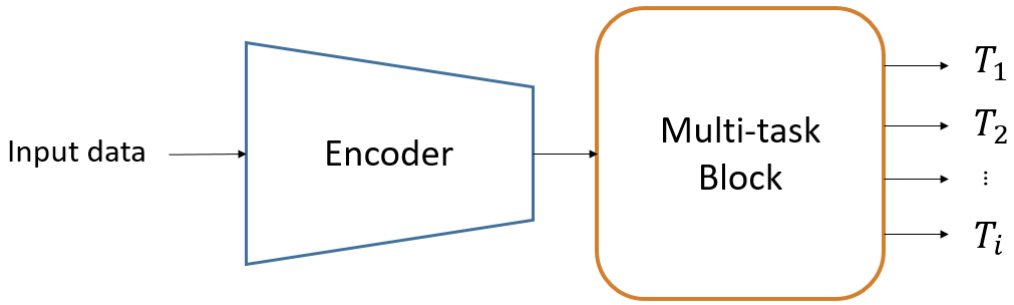


Figure 7.4 – Exemple d'architecture *hard parameter sharing*. L'encodeur est partagé par toutes les tâches. Le bloc multitâche défini le chemin spécifique à chaque tâche.

Si en théorie le multitâche permet de réduire le risque de surapprentissage, il est en pratique fondamental d'équilibrer les différentes tâches lors du calcul de la fonction de coût afin d'éviter qu'une tâche plus simple à apprendre ne prenne l'ascendant sur les autres. Dans la plupart des travaux sur l'apprentissage profond multitâche [Kokkinos, 2017, Han et al., 2017, Luvizon et al., 2018, Ren and Jae Lee, 2018], lorsque précisé,

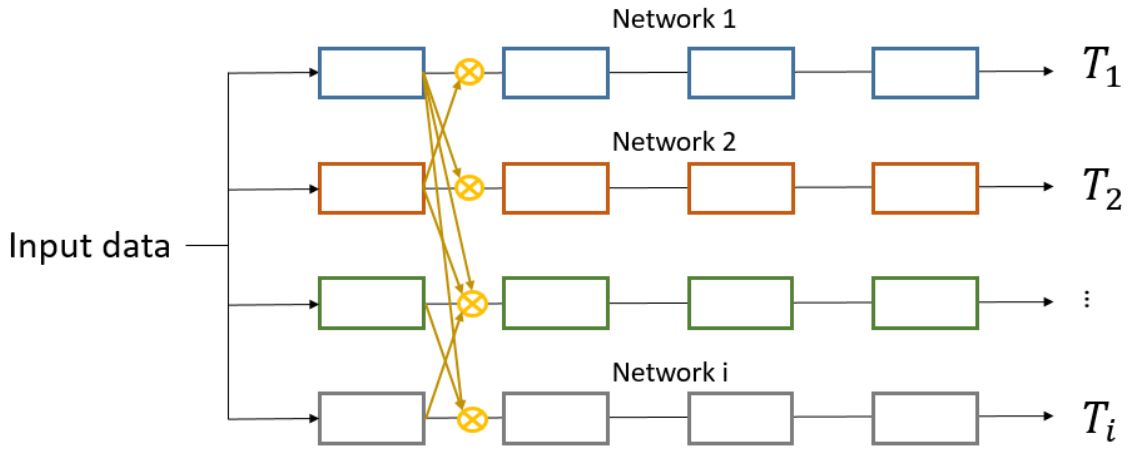


Figure 7.5 – Exemple d'architecture *soft parameter sharing*. Chaque tâche est apprise par un réseau différent, tous les réseaux ayant toutefois la même structure. L'information est échangée entre les tâches grâce à un opérateur spécifique qui combine les cartes de caractéristiques produites par le premier étage de chaque réseau. Pour plus de clarté, les flux de ces cartes de caractéristiques ne sont pas toutes représentées.

cet équilibrage est fait manuellement. Cela nécessite cependant un important processus d'optimisation. Il existe néanmoins des méthodes adaptatives permettant d'équilibrer automatiquement l'importance des tâches. [Kendall et al., 2018] modélisent l'incertitude homoscédastique de chaque tâche comme une tâche supplémentaire, et l'utilisent comme un intermédiaire pour les équilibrer. Suivant une approche différente, [Chen et al., 2018] proposent de pondérer chaque tâche pour que le gradient de leur fonction de coût individuelle par rapport à la dernière couche commune ait une amplitude similaire à celle des autres tâches. Cette méthode revient à pénaliser les tâches prédominantes pour encourager les plus difficiles à apprendre. [Guo et al., 2018] utilisent des signaux de progrès de l'apprentissage aussi appelés indicateurs clés de performance, tels que la précision moyenne, plutôt que l'erreur calculée par la fonction de coût de chaque tâche, afin de donner la priorité aux cas difficiles. Cette hiérarchisation est effectuée à deux niveaux : du point de vue de la tâche à apprendre d'une part et de l'exemple d'apprentissage lui-même d'autre part. [Sener and Koltun, 2018] considèrent l'apprentissage profond multi-tâche comme un problème d'optimisation multiobjectif. Leur méthode permet d'atteindre l'optimalité Pareto pour les poids équilibrant chaque tâche.

7.2.3 Mécanismes d'attention

Du fait des dégénérescences introduites dans les paramètres par la détection atmosphérique des rayons gamma, des informations clés permettant une bonne reconstruction des évènements peuvent se trouver dans le contour et la répartition de l'intensité du signal présent dans l'image. Les mécanismes d'attention peuvent aider le modèle à capturer ces détails subtils et ainsi améliorer les performances. En effet, à partir d'un contexte défini, ces mécanismes aident le modèle à se concentrer sur les caractéristiques pertinentes des données. D'un point de vue biologique, nous pouvons faire le parallèle avec le système visuel humain qui se concentre sur les parties pertinentes de son champ de vision afin d'aider à la perception [Saenz et al., 2002, Boynton, 2005]. En apprentissage profond, ces mécanismes d'attention apprennent les dépendances de courte et longue étendue, suivant

le contexte pris en compte, présentes dans les données.

Les mécanismes d'attention ont fait leur apparition dans le domaine du traitement du langage naturel [Bahdanau et al., 2015]. Ce sont les composants principaux des modèles Transformer [Vaswani et al., 2017, Devlin et al., 2019] qui obtiennent les meilleures performances dans la traduction automatique et la description d'images. Ces dernières années, les mécanismes d'attention sont devenus des composants essentiels des réseaux de neurones, y compris dans le traitement d'images. Dans ce domaine, nous pouvons distinguer trois types d'attention :

- l'attention spatiale, ou de pixel à pixel,
- l'attention entre canaux,
- la combinaison des deux.

A travers l'auto-attention restreinte, [Parmar et al., 2018] généralisent l'architecture Transformer pour la génération d'image. Ce mécanisme d'attention spatiale apprend les dépendances pertinentes entre les pixels d'un voisinage local. Alors que pour les applications de vision par ordinateur les mécanismes d'attention sont généralement utilisés en complément des couches de convolution, [Parmar et al., 2019] proposent un modèle composé entièrement de couches d'auto-attention locale pour la classification d'images et la détection d'objets. D'un autre côté, [Wang et al., 2018] étendent ce concept local à l'auto-attention globale afin de prendre en compte l'image entière en tant que contexte de l'apprentissage des dépendances entre pixels. Ce mécanisme est appliqué à la classification de vidéos, la segmentation d'image et l'estimation de pose. [Zhang et al., 2019] adaptent l'auto-attention globale pour les réseaux adverses génératifs. Cette méthode est appelée *Self-Attention* ou SA dans cette thèse.

Au contraire de l'attention spatiale, [Hu et al., 2018] proposent un mécanisme pour apprendre les dépendances entre les canaux des images ou des représentations intermédiaires (cartes de caractéristiques) dans le réseau de neurones. Ce mécanisme, appelé *Squeeze-and-Excitation*, met en valeur les canaux pertinents pour la décision du modèle grâce à trois opérations. Ainsi qu'illustré en Figure 7.6, l'opération (i) *squeeze* produit d'abord un descripteur par canal des données d'entrée. Ensuite, l'opération (ii) *excitation* réalise un recalibrage adaptatif de ces descripteurs qui sont ensuite utilisés pour (iii) pondérer les canaux des données d'entrée. A noter que l'excitation agit comme un goulot d'étranglement contrôlé par un ratio de réduction, paramètre important de la méthode.

Les attentions spatiales et par canal peuvent être associées, comme dans le mécanisme appelé *Dual Attention* [Sun et al., 2020]. Celui-ci est composé, comme illustré en Figure 7.7, d'un chemin d'attention par canal, consistant en un bloc Squeeze-and-Excitation, et d'un chemin d'attention spatiale relativement simple. Ce dernier comporte deux convolutions destinées à réduire le nombre de canaux à 1, suivies d'une fonction sigmoïde générant une carte d'activation comprenant le poids de chaque pixel (compris entre 0 et 1). Afin de ne permettre uniquement au module de mettre en valeur des pixels, et non d'en éteindre, le scalaire 1 est ajouté à chaque pixel de la carte d'activation, produisant ainsi une carte d'attention spatiale. Cette dernière carte est utilisée ensuite pour recalibrer la sortie du chemin d'attention par canal. D'autres mécanismes combinant attentions spatiales et par canal existent, tel que SCA-CNN [Chen et al., 2017] conçu pour la génération de légende d'images, ou CBAM [Woo et al., 2018] dont le chemin d'attention spatiale possède une moindre expressivité que celui de la Dual Attention.

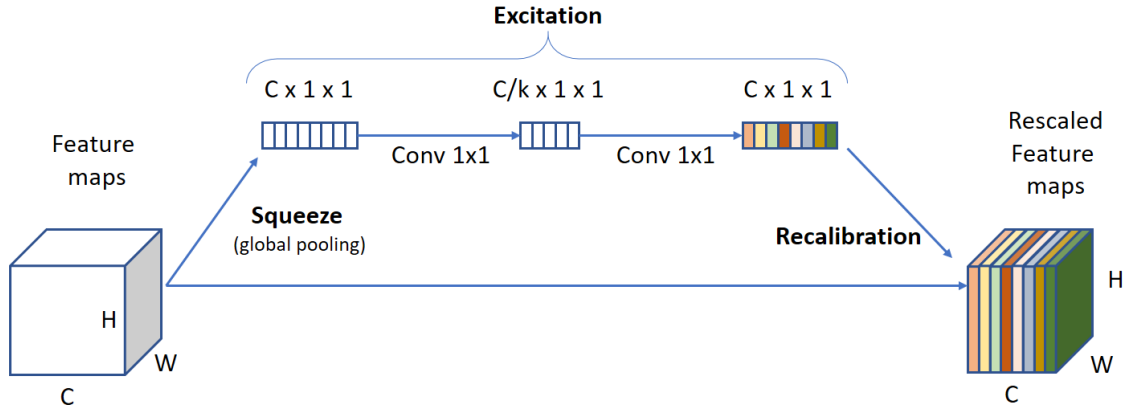


Figure 7.6 – Mécanisme d'attention par canal *Squeeze-and-Excitation*. Le paramètre k contrôle le goulot d'étranglement. Source : [Hu et al., 2018].

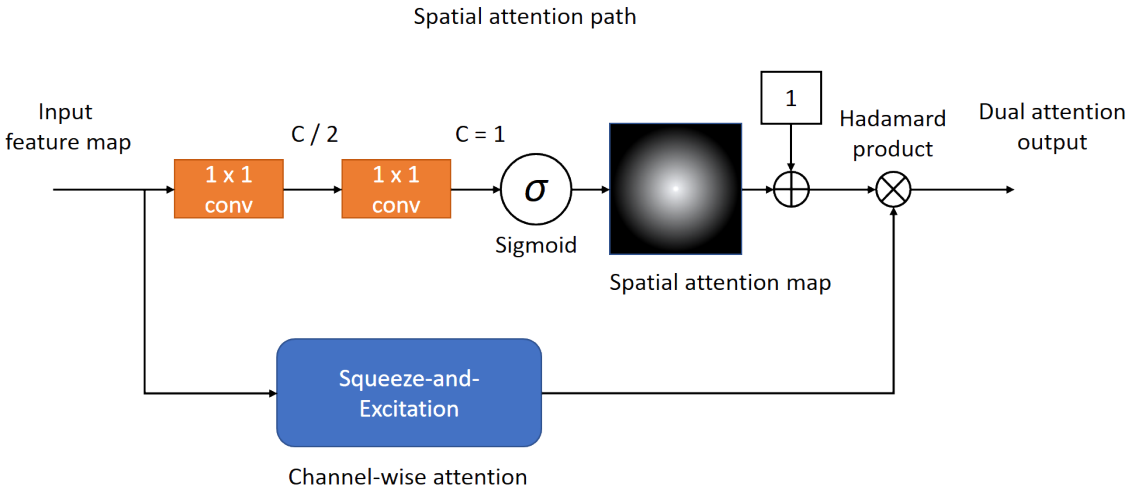


Figure 7.7 – Mécanisme *Dual Attention* combinant attention spatiale et par canal. Source : [Sun et al., 2020].

En aidant le modèle à se concentrer sur les caractéristiques pertinentes, les mécanismes d'attention permettent d'améliorer la robustesse des réseaux de neurones. Par ailleurs, ils peuvent également fournir un aperçu de la façon dont le modèle prend ses décisions. En particulier, l'observation des cartes d'attention spatiale permet d'identifier les pixels mis en valeur, et donc prenant une plus grande part dans le résultat fourni par le modèle.

7.2.4 Explicabilité des réseaux de neurones

Tandis que les systèmes experts traditionnels sont hautement interprétables et explicables, les réseaux de neurones profonds sont souvent perçus comme des *boîtes noires*. L'effort engagé par la communauté de l'apprentissage profond pour progresser dans ces domaines a considérablement augmenté ces dernières années. Certaines méthodes explorent le rôle de neurones individuels ou de la combinaison linéaire de neurones par ablation [Zhou et al., 2018, Morcos et al., 2018b] ou optimisation [Olah et al., 2017]. D'autres estiment l'importance des caractéristiques d'entrée pour l'activation d'une sortie particulière. Ces méthodes produisent des cartes d'importance [Springenberg et al., 2015, Cao et al., 2015, Srinivas and Fleuret, 2019], également appelées cartes de chaleur [Selvaraju et al., 2017], pour la visualisation du comportement des réseaux de neurones. Dans cette thèse, nous nous appuyons sur l'une d'entre elles, Grad-CAM [Selvaraju et al., 2017], pour l'analyse de l'impact de l'attention sur le comportement de l'architecture proposée. Grad-CAM est une méthode de localisation de ces caractéristiques discriminant fortement les sorties du modèle. Elle produit des cartes de chaleur des pixels les plus importants dans la décision du modèle, ces cartes étant spécifiques à chaque sortie du réseau de neurones. Grad-CAM est applicable à tout type d'architecture de réseau. Les cartes de chaleur sont calculées à partir de la dernière couche de convolution du modèle. Dans le cadre de l'apprentissage multitâche, Grad-CAM est donc particulièrement adaptée aux architectures de type *hard parameter sharing* décrites en Section 7.2.2 et qui partagent un encodeur convolutif entre toutes les tâches.

En outre, comme déjà évoqué dans la section précédente, lorsque les modèles intègrent des mécanismes d'attention, l'observation des cartes d'attention, notamment spatiales, peut également donner des indications à propos des régions d'intérêt sur lesquelles le modèle se concentre, et ainsi peut aider à expliquer les décisions du réseau.

7.3 Convolutions Indexées

L'opérateur de convolution est un composant majeur des réseaux de neurones artificiels modernes. La convolution a joué un rôle crucial dans le développement de l'apprentissage profond, et dans son essor au cours des 10 dernières années. En effet, associée au pooling, la convolution permet de réduire le nombre de paramètres du réseau, diminuant les coûts de calcul, et donne au modèle des propriétés intéressantes telles que l'invariance en translation.

Dans les bibliothèques d'apprentissage profond comme tensorflow ou PyTorch, les convolutions sont implémentées uniquement pour les images conventionnelles, possédant des pixels carrés (ou rectangulaires). Or, ainsi qu'illustré en Figure 7.1 et 7.2, les images produites par les télescopes de CTA sont non conventionnelles. En particulier, la caméra

du LST1 est de forme hexagonale, mais surtout, possède des pixels hexagonaux.

Traditionnellement, pour appliquer les techniques d'apprentissage profond aux images à pixels hexagonaux, ces dernières sont soit ré-échantillonnées, soit subissent un décalage afin d'être transformées en images à pixels carrés. Ces méthodes présentent toutefois des inconvénients importants, tels qu'une augmentation de la taille des images, l'ajout de pixels en réalité inexistant ou l'introduction de distorsions dans les images.

Pour nous affranchir de ces problèmes, nous avons proposé la convolution indexée (ainsi que le pooling indexé). Cette méthode s'appuie sur l'implémentation GEMM (General Matrix Multiplication) de la convolution, transformant cette dernière en multiplication de matrice grâce à un réarrangement des données. La convolution indexée permet d'appliquer l'opérateur de convolution à tout type d'organisation de pixels, et n'est donc pas limitée aux pixels hexagonaux. Nos expériences ont montré que l'utilisation de convolutions indexées pour le traitement des images de CTA permet d'obtenir d'aussi bonnes, voir de meilleures performances que les méthodes traditionnelles suivant les tâches à résoudre et la profondeur du modèle utilisé.

7.4 γ-PhysNet : une architecture multitâche pour la reconstruction complète des évènements gamma

Afin de réaliser la reconstruction complète (type de particule, énergie et direction) des évènements gamma à partir d'images Cherenkov, nous proposons γ-PhysNet, une architecture profonde multitâche. C'est une architecture de type *hard parameter sharing*, comme illustré en Figure 7.8. Elle permet de réduire le nombre de paramètres du modèle et de réduire le temps de calcul. La vitesse d'inférence est en effet cruciale pour l'analyse des images de CTA du fait de la quantité de données qui s'accumule chaque année et qui doit être traitée à nouveau régulièrement. γ-PhysNet est composé d'un encodeur et d'un bloc multitâche inspiré de la physique du phénomène. À partir des données produites par le LST1, cartes d'intensité et cartes d'information temporelle, γ-PhysNet réalise en même temps la séparation des gammas du bruit de fond (classification gamma/proton), et la reconstruction des paramètres de la particule détectée, c'est-à-dire la régression de son énergie, sa direction incidente, mais également de son point d'impact virtuel comme tâche auxiliaire.

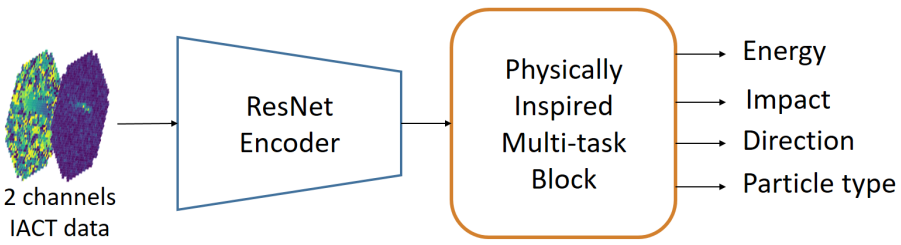


Figure 7.8 – Architecture γ-PhysNet pour l'analyse d'images d'IACT.

7.4.1 Encodeur

Une étude approfondie a permis de définir l'encodeur de γ-PhysNet. Nous avons comparé des réseaux convolutifs de faible profondeur (3, 4 et 5 couches), un réseau neuronal pro-

fond (le VGG-16 [Simonyan and Zisserman, 2015]) ainsi que des réseaux très profonds, dont des DenseNets [Huang et al., 2017] avec un nombre de couches variable (jusqu'à 97) et différents ResNets [He et al., 2016b, He et al., 2016a]. Pour ces derniers, nous avons notamment évalué la version CIFAR-10. Les images produites par le LST1 sont en effet comparables en taille à celle du jeu de données CIFAR-10 (1855 pixels pour le LST1 contre 1024 pour CIFAR-10). Les ResNets se sont montrés globalement supérieurs, et parmi eux le ResNet-56 en version CIFAR-10 avec pré-activation complète (les couches de normalisation et d'activation sont placées avant la convolution) a obtenu les meilleurs résultats en termes de performance et de stabilité de l'entraînement.

Nous adaptons donc la partie convolutive de ce ResNet-56 comme encodeur de γ-PhysNet. En effet, les images générées par le LST1 ayant des pixels hexagonaux, afin d'éviter une étape supplémentaire de pré-traitement nous avons implémenté l'encodeur de γ-PhysNet avec des convolutions indexées[5] [Jacquemont et al., 2019a]. Ces dernières permettent d'appliquer la convolution à tout type de maillage en respectant le voisinage réel des pixels. De plus, il n'a pas été clairement démontré l'intérêt de ré-échantillonner les images selon une grille cartésienne pour la performance du modèle [Nieto et al., 2019b].

7.4.2 Bloc multitâche

La conception du bloc multitâche de γ-PhysNet, illustrée en Figure 7.9, est directement inspirée de la physique des tâches à résoudre. Ce bloc, composé de couches complètement connectées, est en effet divisé en deux sous-réseaux, un réseau reposant sur des caractéristiques globales (appelé *global feature network*) et un autre s'appuyant sur des caractéristiques locales (*local feature network*).

Le sous-réseau global est dédié à la régression de l'énergie. En effet, ce paramètre peut être considéré comme global, car pour une direction et un point d'impact donnés, le nombre total de photoélectrons captés par la caméra est environ proportionnel à l'énergie de la particule gamma ayant généré la gerbe [Völk and Bernlöhr, 2009]. Ce sous-réseau commence donc par une moyenne globale réduisant les cartes de caractéristiques produites par l'encodeur à un descripteur par canal. S'ensuit une couche complètement connectée de 256 neurones complétée par une activation de type ReLU, et une dernière couche complètement connectée qui produit la régression de l'énergie.

D'un autre côté, la classification gamma/proton et la régression de la direction et du point d'impact reposent sur des informations locales et spatiales. En première approximation :

- la position du signal dans l'image dépend de la direction,
- son orientation dépend du point d'impact,
- sa forme (allongée ou non) dépend à la fois de la direction et du point d'impact,
- la forme de son contour ainsi que sa distribution d'intensité dépendent du type de particule.

[5] https://github.com/IndexedConv/IndexedConv

Pour exploiter ces dépendances, le sous-réseau local conserve toutes les caractéristiques des cartes produites par l'encodeur en les vectorisant. Ensuite, la classification gamma/proton est réalisée directement à partir de ce vecteur à l'aide d'une couche complètement connectée. Les régressions de la direction et du point d'impact partagent quant à elles une première couche complètement connectée de 256 neurones suivie d'une activation de type ReLU. Enfin, chaque tâche est réalisée à l'aide d'une couche complètement connectée propre de 2 neurones, la direction étant reconstruite par son altitude et son azimut, et le point d'impact par ses coordonnées x et y dans le repère du réseau de télescopes (pour le site de CTA concerné, par exemple La Palma).

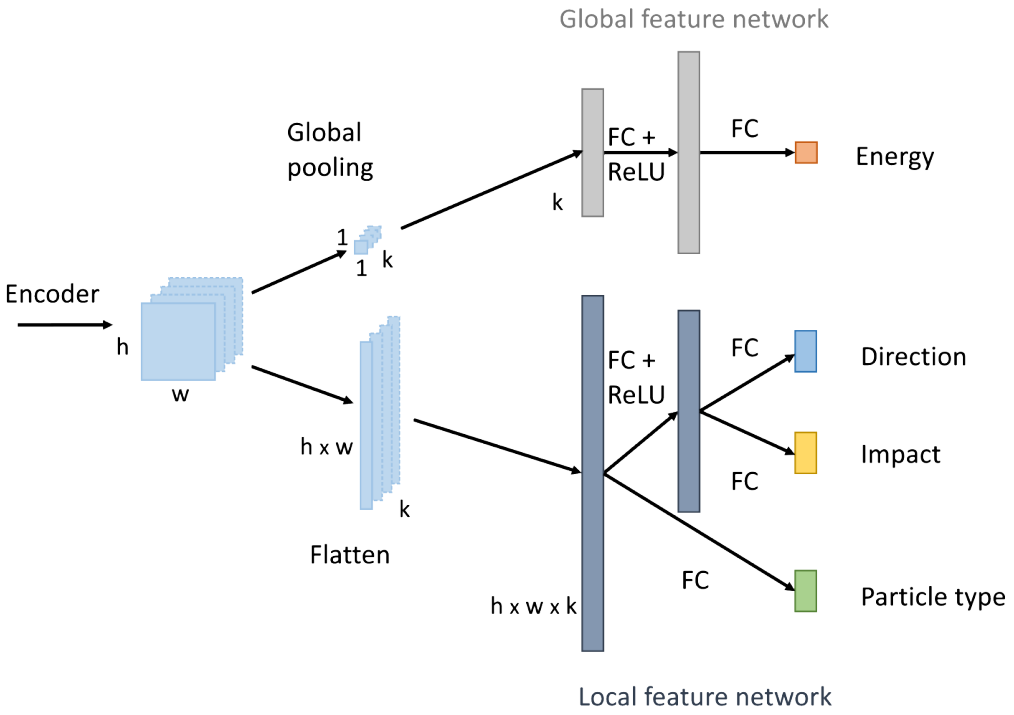


Figure 7.9 – Bloc multitâche inspiré de la physique de la reconstruction.

Par ailleurs, à la recherche d'un modèle qui capture au mieux la physique sous-tendant à la reconstruction des évènements gamma, nous avons aussi évalué d'autres architectures, plus simples et plus complexes, pour ce bloc multitâche. Aucune n'a cependant donné de meilleures performances que celle présentée ici.

7.4.3 Augmentation de l'encodeur avec de l'attention

Pour mieux adapter notre modèle au contexte de l'analyse des données produites par des IACT, nous augmentons l'encodeur de γ-PhysNet avec de l'attention. Cette dernière doit permettre au réseau d'apprendre à se concentrer sur les caractéristiques pertinentes dans les données, et ainsi mieux exploiter l'information relative au signal parmi le bruit des images. Ceci pourrait également améliorer le traitement des gerbes tronquées sur les bords de la caméra qui peuvent perturber le processus d'apprentissage.

Comme précisé précédemment, l'encodeur de γ-PhysNet est la partie convolutive du ResNet-56 en version CIFAR-10. Ainsi qu'illustrée en Figure 7.10, celle-ci est composée d'une couche de convolution initiale, puis de trois étages résiduels dont la première couche réalise un sous-échantillonnage à l'aide d'une convolution. Le temps d'inférence

étant déterminant pour l'analyse des données de CTA, il serait coûteux d'insérer les blocs d'attention après chacune des 55 convolutions de l'encodeur. Nous insérons donc ces blocs après chaque étage, afin de bénéficier de l'attention à chacune des échelles de tailles des cartes de caractéristiques tout en ayant un coût en temps de calcul raisonnable.

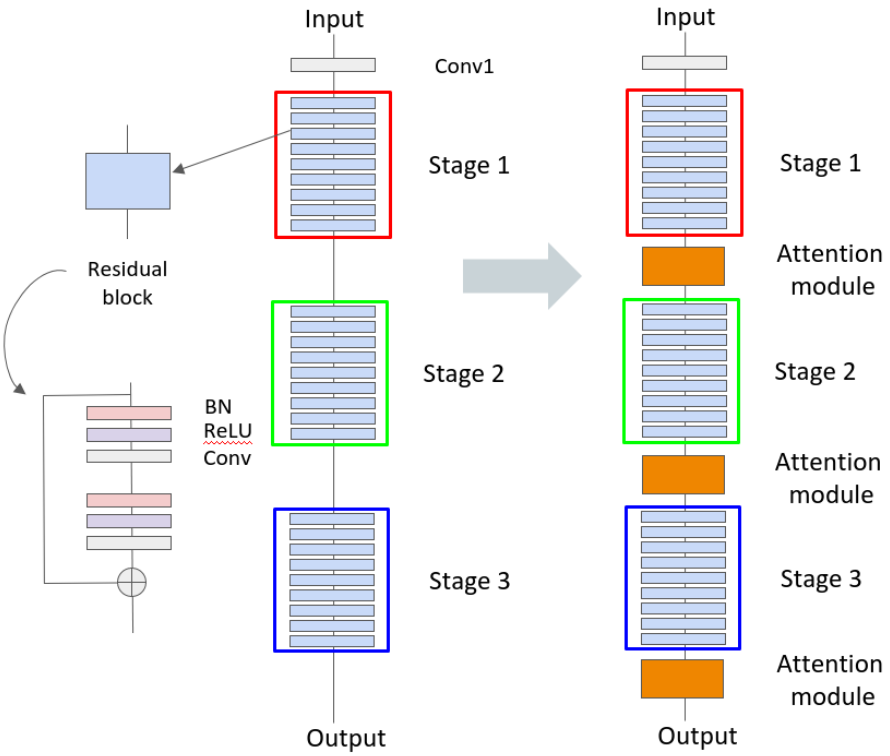


Figure 7.10 – Ajout de mécanismes d'attention à l'encodeur de γ-PhysNet. Les modules d'attention sont insérés après chaque étage du ResNet-56.

Nous avons comparé dans une étude de grande envergure trois mécanismes d'attention pour γ-PhysNet en faisant varier l'hyperparamètre contrôlant leur goulot d'étranglement : le Squeeze-and-Excitation, le Self-Attention et le Dual Attention. Le Squeeze-and-Excitation et le Dual Attention ont constamment obtenu de meilleurs résultats que le Self-Attention, sans pouvoir les départager entre eux. Dans cette thèse nous nous concentrons sur le Dual Attention, composé d'un chemin d'attention par canal et d'un chemin d'attention spatiale. Dans une démarche de recherche d'explicabilité du modèle, l'observation des cartes d'attention produites par ce dernier peut en effet donner des indications sur la manière dont le réseau prend ses décisions. La version du modèle augmentée des blocs Dual Attention est appelée γ-PhysNet DA.

7.5 Évaluation des performances

Afin de valider notre approche, nous évaluons les performances de γ-PhysNet (sans attention) sur les données de simulations du LST1 dans un contexte d'analyse simple télescope. Nous démontrons tout d'abord l'apport du multitâche pour la reconstruction et comparons les résultats de notre architecture avec ceux de l'approche standard Hillas + RF. Ensuite, nous mesurons l'impact de l'attention sur le comportement du modèle,

notamment sur sa robustesse, et nous comparons deux méthodes d'explicabilité visuelle pour comprendre cet impact.

7.5.1 Jeu de données

La vérité terrain étant impossible à obtenir pour les données réelles en astronomie gamma, l'évaluation des performances de γ-PhysNet est réalisée à l'aide de simulation de grande qualité, grâce à une très bonne connaissance du modèle physique de développement des gerbes de particules. La simulation des évènements se déroule en deux temps. La gerbe de particules est d'abord générée à partir des paramètres de la particule incidente à l'aide d'une méthode Monte-Carlo. L'outil utilisé est CORSIKA [Heck et al., 1998]. Ensuite, la réponse du télescope est simulée grâce au logiciel sim_telarray [Bernlöhr, 2008] en s'appuyant sur une modélisation de ses systèmes optique et électronique. Il est intéressant de noter que cette modélisation est mise à jour grâce aux retours d'expérience du fonctionnement du télescope. Le jeu de données de référence utilisé dans cette thèse est le LST4 mono-trigger Production (du 15/04/2019). Il résulte d'une production à grande échelle générée par la collaboration LST pour la mise en service du LST1, et contient les données générées pour les quatre LSTs du site nord de CTA. Ces données sont composées d'évènements de différents types, dont des gammas diffus, c.-à-d. correspondant à des sources étendues, des gammas provenant d'une source ponctuelle (nommés *point-like*) et des protons (toujours diffus). Les gammas et les protons sont simulés avec des distributions en énergie différentes (bien que suivant toutes deux une loi de puissance d'indice spectral -2), conduisant à un jeu de données déséquilibré en matière de nombre d'évènements par bande d'énergie.

7.5.2 Sélection et préparation des données

La première étape d'analyse de la méthode standard Hillas + RF consiste en la préparation des données afin de supprimer le bruit dans les images et en la sélection des données pour éliminer les évènements trop difficiles à reconstruire. Classiquement, cette sélection repose sur trois critères : la taille du signal après suppression du bruit, son intensité totale, et la fraction du signal située sur le bord de la caméra, indication que celui-ci est tronqué ou non.

Nous adoptons la même approche afin de pouvoir comparer nos modèles à la référence Hillas + RF, à l'exception près que nous ne supprimons pas le bruit des images. La force de l'apprentissage profond est en effet d'extraire de fines informations à partir de l'ensemble des pixels de l'image. Ainsi que nous l'avons expérimenté, supprimer le bruit des images limite la performance du modèle à la qualité de la suppression.

Nous considérons alors des bases de données d'entraînement, de validation et de test communes.

7.5.3 Entraînement des modèles

Pour les expériences présentées dans cette thèse, tous les modèles (réseaux de neurones et Hillas + RF) sont entraînés en utilisant les quatre télescopes du jeu de données afin d'avoir un aperçu plus juste de la variabilité des données. Les modèles sont entraînés sur des gammas diffus pour être capables de reconstruire des évènements provenant de toutes directions dans le champ de vue des télescopes, et sur des évènements protons.

Tous les réseaux de neurones sont entraînés à l'aide des mêmes hyperparamètres. En effet, du fait de la durée des entraînements (entre 10 et 40 heures suivant les sélections appliquées aux données) il n'est pas envisageable à ce stade de l'étude de réaliser une optimisation avancée de chaque modèle. Cependant, en partant des hyperparamètres optimisés pour le ResNet-56, des expériences préliminaires conséquentes ont permis d'identifier un ensemble d'hyperparamètres communs performants utilisés pour les expériences présentées ci-après, et permettant à chaque modèle d'atteindre son meilleur niveau de performance.

Plus précisément, les réseaux neuronaux sont entraînés sur 25 epochs à l'aide de l'optimiseur Adam [Kingma and Ba, 2015]. Le taux d'apprentissage est défini à 10^{-3} et est divisé par 10 toutes les 10 epochs. L'apprentissage est régularisé en appliquant une pénalité de 10^{-4} sur la norme L2 des poids des modèles.

Comme souligné dans la section 7.2.2, il est nécessaire pour les réseaux multitâches d'équilibrer les tâches. Nous utilisons la méthode proposée par [Kendall et al., 2018] et reposant sur l'estimation de l'incertitude liée à chaque tâche. Celle-ci donne en effet les meilleurs résultats pour notre problème. L'incertitude des tâches étant un paramètre appris, nous utilisons de nouveau l'optimiseur Adam, avec cette fois-ci un taux d'apprentissage de 0.025 et une pénalité sur la norme des poids de 10^{-4}.

Pour la tâche de classification (séparation gamma/proton), nous utilisons l'entropie croisée classique comme critère d'optimisation. Pour les tâches de régression (énergie, direction et point d'impact virtuel), le critère sélectionné est une distance de type L1 (erreur moyenne absolue) qui a permis d'obtenir de meilleures performances que l'erreur moyenne au carré (distance L2 classique) dans nos expériences préliminaires. Par ailleurs, afin d'éviter que les protons ne perturbent l'apprentissage de l'énergie, la direction et le point d'impact des gammas, nous employons une stratégie dite de masquage (*masked loss*). Celle-ci consiste en la mise à zéro de l'erreur des tâches de régression calculée par la fonction de coût lorsque la particule est un proton :

$$L_t = \frac{1}{M} \sum_{N} \begin{cases} l_i, & \text{si la particule est un gamma} \\ 0, & \text{sinon} \end{cases} \tag{7.1}$$

avec L_t l'erreur de la tâche t pour un lot de données de taille N, un nombre de d'évènements gamma M dans le lot, et l_i l'erreur pour l'exemple i.

7.5.4 Méthodologie d'évaluation des performances

Pour se conformer à la pratique standard dans l'astronomie gamma, les différents modèles sont évalués sur des gammas provenant d'une source ponctuelle (*point-like*) et des protons, cette configuration correspond aux cas d'utilisation classiques du domaine.

Les performances des différents modèles pour les tâches de régression (énergie et direction) sont mesurées à l'aide de courbes de résolution. Pour l'énergie, la résolution représente, par bande d'énergie, la largeur de l'intervalle autour de 0 contenant 68% de la distribution de l'erreur relative faite par le modèle lors de la prédiction. La résolution angulaire, quant à elle, représente, par bande d'énergie, l'angle dans lequel 68% des gammas reconstruits tombent, relativement à leur vraie direction. Pour ces deux courbes de résolution, des valeurs plus faibles indiquent de meilleurs résultats. Concernant la tâche de classification gamma/proton, la performance globale des modèles est exprimée par l'aire sous la courbe ROC (AUC), tandis que la précision indique la capacité des modèles

à rejeter les protons, et le rappel illustre leur capacité à conserver les gammas. La précision et le rappel sont calculés pour une valeur de seuil par défaut de 0.5 pour la sortie gamma du modèle.

Dans le cadre de l'apprentissage profond, l'initialisation stochastique des poids des réseaux de neurones joue un rôle important dans la performance finale du modèle entraîné. Afin d'avoir une estimation des performances moins biaisée et ne dépendant pas d'une graine aléatoire particulière, nous répétons l'apprentissage de tous les réseaux de neurones six fois, avec des graines différentes. Cela permet en outre de mesurer la variabilité des modèles due à l'initialisation des paramètres, et donc leur robustesse. Pour rendre compte de cette variabilité, les courbes de résolution sont tracées comme des bandes représentant deux écarts-types autour de la moyenne des résultats illustrée par des points. Cette résolution moyenne n'a pas de réalité physique, car la résolution est une mesure statistique de l'erreur d'un modèle particulier. Toutefois, elle donne une tendance de la performance des modèles et est utile pour la lisibilité des résultats.

7.5.5 Intérêt du multitâche et comparaison avec la méthode standard

Pour évaluer l'intérêt de l'apprentissage profond multitâche pour l'analyse des données Cherenkov, à savoir classification gamma/proton et régression de l'énergie et de la direction, nous comparons les performances des approches suivantes :

- γ-PhysNet. L'architecture que nous proposons.
- γ-PhysNet w/o impact. Pour évaluer l'apport du point d'impact en tant que tâche auxiliaire, nous entraînons γ-PhysNet en supprimant cette tâche.
- ResNet-56. Pour démontrer la contribution de l'apprentissage multitâche à la performance de γ-PhysNet, nous entraînons trois ResNet-56, correspondant à l'encodeur de γ-PhysNet, pour résoudre les différentes tâches. La différence principale avec l'architecture décrite par [He et al., 2016b] réside dans le fait que pour les tâches de régression de la direction et de classification gamma/proton, nous remplaçons la moyenne globale finale précédant la dernière couche complètement connectée par une vectorisation des caractéristiques produites par l'encodeur afin de conserver les informations spatiales.
- Hillas + RF. Cette méthode très répandue pour la reconstruction des évènements gamma à partir de données Cherenkov est utilisée comme référence.

Nous ne pouvons pas comparer notre approche avec celles proposées par [Holch et al., 2017], [Shilon et al., 2019], [Parsons and Ohm, 2020] et [Mangano et al., 2018] car celles-ci sont conçues pour l'analyse stéréoscopique des données Cherenkov quand notre architecture réalise l'analyse à partir des données d'un seul télescope. Nous ne pouvons pas non plus comparer nos résultats avec ceux de [Nieto et al., 2017] car ces derniers ont été obtenus pour un type de télescope différent, uniquement pour la tâche de classification et sans utiliser les informations temporelles des données.

Pour l'étude présentée dans cette thèse, nous appliquons une sélection des données telle que décrite dans la Section 7.5.2. Plus précisément, les évènements dont les caractéristiques sont les suivantes sont conservés :

- intensité totale supérieure à 300 photoélectrons,

- au moins un pixel survit à l'opération de suppression du bruit,
- la fraction de signal, en intensité, située sur les deux rangées bordant l'image, est inférieure à 20%.

Ces coupures sont standard en astronomie gamma, quoiqu'un peu fortes à basse énergie. Il en résulte un jeu d'entraînement composé de 388k gammas diffus et 236k protons.

7.5.5.1 Classification gamma/proton

Les résultats reportés en Table 7.1 montrent clairement que, à précision comparable, tous les réseaux neuronaux obtiennent de bien meilleurs rappels que la méthode standard Hillas + RF. En particulier, l'architecture proposée améliore le rappel de 60.1%. Cette amélioration permettra d'augmenter la sensibilité de l'analyse en conservant 60% d'évènements gamma en plus. La contribution du multitâche est également significative comparée à l'approche simple tâche réalisée avec le modèle ResNet-56, avec une fois encore une amélioration du rappel. Toutefois, l'ajout de la régression du point d'impact comme tâche auxiliaire ne semble pas bénéficier à la tâche de classification gamma/proton.

Table 7.1 – Score AUC, précision et rappel de la tâche classification gamma/proton pour les différents modèles

Modèle	AUC	Précision	Rappel
Hillas + RF	0.898	0.956	0.593
ResNet-56	0.954±0.001	0.956±0.001	0.942±0.001
γ-PhysNet	**0.960±0.002**	0.957±0.003	**0.956±0.006**
γ-PhysNet w/o Impact	**0.961±0.002**	**0.960±0.001**	0.949±0.003

7.5.5.2 Régression de l'énergie

Les résultats présentés sur la Figure 7.11 montrent que, comme pour la tâche de classification, tous les modèles à base de réseaux de neurones obtiennent de meilleures performances que la méthode standard Hillas + RF. En particulier, γ-PhysNet réduit l'erreur en énergie jusqu'à 35% à haute énergie, et jusqu'à 75% à 31 GeV (le point de la courbe Hillas + RF est en dehors de la figure). Aux énergies supérieures à 400 GeV, l'approche multitâche sans point d'impact dégrade les performances en termes de résolution et de dispersion. Ceci peut s'expliquer par le fait que l'intensité lumineuse émise par la gerbe électromagnétique est fortement corrélée à l'énergie de la particule détectée. L'intensité détectée par le télescope dépend alors de sa distance au point d'impact virtuel de la particule. L'ajout de la régression du point d'impact en tant que tâche auxiliaire permet à γ-PhysNet d'obtenir des performances comparables à l'approche simple tâche réalisée avec le ResNet-56, avec toutefois une dispersion plus importante à hautes énergies.

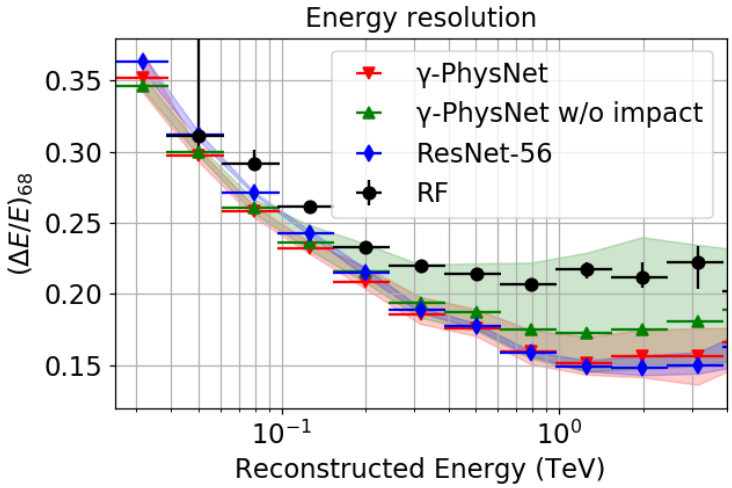


Figure 7.11 – Résolution en énergie en fonction de l'énergie reconstruite dans la bande d'énergie du LST (des valeurs plus faibles indiquent un meilleur résultat). Comparaison des performances des différents modèles.

7.5.5.3 Régression de la direction

Comme pour les tâches de classification gamma/proton et de régression de l'énergie, la Figure 7.12 montre que les réseaux de neurones obtiennent de meilleures performances pour la tâche de régression de la direction que la méthode Hillas + RF. En particulier, γ-PhysNet améliore la reconstruction de 0.03 à 0.3° suivant l'énergie considérée. De plus, l'approche multitâche permet cette fois-ci de réduire l'erreur de reconstruction jusqu'à 0.08° à basse énergie. Les deux approches multitâches obtiennent des résultats similaires, l'ajout de la régression du point d'impact permettant de réduire la variabilité due à l'initialisation des poids.

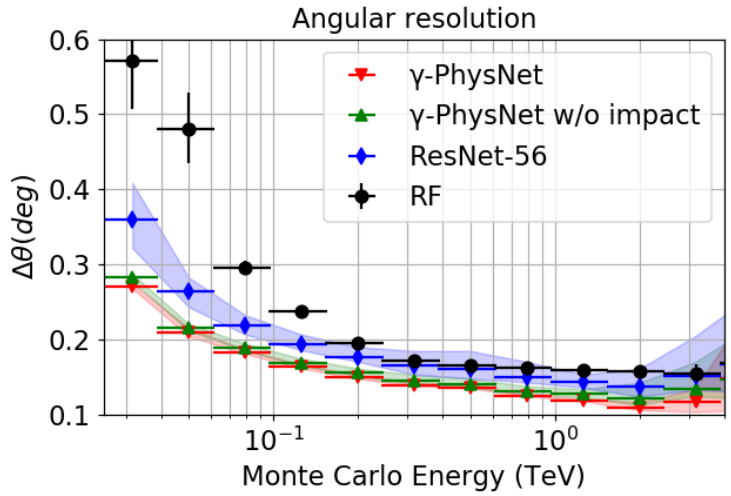


Figure 7.12 – Résolution angulaire en fonction de l'énergie simulée dans la bande d'énergie du LST (des valeurs plus faibles indiquent un meilleur résultat). Comparaison des performances des différents modèles.

7.5.5.4 Conclusion

L'analyse des résultats pour les trois tâches de la reconstruction des évènements gamma à partir de données Cherenkov montre que l'architecture proposée obtient clairement de meilleures performances que la méthode standard Hillas + RF. L'approche multitâche quant à elle permet de réduire l'erreur sur la régression de la direction, en conservant des performances similaires sur la régression de l'énergie grâce à l'ajout du point d'impact virtuel comme tâche auxiliaire. Cet ajout permet également de réduire la variabilité introduite par l'initialisation des paramètres des modèles multitâches.

7.5.6 Impact de l'attention

Comme indiqué à la Section 7.4, nous proposons d'augmenter γ-PhysNet avec des blocs Dual Attention. L'objectif est d'introduire des effets de régularisation et d'adaptation du modèles à différents contextes d'analyse, et également d'introduire une explicabilité des prédictions par l'analyse des cartes d'attention produites pour chaque prédiction. Dans cette section, nous évaluons l'apport effectif de ces mécanismes en comparant les performances de γ-PhysNet et γ-PhysNet DA pour les trois tâches de la reconstruction d'évènements gamma à partir de données Cherenkov : classification gamma/proton, régression de l'énergie et de la direction. Comme nous ne nous comparons pas cette fois-ci à la méthode standard Hillas + RF, nous pouvons relâcher la sélection des données afin d'étendre la gamme d'énergie d'intérêt par rapport aux expériences présentées dans la Section 7.5.5. Plus précisément, les évènements ayant une intensité totale supérieure à 50 photoélectrons sont conservés, contre 300 précédemment. En conservant plus d'évènements de basse énergie classiquement rejetés par l'analyse standard, nous ouvrons la possibilité d'améliorer la sensibilité de l'analyse pour traiter des cas d'usage pertinents tels que l'étude des objets extragalactiques et des sources transitoires. Avec cette sélection, le jeu d'entraînement est maintenant composé de $874k$ gammas et $506k$ protons.

Les mesures de performance moyenne sur la tâche de classification sont rapportées dans la Table 7.2. Nous constatons que l'attention ne permet pas d'améliorer les résultats de la tâche de classification : avec ou sans attention, les performances sont identiques. Cependant, pour les tâches de régression de la direction et de l'énergie, nous observons sur la Figure 7.13 que γ-PhysNet DA obtient des résultats sensiblement meilleurs à haute énergie. Surtout, l'attention permet de réduire significativement la dispersion des résultats. En particulier, pour la tâche de régression de l'énergie la dispersion avec attention est jusqu'à trois fois moindre au-dessus de 200 GeV. En outre, cette dispersion plus faible des résultats de γ-PhysNet DA sur les deux tâches de régression indique une meilleure robustesse à l'initialisation des paramètres du modèle. Ceci permettra d'obtenir une estimation plus fiable des paramètres de la particule détectée lors de l'analyse des données réelles produites par le LST1.

7.5.7 Comprendre l'effet de l'attention sur la robustesse

Afin de comprendre pourquoi l'ajout de blocs Dual Attention à γ-PhysNet n'a pas d'effet sur la tâche de classification, mais améliore la robustesse des tâches de régression de l'énergie et de la direction, nous mettons en oeuvre des méthodes d'explication des prédictions décrites en Section 7.2.4. Tout d'abord, nous analysons soigneusement les cartes

Table 7.2 – Score AUC, précision et rappel de la tâche classification gamma/proton pour γ-PhysNet avec et sans attention.

Modèle	AUC	Précision	Rappel
γ-PhysNet	0.882±0.001	0.929±0.001	0.935±0.007
γ-PhysNet DA	0.882±0.001	0.929±0.001	0.935±0.005

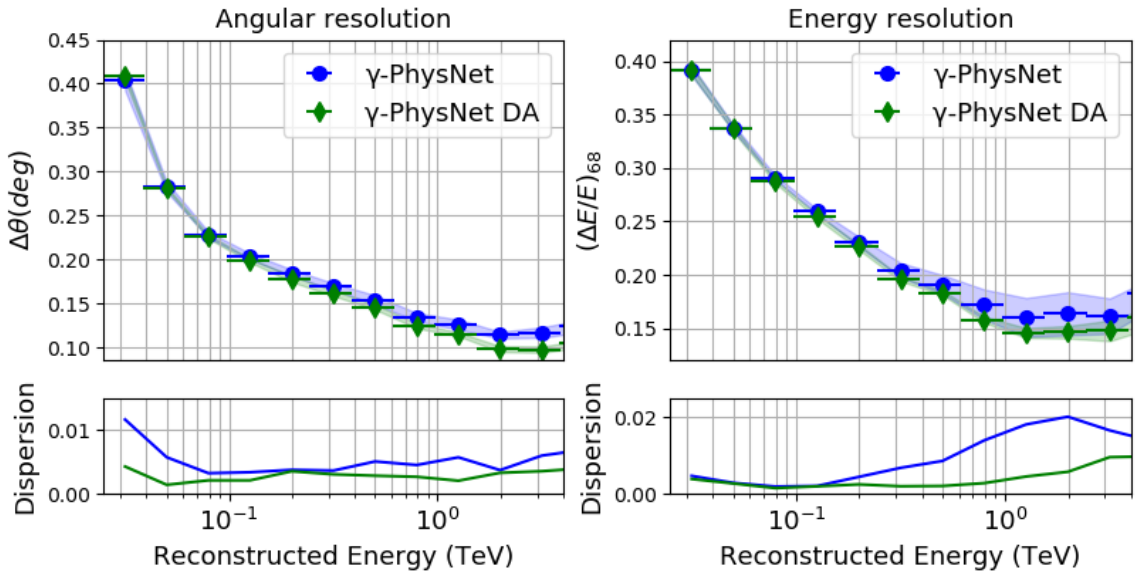


Figure 7.13 – Résolutions angulaire (*à gauche*) et en énergie (*à droite*) en fonction de l'énergie reconstruite dans la bande d'énergie du LST1 (des valeurs plus faibles indiquent un meilleur résultat). Ces courbes représentent l'erreur du modèle pour la régression respectivement de la direction et de l'énergie du rayon gamma détecté. La dispersion, représentant la variabilité induite par l'initialisation des poids, est une mesure de la robustesse du modèle.

de chaleurs Grad-CAM produites par γ-PhysNet et γ-PhysNet DA pour 25 évènements bien et mal reconstruits du jeu de test. Nous observons également les cartes d'attention spatiales des trois blocs Dual Attention. De plus, nous produisons une carte d'attention spatiale globale du modèle en combinant les cartes d'attention à l'aide du produit de Hadamard. Enfin, les modèles étant entraînés avec six graines aléatoires, pour un évènement particulier nous calculons la moyenne des cartes de chaleur Grad-CAM et des cartes d'attention spatiale pour chaque pixel.

7.5.7.1 Observation des cartes de chaleur Grad-CAM

La Figure 7.14 présente un évènement représentatif de la tendance générale observée sur les 25 évènements étudiés. Le premier enseignement concerne la tâche de classification. L'observation des cartes de chaleur avec et sans attention montre que, dans les deux cas, les pixels les plus importants pour la décision du modèle sont situés dans la zone du signal. γ-PhysNet DA (avec attention) prend toutefois en compte une région plus large autour de celui-ci.

Pour la régression de l'énergie, nous observons que sans attention le réseau exploite majoritairement les pixels situés sur les bords de la caméra. D'un point de vue physique, cela peut faire sens, car le fait que le signal soit tronqué ou non est une donnée importante pour la reconstruction de l'énergie. Pour une direction et un point d'impact donnés, la quantité de signal reçue par la caméra du télescope est en effet environ proportionnelle à l'énergie du rayon gamma détecté. Dans le cas de γ-PhysNet DA (avec attention), nous remarquons que le modèle se concentre cette fois sur une large zone autour du signal.

L'analyse des cartes de chaleur de l'altitude et de l'azimut montre un comportement similaire pour γ-PhysNet sans attention. Une fois encore, le fait qu'un signal soit tronqué ou non influe sur la reconstruction de la direction, notamment en altérant l'information sur le développement de la gerbe donnée par le canal des images contenant l'information temporelle. Le modèle avec attention se concentre quant à lui sur la zone du signal, plus particulièrement sur les extrémités de la gerbe.

7.5.7.2 Observation des cartes d'attention spatiale

L'observation des cartes d'attention spatiales de γ-PhysNet DA montre que l'effet des blocs Dual Attention est différent en fonction de l'échelle des cartes de caractéristiques. En sortie de l'étage 1 de l'encodeur, les cartes de caractéristiques sont de même résolution que les données d'entrée du modèle. À cette échelle, le mécanisme d'attention met fortement en valeur les pixels de signal. Les modules Dual Attention produisant des pondérations dans l'intervalle $[1;2]$, la zone de signal à cette échelle reçoit une pondération de 1.8. À ce stade, les cartes d'attention sont toutefois assez bruitées. Après l'étage 2 de l'encodeur, les cartes d'attention spatiale mettent également fortement en valeur la zone de signal dans les données. Elles sont cependant moins bruitées. Ensuite, on observe que le dernier module d'attention, après l'étage 3, a un impact plus faible sur les valeurs des cartes de caractéristiques. Enfin, l'analyse de la combinaison des 3 cartes d'attention met en évidence que l'attention aide fortement le modèle à se concentrer sur les pixels de signal, ce qui est cohérent avec l'observation des cartes de chaleur Grad-CAM.

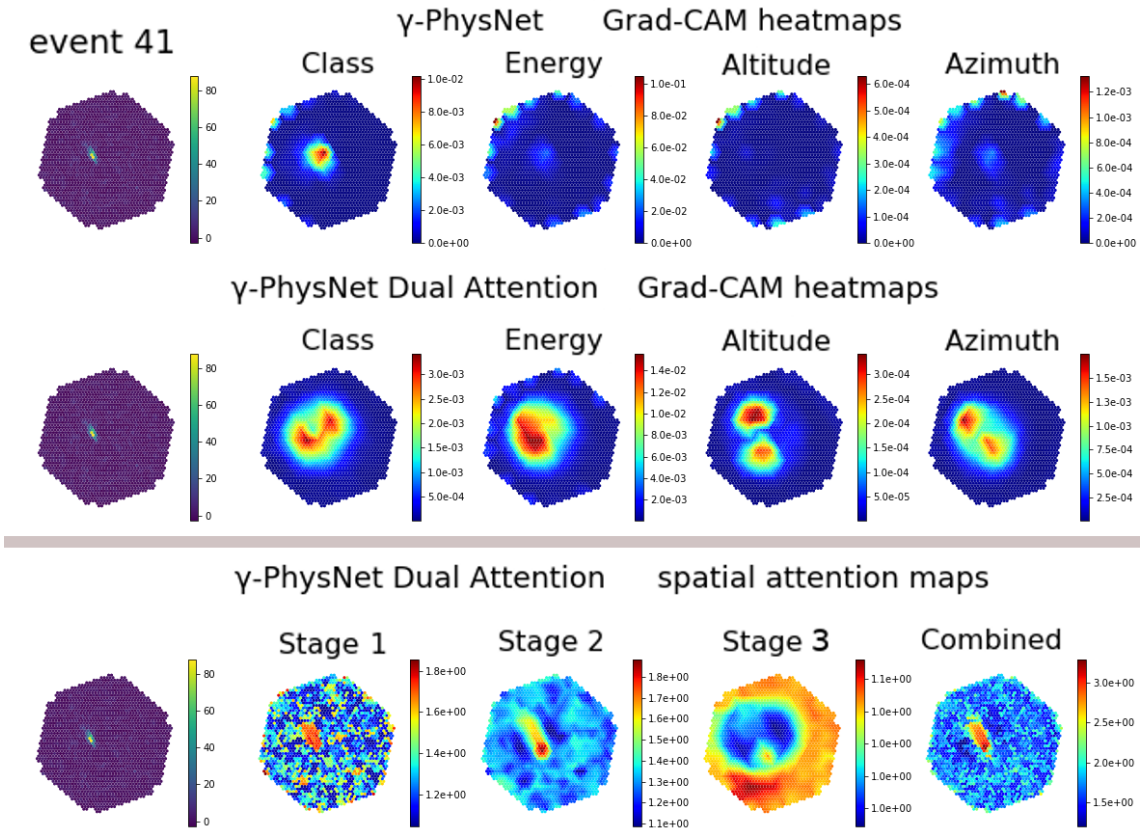


Figure 7.14 – Analyse du comportement de γ-PhysNet avec et sans attention pour un évènement représentatif. La première colonne montre l'évènement analysé (uniquement le canal contenant l'intensité de chaque pixel). La première ligne comporte les cartes de chaleur Grad-CAM produites avec le modèle sans attention pour chacune des tâches (classification gamma/proton, régression de l'énergie et régression de la direction en tant qu'altitude et azimut). La deuxième ligne montre les cartes de chaleur pour le modèle avec attention. La troisième ligne représente les cartes d'attention spatiales situées après chacun des trois étages du modèle avec attention, ainsi que la combinaison de ces trois cartes. Chaque modèle étant entraîné avec six graines aléatoires différentes, les différentes cartes sont moyennées par pixel sur les six versions de chaque modèle.

7.5.7.3 Conclusion

L'utilisation de deux méthodes d'explicabilité visuelle permet de comprendre l'effet du mécanisme Dual Attention sur les performances de γ-PhysNet. Pour la tâche de classification gamma/proton, les versions avec et sans attention du modèle se concentrent toutes deux sur la zone de l'image contenant le signal, ce qui peut expliquer leurs résultats identiques sur cette tâche, tant en matière de performance que de variabilité. En revanche, pour les deux tâches de régression, l'attention permet au modèle de se concentrer sur la zone du signal plutôt que sur les bords de la caméra. Le modèle prend alors mieux en compte les caractéristiques réelles du signal, et peux s'affranchir des limites spatiales du capteur. C'est une piste pour expliquer les performances légèrement meilleures de γ-PhysNet DA sur ces deux tâches, et sa plus grande robustesse à l'initialisation des poids du réseau.

7.6 Analyse préliminaire de données réelles

Durant la rédaction de cette thèse, le LST1 a produit de premières données exploitables en observant quatre sources de rayonnement gamma déjà connues, dont la nébuleuse du Crabe. En astronomie, cette source a été beaucoup étudiée, et est souvent choisie comme chandelle standard. Dans cette thèse, nous réalisons une analyse très préliminaire de deux observations de la nébuleuse du Crabe à l'aide du modèle multitâche γ-PhysNet DA[16] que nous proposons. Cette analyse met en lumière les nombreuses différences existant entre les données de simulation utilisées pour l'entraînement du réseau et les données réelles produites par le télescope. Malgré ces différences, et avec quelques adaptations des données, γ-PhysNet DA[16] obtient une détection statistiquement significative de la source.

7.7 Discussions et perspectives

L'approche profonde multitâche montre clairement son intérêt pour la reconstruction d'évènements gamma à partir des images d'IACT. En particulier, notre architecture, γ-PhysNet, obtient des résultats significativement meilleurs que la méthode standard Hillas + RF pour l'analyse des données de simulation du LST1 servant à préparer les outils d'analyse. L'amélioration de la résolution en énergie devrait permettre de produire des spectres plus détaillés lors de l'analyse des données réelles produites par le télescope. L'amélioration de la résolution angulaire et de la classification gamma/proton apportée par notre approche permettra de détecter des sources plus faibles. La prochaine étape consiste donc à transférer ces performances à l'analyse des données réelles, car bien que les simulations utilisées pour l'entraînement des modèles soient de grande qualité, les données produites par le LST1 en diffèrent. L'analyse préliminaire réalisée sur des données produites par le LST1 montre que ce transfert de performance est un défi important. La production de données de simulation plus proches des données réelles, grâce à une meilleure connaissance de la réponse du télescope, sera un élément clé. En outre, dans l'optique de relever ce défi, nous pourrons également incorporer des données réelles lors de l'entraînement du modèle.

L'ajout de blocs d'attention à notre architecture permet d'améliorer sa robustesse en réduisant la variabilité induite par l'initialisation stochastique des paramètres du réseau.

L'utilisation et la comparaison de deux méthodes d'explicabilité visuelle montrent en effet que le mécanisme Dual Attention aide le modèle à mieux exploiter la zone de l'image contenant le signal. Pour aller plus loin dans l'interprétabilité du modèle, il sera sans doute nécessaire de construire, à partir de ces méthodes visuelles, des indicateurs statistiques permettant une analyse systématique des performances, tant sur les données de simulation que les données réelles.

Ensuite, le temps d'exécution du modèle est un paramètre important pour l'analyse des données de CTA du fait de la quantité de données générées. Pour améliorer l'efficacité computationnelle de notre architecture multitâche, nous réduirons la profondeur de son encodeur. De premières expériences montrent un gain significatif en matière de vitesse d'exécution sans perte de performance. Nous pourrons également tester d'autres architectures plus modernes.

Enfin, cette thèse présente l'analyse des données de CTA à partir d'un seul télescope, le LST1. Or CTA sera un réseau de télescopes. En nous appuyant sur les résultats obtenus avec un seul télescope, nous construirons un modèle d'analyse stéréoscopique des données de CTA.

Bibliography

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

[Abdalla et al., 2019a] Abdalla, H., Adam, R., Aharonian, F., Ait Benkhali, F., Angüner, E., Arakawa, M., Arcaro, C., Armand, C., Ashkar, H., Backes, M., et al. (2019a). Resolving the crab pulsar wind nebula at teraelectronvolt energies. *Nature astronomy*, 3.

[Abdalla et al., 2019b] Abdalla, H., Adam, R., Aharonian, F., Benkhali, F. A., Angüner, E., Arakawa, M., Arcaro, C., Armand, C., Ashkar, H., Backes, M., et al. (2019b). A very-high-energy component deep in the γ-ray burst afterglow. *Nature*, 575(7783):464–467.

[Abdalla et al., 2018] Abdalla, H., Aharonian, F., Benkhali, F. A., Angüner, E., Arakawa, M., Arcaro, C., Armand, C., Arrieta, M., Backes, M., Barnard, M., et al. (2018). Searches for gamma-ray lines and 'pure wimp'spectra from dark matter annihilations in dwarf galaxies with hess. *Journal of Cosmology and Astroparticle Physics*, 2018(11):037.

[Abdallah et al., 2016] Abdallah, H., Abramowski, A., Aharonian, F., Benkhali, F. A., Akhperjanian, A., Angüner, E., Arrieta, M., Aubert, P., Backes, M., Balzer, A., et al. (2016). Search for dark matter annihilations towards the inner galactic halo from 10 years of observations with hess. *Physical Review Letters*, 117(11):111301.

[Acciari et al., 2019] Acciari, V. A. et al. (2019). Teraelectronvolt emission from the γ-ray burst grb 190114c. *Nature*, 575:455.

[Acharyya et al., 2019] Acharyya, A., Agudo, I., Angüner, E., Alfaro, R., Alfaro, J., Alispach, C., Aloisio, R., Batista, R. A., Amans, J.-P., Amati, L., et al. (2019). Monte carlo studies for the optimisation of the cherenkov telescope array layout. *Astroparticle Physics*, 111:35–53.

[Aharonian et al., 2019] Aharonian, F., Ait, B. F., Bernlöhr, K., Bordas, P., Casanova, S., Chakraborty, N., Deil, C., Donath, A., Hahn, J., Hermann, G., et al. (2019). Resolving the crab pulsar wind nebula at teraelectronvolt energies. *Nature Astronomy*.

[Albert et al., 2008a] Albert, J., Aliu, E., Anderhub, H., Antonelli, L., Antoranz, P., Backes, M., Baixeras, C., Barrio, J., Bartko, H., Bastieri, D., et al. (2008a). Very-high-energy gamma rays from a distant quasar: how transparent is the universe? *Science*, 320(5884):1752–1754.

[Albert et al., 2008b] Albert, J., Aliu, E., Anderhub, H., Antoranz, P., Armada, A., Asensio, M., Baixeras, C., Barrio, J., Bartko, H., Bastieri, D., et al. (2008b). Implementation of the random forest method for the imaging atmospheric cherenkov telescope magic. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 588(3):424–432.

[Ambrosi et al., 2013a] Ambrosi, G., Awane, Y., Baba, H., Bamba, A., Barceló, M., de Almeida, U. B., Barrio, J. A., Bigas, O. B., Boix, J., Brunetti, L., Carmona, E., Chabanne, E., Chikawa, M., Colin, P., Conteras, J. L., Cortina, J., Dazzi, F., Deangelis, A., Deleglise, G., Delgado, C., Díaz, C., Dubois, F., Fiasson, A., Fink, D., Fouque, N., Freixas, L., Fruck, C., Gadola, A., García, R., Gascon, D., Geffroy, N., Giglietto, N., Giordano, F., Grañena, F., Gunji, S., Hagiwara, R., Hamer, N., Hanabata, Y., Hassan, T., Hatanaka, K., Haubold, T., Hayashida, M., Hermel, R., Herranz, D., Hirotani, K., Inoue, S., Inoue, Y., Ioka, K., Jablonski, C., Kagaya, M., Katagiri, H., Kishimoto, T., Kodani, K., Kohri, K., Konno, Y., Koyama, S., Kubo, H., Kushida, J., Lamanna, G., Flour, T. L., López-Moya, M., López, R., Lorenz, E., Majumdar, P., Manalaysay, A., Mariotti, M., Martínez, G., Martínez, M., Mazin, D., Miranda, J. M., Mirzoyan, R., Monteiro, I., Moralejo, A., Murase, K., Nagataki, S., Nakajima, D., Nakamori, T., Nishijima, K., Noda, K., Nozato, A., Ohira, Y., Ohishi, M., Ohoka, H., Okumura, A., Orito, R., Panazol, J. L., Paneque, D., Paoletti, R., Paredes, J. M., Pauletta, G., Podkladkin, S., Prast, J., Rando, R., Reimann, O., Ribó, M., Rosier-Lees, S., Saito, K., Saito, T., Saito, Y., Sakaki, N., Sakonaka, R., Sanuy, A., Sasaki, H., Sawada, M., Scalzotto, V., Schultz, S., Schweizer, T., Shibata, T., Shu, S., Sieiro, J., Stamatescu, V., Steiner, S., Straumann, U., Sugawara, R., Tajima, H., Takami, H., Tanaka, S., Tanaka, M., Tejedor, L. A., Terada, Y., Teshima, M., Totani, T., Ueno, H., Umehara, K., Vollhardt, A., Wagner, R., Wetteskind, H., Yamamoto, T., Yamazaki, R., Yoshida, A., Yoshida, T., Yoshikoshi, T., and Consortium, f. t. C. (2013a). The Cherenkov Telescope Array Large Size Telescope. *Proceedings of the 33rd International Cosmic Ray Conference*, pages 8–11.

[Ambrosi et al., 2013b] Ambrosi, G., Awane, Y., Baba, H., et al. (2013b). The cherenkov telescope array large size telescope. *arXiv preprint arXiv:1307.4565*.

[Anderson et al., 2017] Anderson, A., Vasudevan, A., Keane, C., and Gregg, D. (2017). Low-memory gemm-based convolution algorithms for deep neural networks. *arXiv preprint arXiv:1709.03395*.

[Araujo et al., 2019] Araujo, A., Norris, W., and Sim, J. (2019). Computing receptive fields of convolutional neural networks. *Distill*, 4(11):e21.

[Asharindavida et al., 2012] Asharindavida, F., Hundewale, N., and Aljahdali, S. (2012). Study on Hexagonal Grid in Image Processing. *Proc. ICIKM*, 45(Icikm):282–288.

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Barabasz and Gregg, 2019] Barabasz, B. and Gregg, D. (2019). Winograd convolution for dnns: Beyond linear polynomials. In *International Conference of the Italian Association for Artificial Intelligence*, pages 307–320. Springer.

[Barrantes et al., 2018] Barrantes, M., Valdés-Galicia, J., Musalem, O., Hurtado, A., Anzorena, M., García, R., Taylor, R., Muraki, Y., Matsubara, Y., Sako, T., et al. (2018). Atmospheric corrections of the cosmic ray fluxes detected by the solar neutron telescope at the summit of the sierra negra volcano in mexico. *Geofísica internacional*, 57(4):253–275.

[Basu, 2002] Basu, M. (2002). Gaussian-based edge-detection methods-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(3):252–260.

[Baxter, 1997] Baxter, J. (1997). A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39.

[Bengio et al., 2017] Bengio, Y., Goodfellow, I., and Courville, A. (2017). *Deep learning*, volume 1. MIT press Massachusetts, USA:.

[Bergström, 2013] Bergström, L. (2013). Dark matter and imaging air cherenkov arrays. *Astroparticle Physics*, 43:44–49.

[Bernlöhr, 2008] Bernlöhr, K. (2008). Simulation of imaging atmospheric Cherenkov telescopes with CORSIKA and sim_telarray. *Astroparticle Physics*, 30(3):149–158.

[Bernlöhr et al., 2013] Bernlöhr, K., Barnacka, A., Becherini, Y., Bigas, O. B., Carmona, E., Colin, P., Decerprit, G., Di Pierro, F., Dubois, F., Farnier, C., et al. (2013). Monte carlo design studies for the cherenkov telescope array. *Astroparticle Physics*, 43:171–188.

[Bethe and Heitler, 1934] Bethe, H. and Heitler, W. (1934). On the stopping of fast particles and on the creation of positive electrons. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 146(856):83–112.

[Bock et al., 2004] Bock, R., Chilingarian, A., Gaug, M., Hakl, F., Hengstebeck, T., Jiřina, M., Klaschka, J., Kotrč, E., Savickỳ, P., Towers, S., et al. (2004). Methods for multidimensional event classification: a case study using images from a cherenkov gamma-ray telescope. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 516(2-3):511–528.

[Bolmont et al., 2014] Bolmont, J., Corona, P., Gauron, P., et al. (2014). The camera of the fifth h.e.s.s. telescope. part i: System description. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 761:46 – 57.

[Boynton, 2005] Boynton, G. M. (2005). Attention and visual perception. *Current opinion in neurobiology*, 15(4):465–469.

[Brill et al., 2019] Brill, A., Kim, B., Nieto, D., Miener, T., and Feng, Q. (2019). CTLearn: Deep learning for imaging atmospheric Cherenkov telescopes event reconstruction.

[Bronstein et al., 2017] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

[Brunel et al., 2019] Brunel, A., Pasquet, J., PASQUET, J., Rodriguez, N., Comby, F., Fouchez, D., and Chaumont, M. (2019). A cnn adapted to time series for the classification of supernovae. *Electronic Imaging*, 2019(14):90–1.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698.

[Cao et al., 2015] Cao, C., Liu, X., Yang, Y., Yu, Y., Wang, J., Wang, Z., Huang, Y., Wang, L., Huang, C., Xu, W., et al. (2015). Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2956–2964.

[Cao et al., 2018] Cao, J., Li, Y., and Zhang, Z. (2018). Partially shared multi-task convolutional neural network with local constraint for face attribute learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4290–4299.

[Carlini and Wagner, 2017] Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

[Chellapilla et al., 2006] Chellapilla, K., Puri, S., and Simard, P. (2006). High Performance Convolutional Neural Networks for Document Processing. In Lorette, G., editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France). Université de Rennes 1, Suvisoft. http://www.suvisoft.com.

[Chen et al., 2017] Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W., and Chua, T.-S. (2017). Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5659–5667.

[Chen et al., 2018] Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. (2018). GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 794–803. PMLR.

[Cheng et al., 2016] Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

[Chetlur et al., 2014] Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.

[Cho et al., 2013] Cho, D., Lee, M., Kim, S., and Tai, Y.-W. (2013). Modeling the calibration pipeline of the lytro camera for high quality light-field image reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3280–3287.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[Chollet, 2017] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

[Collaboration et al., 2019] Collaboration, H., Abdalla, H., Aharonian, F., Ait Benkhali, F., Angüner, E., Arakawa, M., Arcaro, C., Arm, C., Backes, M., Barnard, M., et al. (2019). Resolving the crab pulsar wind nebula at teraelectronvolt energies. *arXiv preprint arXiv:1909.09494*.

[Cumani et al., 2017] Cumani, P., Eckner, C., Kukec Mezek, G., Stanič, S., Vorobiov, S., Yang, L., Zaharijas, G., Zavrtanik, D., and Zavrtanik, M. (2017). Baseline telescope layouts of the cherenkov telescope array. In *The 35th International Cosmic Ray Conference-ICRC 2017*, volume 301. Univerza v Novi Gorici.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

[Dai et al., 2017] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017). Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773.

[Davis, 1975] Davis, L. S. (1975). A survey of edge detection techniques. *Computer graphics and image processing*, 4(3):248–270.

[De Naurois, 2006] De Naurois, M. (2006). Analysis methods for atmospheric cerenkov telescopes. *arXiv preprint astro-ph/0607247*.

[de Naurois and Rolland, 2009] de Naurois, M. and Rolland, L. (2009). A high performance likelihood reconstruction of γ-rays for imaging atmospheric Cherenkov telescopes. *Astroparticle Physics*, 32(5):231–252.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

[Dukhan, 2019] Dukhan, M. (2019). The indirect convolution algorithm. *arXiv preprint arXiv:1907.02129*.

[Evans et al., 2016] Evans, N., Sanders, J., and Geringer-Sameth, A. (2016). Simple j-factors and d-factors for indirect dark matter detection. *Physical Review D*, 93(10):103512.

[Evoli, 2018] Evoli, C. (2018). The cosmic-ray energy spectrum. https://doi.org/10.5281/zenodo.2360277.

[Eykholt et al., 2018] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634.

[Feng et al., 2016] Feng, Q., Lin, T. T. Y., Collaboration, V., and Others (2016). The analysis of VERITAS muon images using convolutional neural networks. *Proceedings of the International Astronomical Union*, 12(S325):173–179.

[Fiasson et al., 2010] Fiasson, A., Dubois, F., Lamanna, G., Masbou, J., and Rosier-Lees, S. (2010). Optimization of multivariate analysis for IACT stereoscopic systems. *Astroparticle Physics*, 34.

[Fomin et al., 2020] Fomin, V., Anmol, J., Desroziers, S., Kumar, Y., Kriss, J., Tejani, A., and Rippeth, E. (2020). High-level library to help with training neural networks in pytorch. https://github.com/pytorch/ignite.

[Fout et al., 2017] Fout, A., Byrd, J., Shariat, B., and Ben-Hur, A. (2017). Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*, pages 6530–6539.

[Frosst and Hinton, 2017] Frosst, N. and Hinton, G. (2017). Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*.

[Funk, 2015] Funk, S. (2015). Space-and ground-based gamma-ray astrophysics. *arXiv preprint arXiv:1508.05190*.

[Getreuer, 2013] Getreuer, P. (2013). A survey of gaussian convolution algorithms. *Image Processing On Line*, 2013:286–310.

[Glicenstein et al., 2013] Glicenstein, J., Barcelo, M., Barrio, J., et al. (2013). The NectarCAM camera project. *ArXiv e-prints*.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

[Goodfellow et al., 2015] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Guo et al., 2018] Guo, M., Haque, A., Huang, D.-A., Yeung, S., and Fei-Fei, L. (2018). Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 270–287.

[Han et al., 2017] Han, H., Jain, A. K., Wang, F., Shan, S., and Chen, X. (2017). Heterogeneous face attribute estimation: A deep multi-task learning approach. *IEEE transactions on pattern analysis and machine intelligence*, 40(11):2597–2609.

[Han et al., 2016] Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[He et al., 2017] He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (2017)*.

[He et al., 2016a] He, K., Zhang, J., Ren, S., and Sun, J. (2016a). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[He et al., 2016b] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[He and Jia, 2005] He, X. and Jia, W. (2005). Hexagonal structure for intelligent vision. In *Information and Communication Technologies, 2005. ICICT 2005. First International Conference on*, pages 52–64. IEEE.

[Heck et al., 1998] Heck, D., Knapp, J., Capdevielle, J., Schatz, G., Thouw, T., et al. (1998). Corsika: A monte carlo code to simulate extensive air showers. *Report fzka*, 6019(11).

[Henaff et al., 2015] Henaff, M., Bruna, J., and LeCun, Y. (2015). Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

[Hillas, 1985] Hillas, A. (1985). Cerenkov light images of eas produced by primary gamma. In *International Cosmic Ray Conference*, volume 3.

[Hillas et al., 1998] Hillas, A., Akerlof, C., Biller, S., Buckley, J., Carter-Lewis, D., Catanese, M., Cawley, M., Fegan, D., Finley, J., Gaidos, J., et al. (1998). The spectrum of tev gamma rays from the crab nebula. *The Astrophysical Journal*, 503(2):744.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[Hinton and Hofmann, 2009] Hinton, J. and Hofmann, W. (2009). Teraelectronvolt astronomy. *Annual Review of Astronomy and Astrophysics*, 47:523–565.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hofmann, 2017] Hofmann, W. (2017). Perspectives from cta in relativistic astrophysics. *International Journal of Modern Physics D*, 26(03):1730005.

[Holch et al., 2017] Holch, T. L., Shilon, I., Büchele, M., Fischer, T., Funk, S., Groeger, N., Jankowsky, D., Lohse, T., Schwanke, U., and Wagner, P. (2017). Probing convolutional neural networks for event reconstruction in gamma-ray astronomy with cherenkov telescopes. *arXiv preprint arXiv:1711.06298*.

[Holch et al., 2017] Holch, T. L., Shilon, I., Büchele, M., Fischer, T., Funk, S., Groeger, N., Jankowsky, D., Lohse, T., Schwanke, U., and Wagner, P. (2017). Probing Convolutional Neural Networks for Event Reconstruction in Gamma-Ray Astronomy with Cherenkov Telescopes. In *35th International Cosmic Ray Conference (ICRC2017)*, volume 301 of *International Cosmic Ray Conference*, page 795.

[Holschneider et al., 1990] Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. (1990). A real-time algorithm for signal analysis with the help of the wavelet transform. In Combes, J.-M., Grossmann, A., and Tchamitchian, P., editors, *Wavelets*, pages 286–297, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Hoogeboom et al., 2018] Hoogeboom, E., Peters, J. W., Cohen, T. S., and Welling, M. (2018). Hexaconv. In *International Conference on Learning Representations*.

[Hoppe et al., 2009] Hoppe, S. et al. (2009). Detection of very-high-energy gamma-ray emission from the vicinity of PSR B1706-44 with H.E.S.S. *arXiv e-prints*, page arXiv:0906.5574.

[Howard et al., 2019] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324.

[Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[Hu et al., 2018] Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

[Huang et al., 2017] Huang, G., Liu, Z., v. d. Maaten, L., and Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

[Huennefeld, 2017] Huennefeld, M. (2017). Deep learning in physics exemplified by the reconstruction of muon-neutrino events in icecube. *PoS*, page 1057.

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

[Iizuka et al., 2016] Iizuka, S., Simo-Serra, E., and Ishikawa, H. (2016). Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (ToG)*, 35(4):1–11.

[Inoue et al., 2013] Inoue, S., Granot, J., O'Brien, P. T., Asano, K., Bouvier, A., Carosi, A., Connaughton, V., Garczarczyk, M., Gilmore, R., Hinton, J., et al. (2013). Gamma-ray burst science in the era of the cherenkov telescope array. *Astroparticle Physics*, 43:252–275.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

[Jacquemont et al., 2019a] Jacquemont, M., Antiga, L., Vuillaume, T., Silvestri, G., Benoit, A., Lambert, P., and Maurin, G. (2019a). Indexed operations for non-rectangular lattices applied to convolutional neural networks. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,*, pages 362–371. INSTICC, SciTePress.

[Jacquemont and Vuillaume, 2019] Jacquemont, M. and Vuillaume, T. (2019). Indexed-conv library. https://doi.org/10.5281/zenodo.2542664.

[Jacquemont et al., 2019b] Jacquemont, M., Vuillaume, T., Benoit, A., Lambert, P., Maurin, G., Lamanna, G., and Brill, A. (2019b). Gammalearn: a deep learning framework for iact data. In *ICRC 2019 - 36th International Cosmic Ray Conference*.

[Jordà et al., 2019] Jordà, M., Valero-Lara, P., and Peña, A. J. (2019). Performance evaluation of cudnn convolution algorithms on nvidia volta gpus. *IEEE Access*, 7:70461–70473.

[Katz, 2012] Katz, U. (2012). A neutrino telescope deep in the Mediterranean Sea. *CERN Cour.*, 52N6:31–33.

[Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

[Kendall et al., 2018] Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.

[Keskar and Socher, 2017] Keskar, N. S. and Socher, R. (2017). Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*.

[Kim et al., 2019] Kim, B., Brill, A., Miener, T., Nieto, D., and Feng, Q. (2019). DL1-Data-Handler: DL1 HDF5 writer, reader, and processor for IACT data. https://doi.org/10.5281/zenodo.3336561. v0.8.1-legacy.

[Kim and Brunner, 2016] Kim, E. J. and Brunner, R. J. (2016). Star-galaxy classification using deep convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, page stw2672.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Kokkinos, 2017] Kokkinos, I. (2017). Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138.

[Kortylewski et al., 2019] Kortylewski, A., Egger, B., Schneider, A., Gerig, T., Morel-Forster, A., and Vetter, T. (2019). Analyzing and reducing the damage of dataset bias to face recognition with synthetic data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Lavin and Gray, 2016] Lavin, A. and Gray, S. (2016). Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4013–4021.

[Le and Borji, 2017] Le, H. and Borji, A. (2017). What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks? *arXiv preprint arXiv:1705.07049*.

[LeCun et al., 1989a] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989a). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[LeCun et al., 1989b] LeCun, Y. et al. (1989b). Generalization and network design strategies. *Connectionism in perspective*, 19:143–155.

[Li and Ma, 1983] Li, T.-P. and Ma, Y.-Q. (1983). Analysis methods for results in gamma-ray astronomy. *The Astrophysical Journal*, 272:317–324.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

[Lopez-Coto et al. for CTA LST project, 2021] Lopez-Coto et al. for CTA LST project, R. (2021). lstchain: An analysis pipeline for LST-1, the first prototype Large-Sized Telescope of CTA. In *Astronomical Data Analysis Software and Systems XXX*, Astronomical Society of the Pacific Conference Series.

[Luo et al., 2017] Luo, J.-H., Wu, J., and Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066.

[Luo et al., 2019] Luo, L., Xiong, Y., Liu, Y., and Sun, X. (2019). Adaptive gradient methods with dynamic bound of learning rate. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[Luo et al., 2016] Luo, W., Li, Y., Urtasun, R., and Zemel, R. (2016). Understanding the effective receptive field in deep convolutional neural networks. In *Advances in neural information processing systems*, pages 4898–4906.

[Luvizon et al., 2018] Luvizon, D. C., Picard, D., and Tabia, H. (2018). 2d/3d pose estimation and action recognition using multitask deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[Mangano et al., 2018] Mangano, S., Delgado, C., Bernardos, M. I., Lallena, M., Vázquez, J. J. R., Consortium, C., et al. (2018). Extracting gamma-ray information from images with convolutional neural network methods on simulated cherenkov telescope array data. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 243–254. Springer.

[Mewaldt, 1994] Mewaldt, R. (1994). Galactic cosmic ray composition and energy spectra. *Advances in Space Research*, 14(10):737–747.

[Middleton and Sivaswamy, 2001] Middleton, L. and Sivaswamy, J. (2001). Edge detection in a hexagonal-image processing framework. *Image and Vision computing*, 19(14):1071–1081.

[Monti et al., 2017] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124.

[Morcos et al., 2018a] Morcos, A. S., Barrett, D. G., Rabinowitz, N. C., and Botvinick, M. (2018a). On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*.

[Morcos et al., 2018b] Morcos, A. S., Barrett, D. G. T., Rabinowitz, N. C., and Botvinick, M. (2018b). On the importance of single directions for generalization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814.

[Neeser et al., 2000] Neeser, W., Bocker, M., Buchholz, P., et al. (2000). The depfet pixel bioscope. *IEEE Transactions on Nuclear Science*, 47(3):1246–1250.

[Nieto et al., 2019a] Nieto, D., Brill, A., Feng, Q., Humensky, T., Kim, B., Miener, T., Mukherjee, R., and Sevilla, J. (2019a). Ctlearn: Deep learning for gamma-ray astronomy. *arXiv preprint arXiv:1912.09877*.

[Nieto et al., 2019b] Nieto, D., Brill, A., Feng, Q., Jacquemont, M., Kim, B., Miener, T., and Vuillaume, T. (2019b). Studying deep convolutional neural networks with hexagonal lattices for imaging atmospheric cherenkov telescope event reconstruction. In *ICRC 2019 - 36th International Cosmic Ray Conference*.

[Nieto et al., 2017] Nieto, D., Brill, A., Kim, B., Humensky, T. B., Others, Consortium, C. T. A., and Others (2017). Exploring deep learning as an event classification method for the Cherenkov Telescope Array. *arXiv preprint arXiv:1709.05889*, pages 1–8.

[Nigro et al., 2019] Nigro, C., Deil, C., Zanin, R., Hassan, T., King, J., Ruiz, J. E., Saha, L., Terrier, R., Brügge, K., Nöthe, M., et al. (2019). Towards open and reproducible multi-instrument analysis in gamma-ray astronomy. *Astronomy & Astrophysics*, 625:A10.

[NVIDIA, 2014] NVIDIA (2014). Nvidia deep learning sdk, cudnn developer guide. https://docs.nvidia.com/deeplearning/sdk/cudnn-api/index.html#cudnnConvolutionFwdAlgo_t. Accessed: 2020-04-28.

[Ohm et al., 2009] Ohm, S., van Eldik, C., and Egberts, K. (2009). γ/hadron separation in very-high-energy γ-ray astronomy using a multivariate analysis method. *Astroparticle Physics*, 31(5):383–391.

[Olah, 2015] Olah, C. (2015). Calculus on computational graphs: Backpropagation. http://colah.github.io/posts/2015-08-Backprop/. Accessed: 2020-07-05.

[Olah et al., 2017] Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill*, 2(11):e7.

[Papakyriakopoulos et al., 2020] Papakyriakopoulos, O., Hegelich, S., Serrano, J. C. M., and Marco, F. (2020). Bias in word embeddings. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 446–457.

[Parmar et al., 2019] Parmar, N., Ramachandran, P., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. (2019). Stand-alone self-attention in vision models. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 68–80. Curran Associates, Inc.

[Parmar et al., 2018] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. (2018). Image transformer. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064. PMLR.

[Parsons et al., 2016] Parsons, R., Murach, T., and Gajdus, M. (2016). Hess ii data analysis with impact. In *The 34th International Cosmic Ray Conference*, volume 236, page 826. SISSA Medialab.

[Parsons and Ohm, 2019] Parsons, R. and Ohm, S. (2019). Background rejection in atmospheric cherenkov telescopes using recurrent convolutional neural networks. *arXiv preprint arXiv:1910.09435*.

[Parsons and Hinton, 2014] Parsons, R. D. and Hinton, J. A. (2014). A Monte Carlo template based analysis for air-Cherenkov arrays. *Astroparticle Physics*, 56:26–34.

[Parsons and Ohm, 2020] Parsons, R. D. and Ohm, S. (2020). Background rejection in atmospheric Cherenkov telescopes using recurrent convolutional neural networks. *European Physical Journal C*, 80(5):363.

[Pasquet et al., 2019] Pasquet, J., Pasquet, J., Chaumont, M., and Fouchez, D. (2019). Pelican: deep architecture for the light curve analysis. *Astronomy & Astrophysics*, 627:A21.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

[Pavllo et al., 2019] Pavllo, D., Feichtenhofer, C., Grangier, D., and Auli, M. (2019). 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7753–7762.

[Paz Arribas, 2017] Paz Arribas, M. (2017). Estimation of trigger rates, data rates and data volumes for cta and observations of snr rx j0852. 0- 4622 with hess.

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

[Pühlhofer et al., 2012] Pühlhofer, G., Bauer, C., Biland, A., et al. (2012). FlashCam: A fully digital camera for CTA telescopes. In Aharonian, F. A., Hofmann, W., and Rieger, F. M., editors, *American Institute of Physics Conference Series*, volume 1505 of *American Institute of Physics Conference Series*, pages 777–780.

[Ragan-Kelley et al., 2014] Ragan-Kelley, M., Perez, F., Granger, B., Kluyver, T., Ivanov, P., Frederic, J., and Bussonnier, M. (2014). The jupyter/ipython architecture: a unified view of computational research, from interactive exploration to communication and publication. In *AGU Fall Meeting Abstracts*.

[Raina et al., 2009] Raina, R., Madhavan, A., and Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880.

[Reddi et al., 2018] Reddi, S. J., Kale, S., and Kumar, S. (2018). On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[Ren and Jae Lee, 2018] Ren, Z. and Jae Lee, Y. (2018). Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 762–771.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

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

[Ruder, 2017] Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Rummelt, 2010] Rummelt, N. I. (2010). *Array Set Addressing: Enabling Efficient Hexagonally Sampled Image Processing*. PhD thesis, University of Florida.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[Saenz et al., 2002] Saenz, M., Buracas, G. T., and Boynton, G. M. (2002). Global effects of feature-based attention in human visual cortex. *Nature neuroscience*, 5(7):631–632.

[Sandler et al., 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.

[Sato et al., 2002] Sato, H., Matsuoka, H., Onozawa, A., and Kitazawa, H. (2002). Hexagonal image representation for 3-d photorealistic reconstruction. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 672–676. IEEE.

[Scherer et al., 2010] Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010*, pages 92–101. Springer.

[Schwarz et al., 1999] Schwarz, M., Hauschild, R., Hosticka, B. J., et al. (1999). Single-chip cmos image sensors for a retina implant system. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(7):870–877.

[Scovell, 2013] Scovell, P. (2013). The Xenon 100 Detector. In Cline, D., editor, *Springer Proceedings in Physics*, volume 148 of *Springer Proceedings in Physics*, page 87.

[Selvaraju et al., 2017] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

[Sener and Koltun, 2018] Sener, O. and Koltun, V. (2018). Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*.

[Shafahi et al., 2019] Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. (2019). Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3358–3369.

[Shen and Sethi, 1996] Shen, B. and Sethi, I. K. (1996). Convolution-based edge detection for image/video in block dct domain. *Journal of Visual Communication and Image Representation*, 7(4):411–423.

[Shilon et al., 2019] Shilon, I., Kraus, M., Büchele, M., Egberts, K., Fischer, T., Holch, T. L., Lohse, T., Schwanke, U., Steppa, C., and Funk, S. (2019). Application of deep learning methods to analysis of imaging atmospheric cherenkov telescopes data. *Astroparticle Physics*, 105:44–53.

[Shima et al., 2010] Shima, T., Sugimoto, S., and Okutomi, M. (2010). Comparison of image alignment on hexagonal and square lattices. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 141–144. IEEE.

[Sifre and Mallat, 2014] Sifre, L. and Mallat, S. (2014). Rigid-motion scattering for image classification. *Ph. D. thesis*.

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Sousa, 2014] Sousa, N. A. (2014). Hexagonal grid image processing algorithms in neural vision and prediction. *Faculdade de Engenharia da Universidade*.

[Springenberg et al., 2015] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. (2015). Striving for simplicity: The all convolutional net. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

[Srinivas and Fleuret, 2019] Srinivas, S. and Fleuret, F. (2019). Full-gradient representation for neural network visualization. In *Advances in Neural Information Processing Systems*, pages 4126–4135.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[Sun et al., 2020] Sun, J., Darbeha, F., Zaidi, M., and Wang, B. (2020). Saunet: Shape attentive u-net for interpretable medical image segmentation. *arXiv preprint arXiv:2001.07645*.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., Hill, C., and Arbor, A. (2015). Going Deeper with Convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[Tan and Le, 2019] Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

[Thrun, 1996] Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pages 640–646.

[Touvron et al., 2019] Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. (2019). Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems*, pages 8252–8262.

[Touvron et al., 2020] Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. (2020). Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*.

[Trottier et al., 2017] Trottier, L., Giguère, P., and Chaib-draa, B. (2017). Multi-task learning by deep collaboration and application in facial landmark detection. *arXiv preprint arXiv:1711.00111*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Verma et al., 2018] Verma, N., Boyer, E., and Verbeek, J. (2018). Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2598–2606.

[Völk and Bernlöhr, 2009] Völk, H. J. and Bernlöhr, K. (2009). Imaging very high energy gamma-ray telescopes. *Experimental Astronomy*, 25(1-3).

[Vuillaume et al., 2018] Vuillaume, T., Jacquemont, M., Antiga, L., Benoit, A., Lambert, P., Maurin, G., Silvestri, G., and the CTA consortium (2018). Gammalearn - first steps to apply deep learning to the cherenkov telescope array data. In *CHEP 2018 - 23rd International Conference on Computing in High Energy and Nuclear Physics*.

[Wang et al., 2020] Wang, C.-Y., Mark Liao, H.-Y., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., and Yeh, I.-H. (2020). Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 390–391.

[Wang et al., 2018] Wang, X., Girshick, R., Gupta, A., and He, K. (2018). Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803.

[Weekes et al., 1989] Weekes, T. C., Cawley, M. F., Fegan, D., Gibbs, K., Hillas, A., Kowk, P., Lamb, R., Lewis, D., Macomb, D., Porter, N., et al. (1989). Observation of tev gamma-rays from the crab nebula using the atmospheric cherenkov imaging technique. *Astrophysical Journal*, 342:379–395.

[Winograd, 1980] Winograd, S. (1980). *Arithmetic complexity of computations*, volume 33. Siam.

[Wolf et al., 2017] Wolf, M. J., Miller, K. W., and Grodzinsky, F. S. (2017). Why we should have seen that coming: comments on microsoft's tay "experiment," and wider implications. *The ORBIT Journal*, 1(2):1–12.

[Woo et al., 2018] Woo, S., Park, J., Lee, J.-Y., and So Kweon, I. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

[Xia et al., 2016] Xia, G., Hu, J., Hu, F., et al. (2016). AID: A benchmark dataset for performance evaluation of aerial scene classification. *CoRR*, abs/1608.05167.

[Xu et al., 2018] Xu, D., Ouyang, W., Wang, X., and Sebe, N. (2018). Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 675–684.

[Yanardag and Vishwanathan, 2015] Yanardag, P. and Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374.

[Yang and Hospedales, 2017] Yang, Y. and Hospedales, T. M. (2017). Trace norm regularised deep multi-task learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

[Yuan et al., 2019] Yuan, Y., Chen, X., and Wang, J. (2019). Object-contextual representations for semantic segmentation. *arXiv preprint arXiv:1909.11065*.

[Zhang et al., 2019] Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2019). Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363.

[Zhang et al., 2020] Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., Sun, Y., He, T., Mueller, J., Manmatha, R., et al. (2020). Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*.

[Zhao et al., 2019] Zhao, R., Hu, Y., Dotzel, J., De Sa, C., and Zhang, Z. (2019). Improving neural network quantization without retraining using outlier channel splitting. In *International Conference on Machine Learning*, pages 7543–7552.

[Zhou et al., 2018] Zhou, B., Sun, Y., Bau, D., and Torralba, A. (2018). Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*.

A

Gammalearn: a Framework to Ease Deep Learning Process with IACT Data

Deep Learning is a highly empirical process requiring many training and testing cycles of different architectures with different hyperparameters. Moreover, the goal of the GammaLearn project is to find the best possible neural networks for gamma / cosmic rays separation and gamma parameters reconstruction, stressing the need of a tool to ensure reproducibility, traceability and easy launch for all the experiments to be run. The GammaLearn framework has been designed and developed to tackle these issues. We published the work relative to this framework as contributions to the CHEP [Vuillaume et al., 2018] and ICRC [Jacquemont et al., 2019b] conferences.

A.1 Framework Description

It is a modular and plug and play Python first tool that relies on PyTorch [Paszke et al., 2017] for DL capabilities, mainly tensor manipulation, automatic differentiation (which is essential for gradient descent optimization) and GPU computations. In GammaLearn the training process itself (i.e. the execution of the training, validating and testing loops) is handled by Ignite (v1.0a) [Fomin et al., 2020] in order to benefit from its event management system. In the following, an *experiment* designates the whole process of training and testing a CNN with particular hyperparameters.
As described on Fig. A.1, GammaLearn is composed of an engine, the experiment_runner, and 7 collections of tool functions and classes. The experiment_runner role is to load and check the experiment settings, via the Experiment class, load the training data, load the CNN, train, validate and test the loaded CNN and produce monitoring data and performance metrics, as defined in the experiment settings file. The tool collections provide all the functions and classes to:

- load datasets,
- pre-process data (filter, augment, transform),

- train, validate and test networks,
- monitor the training process,
- visualize training results.

Each collection of tools, serving a specific purpose, follows the same prototype for function and class definition. For example, the Handlers collection contains functions to handle the events fired by Ignite's engine, like training or validation loss logging when an epoch is completed. To add a new handler or to build one's own collection of handler compatible with GammaLearn , one needs to observe the following prototype:

```
def create_new_handler(experiment):
    """
    Function to create a handler
    Parameters
    ----------
    experiment (Experiment): the experiment
    Returns
    -------
    A function registrable by ignite Trainer
    """
    # do something
    def handle_an_event(trainer):
    # do something
    return handle_an_event
```

The collections allow the user to run various types of experiment on IACT data: classification, regression, single task learning, multi-task learning, mono or stereo analysis.
Beside the different collections, the plug and play quality of GammaLearn also results from its Python first nature. Each component of the framework is written in Python, even the experiment setting file, allowing the user to add his own components by calling them in the experiment settings file.

A.2 Ecosystem

To be an efficient DL framework for IACT data, GammaLearn is integrated in a wide ecosystem of tools:

- IndexedConv [Jacquemont et al., 2019a] : the IndexedConv package provides convolution and pooling operations for images with any kind of grid. Indeed, in the case of hexagonal grid images, standard 2D convolution functions implemented DL frameworks are not suitable because they assume the grid of the image they process to be Cartesian. IndexedConv relies on the list of neighbours of each pixel of interest to compute the convolution, and thus can be applied to any image grid and shape. Moreover, it comes with all the needed functions for hexagonal grid images in particular and CTA images in general (i.e. building the necessary index matrix, extracting the list of neighbours from it). IndexedConv is fully supported by GammaLearn .
- GammaBoard: GammaBoard is an interactive dashboard build to display metrics assessing the reconstructions performances of Imaging Atmospheric Cherenkov

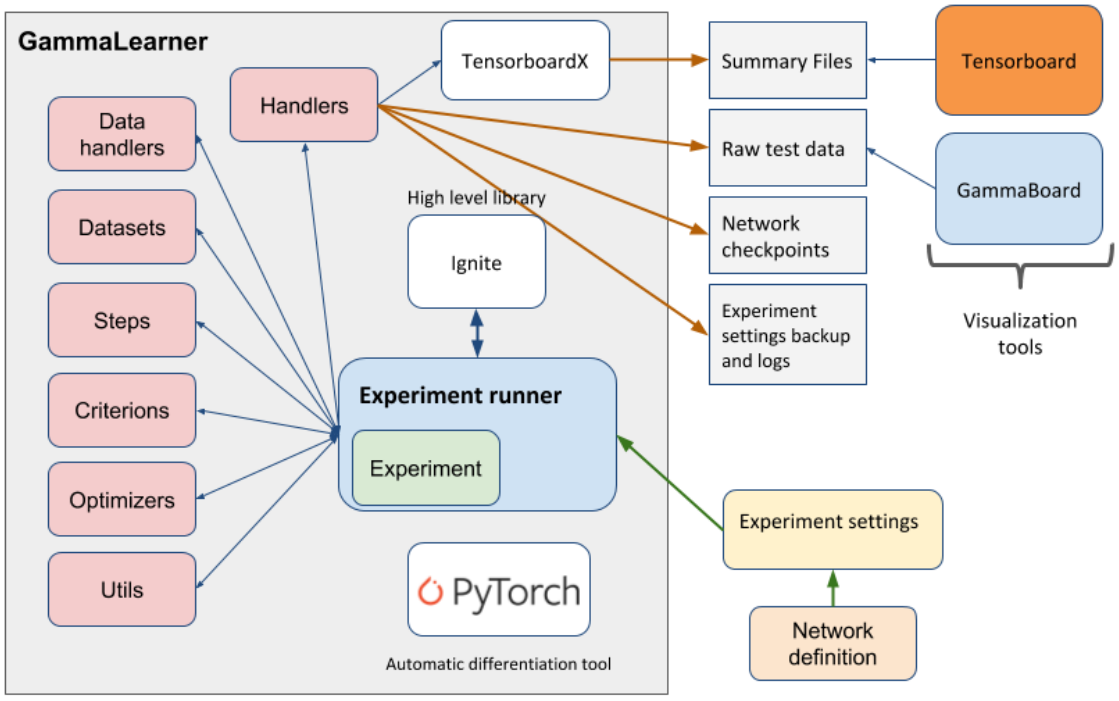


Figure A.1 – Description of the GammaLearn framework. GammaLearn comes with a set of function and class collections (in pink) to process IACT data. It relies on PyTorch for the Deep Learning fundamentals and on Ignite for the training routines. GammaLearn generates monitoring and performances data during the training, directly workable by Tensorboard and GammaBoard.

Telescopes (IACTs). It is built upon the widespread Jupyter Notebook [Ragan-Kelley et al., 2014] technology. It benefits from Matplotlib [Hunter, 2007] interactive plots. Firstly developed as a stand alone application, it is now part of the ctaplot[1] package that provides analysis and plot functions for IACT related metrics (resolutions curves and effective area among others). Thanks to its click-and-play interface, as shown on Fig. A.2, it allows a quick and simple comparison of the experiments.

- Tensorboard

  Tensorboard[2] is a suite of web applications coming with Tensorflow [Abadi et al., 2016]. It offers useful tools to visualize monitoring data (e.g. network weights distribution over the training or GPU memory used), training performances (e.g. loss and accuracy evolution) and to inspect neural networks. Thanks to tensorboardX[3], a module to export data in a format readable by Tensorboard, and the Handlers collection, GammaLearn benefits from the power of Tensorboard.

- DL1 Data Handler

  DL1 data handler (DL1DH)[4] is a Python library to handle calibrated images from CTA. The package has been developed to handle CTA raw data and write, read

[1] https://github.com/cta-observatory/ctaplot
[2] https://github.com/tensorflow/tensorboard
[3] https://github.com/lanpa/tensorboardX
[4] https://github.com/cta-observatory/dl1-data-handler

and apply image processing to calibrated images. DL1DH has been integrated to GammaLearn to ease the process of loading CTA data and provide them seamlessly to the framework. As IACTs image are often non-standard images (e.g. presenting hexagonal pixels in hexagonal lattices), image pre-processing (e.g. oversampling, rebinning or interpolation) can be applied thanks to DL1DH. A study of the effect of these pre-processing has been realised by [Nieto et al., 2019b].

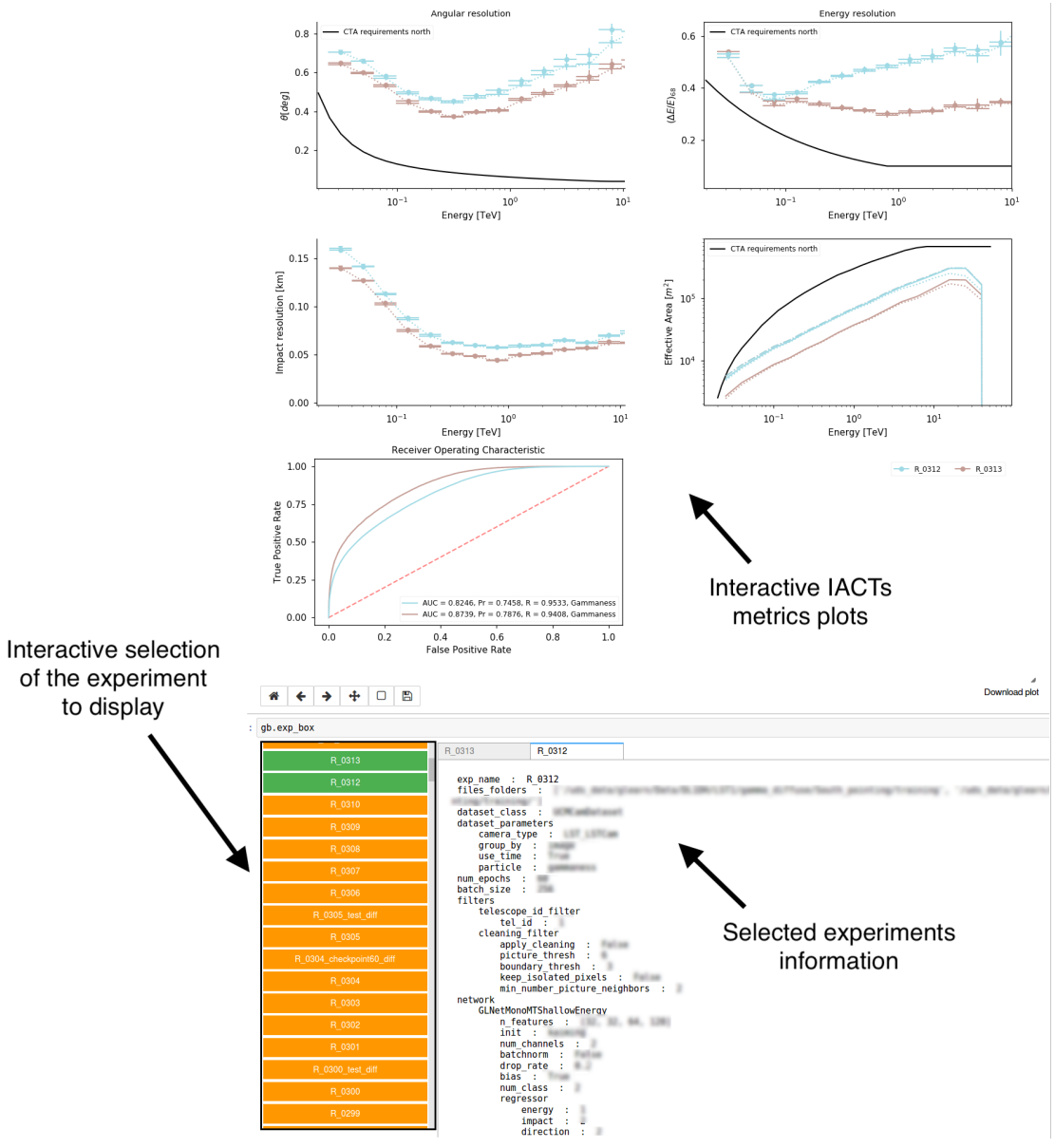


Figure A.2 – Gammaboard interface. It allows the user to interactively display experiment results with metrics specific to IACTs, providing a quick and meaningful comparison.

To fully benefit from this ecosystem, it is highly recommend to use Conda[5] environment manager.

[5] https://conda.io/en/latest/

A.3 Work-flow

As shown on Fig. A.1, the work-flow to train a network on IACT data with GammaLearn is pretty straightforward. For a typical experiment, i.e. taking advantage of the already implemented class and functions in the GammaLearn 's collections, one needs to provide the framework with an experiment settings file (in Python) and the desired network definition (with PyTorch). An example of experiment settings file can be found in the GammaLearn repository, in the folder *examples*, comprising all the mandatory and the optional setting fields handled by the framework. Then, to start the experiment, one executes the following commands in a bash terminal:

```
source activate <GammaLearn conda environment>
cd <path to GammaLearn>/gammalearner
python experiment_runner.py <experiment_settings_path> [--logdir] [--
    debug]
```

The framework trains the network and produces the monitoring and performance data as defined in the experiment settings file.

B

Summary of Experiment Hyperparameters

In this appendix, I summarize the hyperparameters of all the experiments presented in this thesis for reproducibility.

B.1 Chapter 3: Indexed Convolution

B.1.1 Experiment on CIFAR-10

Table B.1 – Training hyperparameters for the experiment on CIFAR-10.

optimizer	weight decay	learning rate	learning rate decay	epochs	batch size
SGD	10^{-3}	0.05	0.1	300	125
momentum 0.9	L2		epochs [50, 100, 150]		

B.1.2 Experiment on AID

Table B.2 – Training hyperparameters for the experiment on AID.

optimizer	weight decay	learning rate	learning rate decay	epochs	batch size
SGD	10^{-3}	0.05	0.1	300	100
momentum 0.9	L2		epochs [50, 100, 150]		

B.1.3 Experiment on CTA Data (Prod3b)

Table B.3 – Model hyperparameters for the experiment on CTA data.

		conv 1	conv 2	conv 3	conv 4
common parameters	features	32	32	64	128
	activation	ReLU	ReLU	ReLU	ReLU
mapping methods	kernel	3×3	3×3	3×3	3×3
	pooling	max	max	max	max
Indexed Convolution	kernel	7	7	7	7
	pooling	no	max	max	max

Table B.4 – Training hyperparameters for the experiment on CTA data.

optimizer	learning rate	iterations	batch size
Adam	$5 \cdot 10^{-5}$	50000	64

B.1.4 Extended Experiment on CTA Data (LST4 Mono-Trigger)

Table B.5 – Training hyperparameters for the experiment extended on CTA data.

optimizer	weight decay	learning rate	learning rate decay	epochs	batch size
Adam	10^{-4}	10^{-3}	0.1	25	128
	L2		every 10 epochs		

Table B.6 – Loss functions for the experiment extended on CTA data.

task	gamma/proton separation	energy	direction
loss function	cross-entropy	$L1$	$L1$

B.2 Chapter 4: γ-PhysNet

B.2.1 Multi-task Learning Performance and Attention Study

Table B.7 – Training hyperparameters for the experiment on Multi-Task Learning.

optimizer	weight decay	learning rate	learning rate decay	epochs	batch size
Adam	10^{-4}	10^{-3}	0.1	25	128
	L2		every 10 epochs		

Table B.8 – Loss balancing hyperparameters for the experiment on Multi-Task Learning.

method	optimizer	learning rate	weight decay
uncertainty estimation	Adam	0.025	10^{-4}

Table B.9 – Loss functions for the experiment on Multi-Task Learning.

task	gamma/proton separation	energy	direction	impact point
loss function	cross-entropy	$L1$	$L1$	$L1$

C

Attention Reduction Ratio Ablation Study

This appendix presents the two-stage ablation study I have realized to select the reduction ratios of the attention mechanisms compared in Section 4.2.4 and detailed in Section 2.3. Each of these attention methods has a bottleneck that compresses the latent representation. In the Squeeze-and-Excitation and dual attention modules, this bottleneck is the main component of the recalibration process that yields the channel-wise attention weights. In the self-attention module, the bottleneck reduces the dimensions of the feature maps before computing the attention map in order to lower the computational cost. In addition, increasing the compression of the latent feature maps (in a trade-off with expressivity) can help the network generalize as it forces to learn a representation of smaller dimension for the data. The compression of these bottlenecks is controlled by the reduction ratio.

To evaluate the performance of different reduction ratios, I train γ-PhysNet augmented with each of the attention mechanisms (see Section 4.1.3 for details) to address gamma / proton separation, energy and direction reconstruction tasks. As the high cuts are very selective, they strongly reduce the size of the training set and thus the experiment duration. The whole training phase of γ-PhysNet with the high cuts lasts roughly 4 hours on an NVIDIA V100 GPU, while it lasts about 22 hours with the mid cuts. So, for each of the three attention method probed (Squeeze-and-Excitation, self-attention and dual attention), I first carry out a more extensive experiment with the high cuts and then, relying on its results, a lighter experiment with the mid cuts. As the duration of an experiment with the low cuts is roughly 38 hours on an NVIDIA V100 GPU, I select for the low cuts the same reduction ratio as for the mid cuts.

C.1 High Cuts

Squeeze-and-Excitation The default reduction ratio for the Squeeze-and-Excitation method is 16. This is rather strong for γ-PhysNet as it corresponds to the number of neurons per layer in the first residual block, *i.e.*, in the first attention module the number of channels is compressed to one during the recalibration phase. For the same reason, we cannot probe higher ratios. To study the effect of relaxing the compression of the

bottleneck present in the attention module, I probe the following ratios: 2, 4, 8, 12 and 16.

These experiments show similar performance for all the ratios on the gamma / proton separation, direction and energy reconstruction tasks, the differences between the configurations being smaller than the initialization variability observed in Section 4.2.4. The Squeeze-and-Excitation mechanism seems to be marginally sensitive to the compression of its bottleneck.

However, the sensitivity curve of γ-PhysNet SE[2] (*i.e.*, Squeeze-and-Excitation module with a ratio of 2) is slightly better below 800 GeV.

Self-Attention. The default reduction ratio for the self-attention mechanism is 8. As for the Squeeze-and-Excitation method, we can increase it up to 16, resulting in a twice stronger compression. To evaluate the effect of relaxing and increasing the compression, I probe the ratios 4, 8, 12 and 16.

All the ratios obtain similar results on gamma / proton separation and energy reconstruction. They also have comparable sensitivity curves. However, γ-PhysNet SA[12] has a slightly better angular resolution above 500 GeV.

Dual Attention. In the dual attention mechanism, the reduction ratio controls the bottleneck of the channel-wise attention path that is a Squeeze-and-Excitation module. Therefore, I probe the same ratios as for the Squeeze-and-Excitation module: 2, 4, 8, 12 and 16 (the default one).

The performance on the classification task is similar for all the ratios. However, the smaller ratios (2, 4 and 8) obtain better results on the direction and energy reconstruction tasks. This is surprising as for the Squeeze-and-Excitation module alone all the ratios have comparable performance on these tasks. Among the smaller ratios, γ-PhysNet DA[8] has the most regular sensitivity curve.

C.2 Mid Cuts

Squeeze-and-Excitation. To extend this ablation study with the mid cuts, I compare the ratio selected with the high cuts (2) with the default one (16) and the ratio of 4 that obtained close results with the high cuts.

All the three ratios have comparable performance and sensitivity curves. However, γ-PhysNet SE[4] has slightly better angular and energy resolution curves above 1 TeV.

Self-Attention. For the self-attention method, I compare the ratio selected with the high cuts (12) with the default one (8). Both obtain similar results on all the tasks. However, γ-PhysNet SA[12] is 4 % faster for inference.

Dual Attention. For the dual attention mechanism, I compare the best ratio of the high cuts (8) with the default one (16). Both have similar performance on all the tasks. However, γ-PhysNet DA[16] has a slightly better angular resolution above 2 TeV.

C.3 Conclusion

This ablation study about the reduction ratio for the Squeeze-and-Excitation, self-attention and dual attention methods highlights that the performance of γ-PhysNet is rather insensitive to the variation of the ratio. Depending on the selection cuts, all the configuration evaluated obtain very similar results on gamma / proton classification, direction and energy reconstruction, and have comparable sensitivity curves. Noteworthy, with the mid cuts the results per attention method are globally indistinguishable. This can be explained by a much greater amount of training data compared to the high cuts.

The selection of the ratios for the study of the impact of attention methods presented in Section 4.2.4 is shown in Table C.1. It then relies on faint differences that may be not significant compared to the variability induced by different weight initialization. However, this stability of the results stresses the robustness of γ-PhysNet architecture.

Table C.1 – "Best" ratio for the three attention methods.

Attention	HC	MC
Squeeze-Excitation	2	4
Self-Attention	4	12
Dual Attention	8	16

D

Attention Mechanisms Performance with the Mid Cuts

This appendix presents the results of the attention study for the mid cuts (see Section 4.2.4 for details about the experiments). As I have a limited computing budget and the mid cut configuration is exploratory, I only train each configuration once.

Classification. On the classification task, as illustrated in Table D.1, all the models have comparable results.

Table D.1 – Mid Cuts. AUC, precision and recall of the gamma/proton classification task for the different models.

Model	AUC	Precision	Recall
γ-PhysNet	0.934	0.947	0.943
γ-PhysNet SE[4]	0.934	0.948	0.931
γ-PhysNet SA[12]	0.930	0.947	0.931
γ-PhysNet DA[16]	0.935	0.947	0.941

Sensitivity. Figure D.1 shows that all models have comparable sensitivity curves. As for the low cuts, they all obtain very good results at low energies, and outperform the MAGIC observatory below 200 GeV.

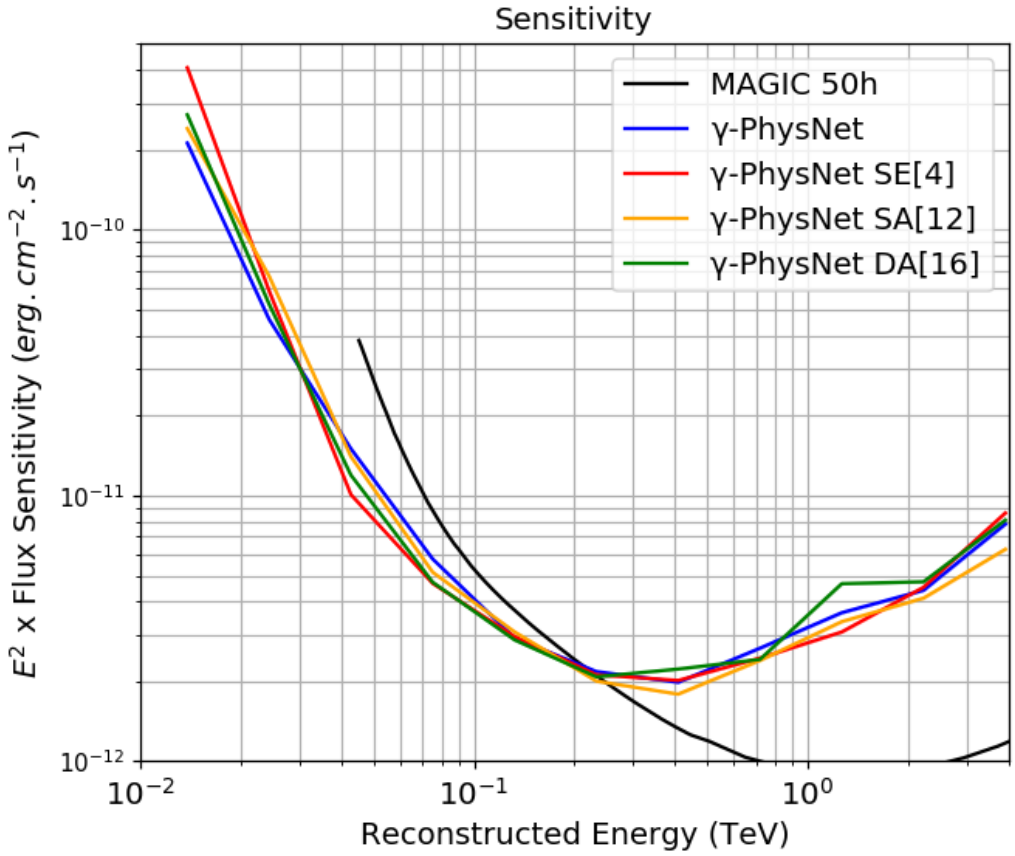


Figure D.1 – Mid cuts. Sensitivity. Comparison of the different attention mechanisms for γ-PhysNet.

Energy Reconstruction. On the energy reconstruction task, the results of all the models are similar below 1 TeV. Above, the models with Squeeze-and-Excitation and dual attention perform slightly better, improving the resolution up to 20% compared to the original γ-PhysNet. All models have comparable biases.

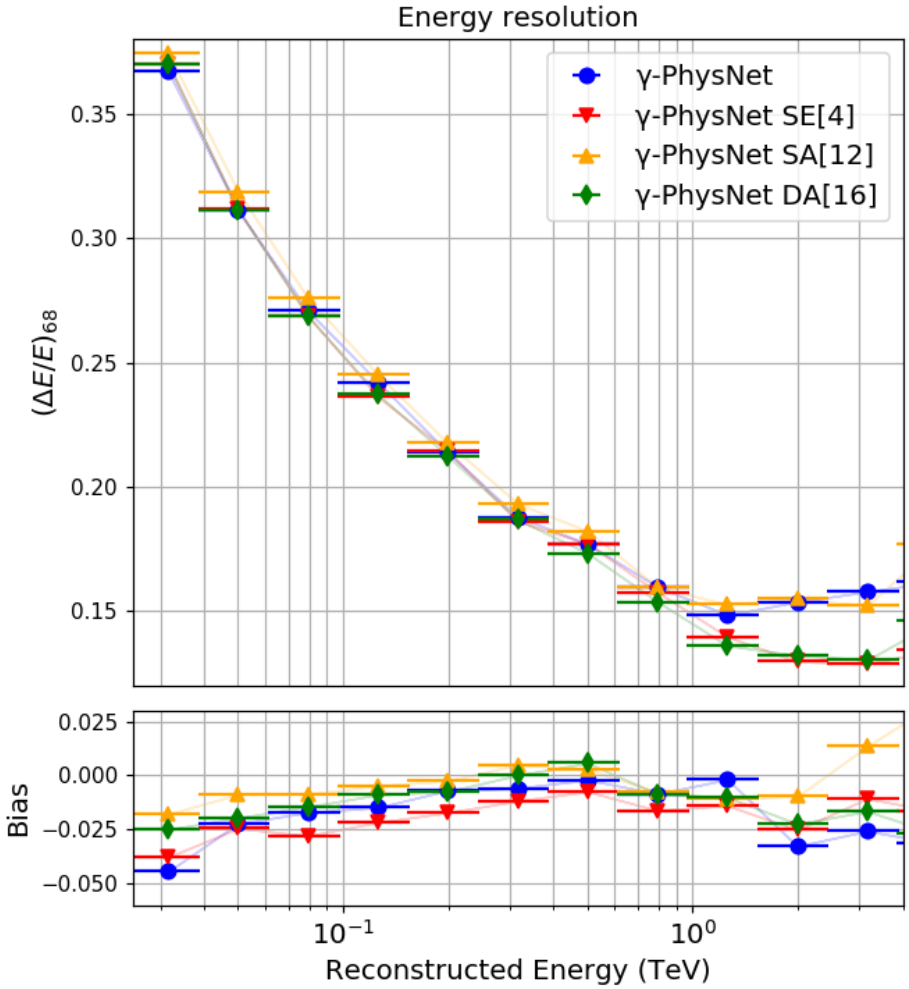


Figure D.2 – Mid cuts. Energy resolution curves of the different attention mechanisms for γ-PhysNet. The bias plot corresponds to the median relative error per energy bin.

Direction Reconstruction. Again, on the direction reconstruction task the models with Squeeze-and-Excitation and dual attention perform slightly better, improving the resolution by about 0.01° between 300 GeV and 2 TeV, up to 0.02° at 3 TeV. All models have comparable altitude biases. The model with dual attention has slightly smaller azimuth biases.

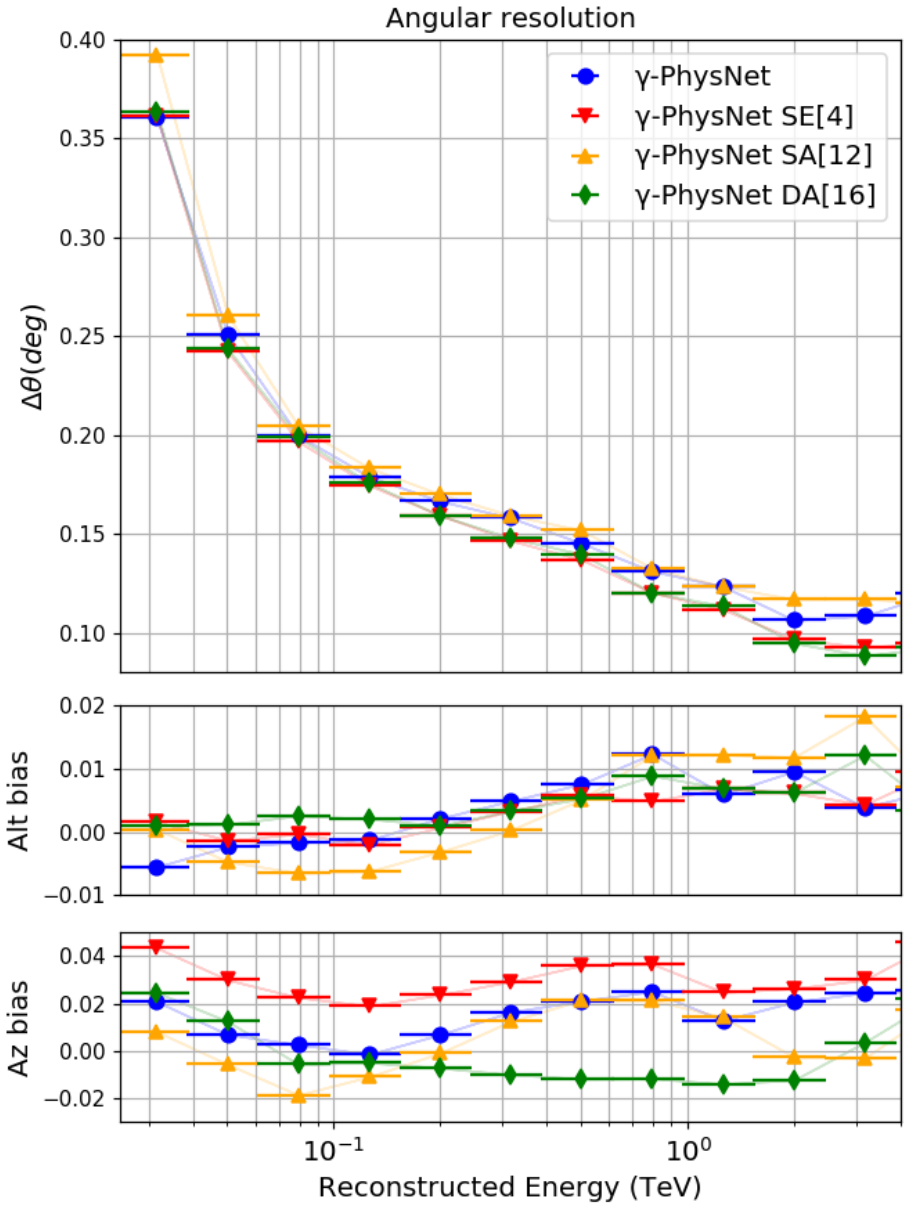


Figure D.3 – Mid cuts. Angular resolution curves of the different attention mechanisms for γ-PhysNet. The bias plots correspond to the median relative error per energy bin for relatively the altitude and the azimuth reconstructions.

Conclusion. All models have comparable results on the classification task and similar sensitivity curves. As for the high cuts and the low cuts, Squeeze-and-Excitation and dual attention models perform slightly better on the energy and the direction reconstruction tasks.

E

γ-PhysNet Receptive Field Computation

The computation of the γ-PhysNet backbone receptive field is rather straightforward. All the convolution kernels have the same size and the residual paths have a much larger receptive field than the identity ones, so we only need to take into account the convolutional path. However, we have to pay attention to the stride, as illustrated in Figure E.1. According to Equation 4.3, to compute the receptive field of layer l, we take into account the stride until layer $l-1$. We can then group layers based on the product of the stride of the $l-1$ layers to compute

$$r_0 = \sum_{l=1}^{20}\left((k_l-1)\prod_{i=1}^{l-1}s_i\right) + \sum_{l=21}^{38}\left((k_l-1)\prod_{i=1}^{l-1}s_i\right) + \sum_{l=39}^{55}\left((k_l-1)\prod_{i=1}^{l-1}s_i\right) + 1 \tag{E.1}$$

with $k_l = 3$ for every layer and $\prod_{i=1}^{l-1} s_i$ computed as shown in Figure E.1, resulting in

$$\begin{aligned} r_0 &= \sum_{l=1}^{20}((3-1)\times 1) + \sum_{l=21}^{38}((3-1)\times 2) + \sum_{l=39}^{55}((3-1)\times 4) + 1 \\ &= 20\times 2 + 18\times 4 + 17\times 8 + 1 \\ &= 249 \end{aligned} \tag{E.2}$$

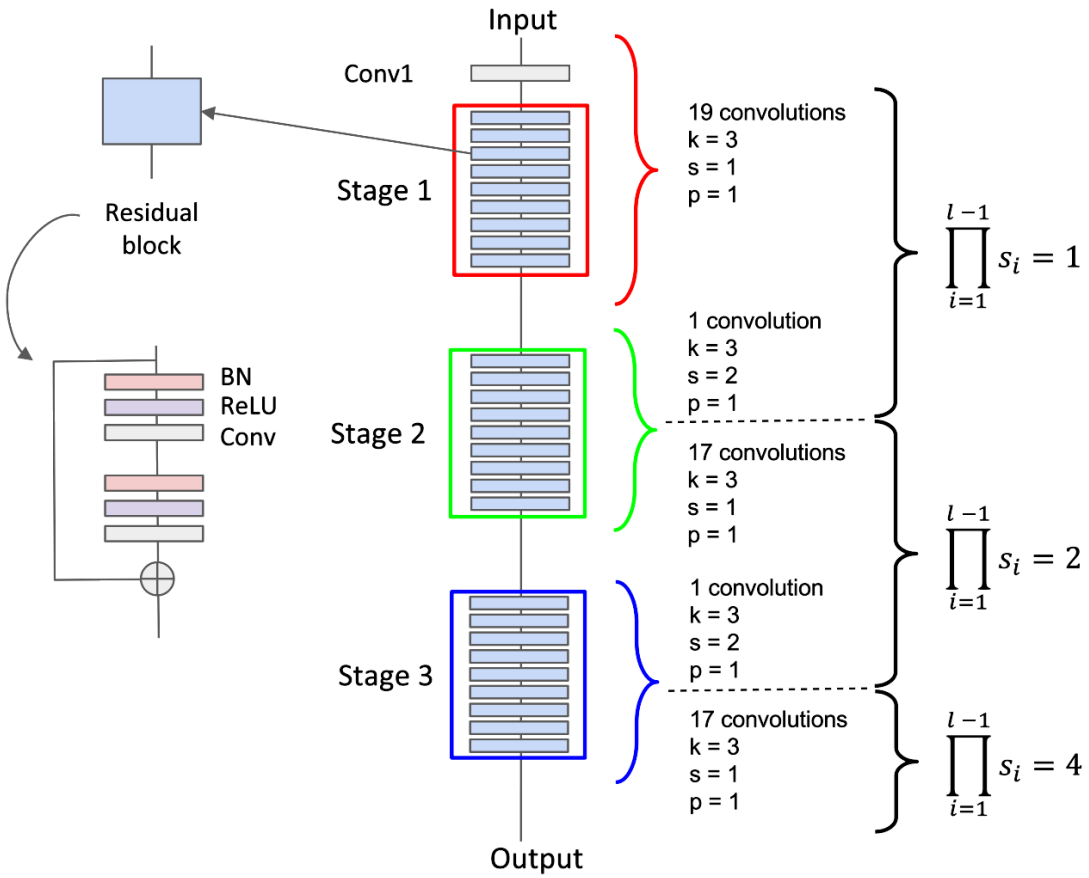


Figure E.1 – γ-PhysNet Receptive Field Computation.

Cherenkov Image Analysis with Deep Multi-Task Learning from Single-Telescope Data

Résumé

L'astronomie gamma consiste en l'observation des photons les plus énergétiques produits par les phénomènes astrophysiques violents. Leur étude permet de mieux comprendre les lois qui, notamment, gouvernent la création des étoiles et l'évolution des galaxies. Elle permet aussi d'explorer une nouvelle physique. Les télescopes Cherenkov situés sur Terre détectent indirectement les rayons gamma grâce à la gerbe électromagnétique qu'ils génèrent en pénétrant l'atmosphère. L'analyse du rayonnement gamma consiste en sa séparation du bruit de fond, les rayons cosmiques, et la reconstruction de son énergie et de sa direction incidente. Cette analyse est complexe, car les rayons cosmiques peuvent produire des images très similaires à celles des gammas et le rapport signal sur bruit est typiquement inférieur à 1/1000.
Le Cherenkov Telescope Array (CTA) constitue la nouvelle génération d'observatoire de l'univers à très haute énergie. Composé de plus de 100 télescopes répartis sur 2 sites, CTA aura une sensibilité 10 fois meilleure que les observatoires actuels tout en améliorant la précision de la reconstruction. En contrepartie, une fois achevé CTA produira une énorme quantité de données (210 Po/an) devant être analysées en temps réel. De plus, toutes les données acquises seront retraitées chaque année afin de bénéficier des améliorations de l'analyse. CTA met à mal les méthodes standard, trop lentes ou manquant de sensibilité à basse énergie. Il est donc nécessaire d'explorer d'autres pistes. S'appuyant sur les récentes avancées des réseaux de neurones artificiels, cette thèse propose une nouvelle approche deep learning pour analyser les données de CTA, en particulier celles du Large-Sized Telescope 1 (LST1), prototype installé sur site.
La première contribution de cette thèse est une méthode rendant possible l'application des algorithmes de deep learning aux images ayant des organisations de pixels quelconques. Celle-ci fournit les opérateurs convolution et pooling, déterminants dans les succès des réseaux de neurones. Cette méthode respecte le voisinage réel des pixels, permet d'éviter un prétraitement et une augmentation artificielle de la taille des images. Son intérêt par rapport aux méthodes classiques de ré-échantillonnage est démontré pour les images à pixels hexagonaux du LST1. La deuxième et principale contribution est une architecture multitâche, γ-PhysNet, inspirée de la physique qui réalise la reconstruction complète des évènements gamma. Celle-ci bénéficie d'un mécanisme d'attention la rendant plus robuste aux conditions d'initialisation. Sur les données de simulation utilisées pour préparer les algorithmes d'analyse, γ-PhysNet obtient des résultats significativement meilleurs, tant en sensibilité qu'en résolution spatiale, que la méthode standard combinant une extraction des paramètres Hillas et une analyse multivariée de type Random Forest. Le gain de sensibilité à basse énergie pourrait permettre d'améliorer l'étude des phénomènes transitoires. γ-PhysNet est également 800 fois plus rapide que la méthode la plus performante. Par ailleurs, cette thèse présente une analyse préliminaire du réseau proposé à l'aide d'une méthode d'explicabilité visuelle afin de mieux comprendre son comportement. Enfin, les premières données d'observation du "Crabe" prise par le LST1, qui est encore en phase de mise en service, sont analysées avec γ-PhysNet. Cette analyse très préliminaire met en évidence le besoin d'une meilleure adaptation aux données réelles. Enfin, la dernière contribution est un ensemble d'outils facilitant l'utilisation du deep learning avec les données de CTA et assurant la robustesse et la reproductibilité des résultats.

Mots-clés : réseau de neurones profond, apprentissage multitâche, convolution indexée, astronomie gamma

Abstract

Gamma-ray astronomy is the astronomical observation of the most energetic photons produced by violent astrophysical phenomena. Their study allows understanding better the physics ruling the birth of stars or the evolution of the galaxies. It also allows exploring a new physics. Ground-based Cherenkov telescopes detect indirectly gamma rays via the particle shower they generate when entering the atmosphere. The purpose of the image analysis is to estimate the energy and direction of the primary particle and to separate the gamma rays from the cosmic ray background. This step is complex because cosmic rays can generate very similar images and the signal-to-noise ratio is typically lower than 1/1000.
The Cherenkov Telescope Array (CTA) is the next generation of very high-energy universe observatories. Composed of more than 100 telescopes distributed on 2 sites, CTA will enhance the sensitivity by a factor of 10 compared to the current observatories, while improving the accuracy of the reconstruction. As a counterpart, once built, CTA will generate a tremendous amount of data (210 PB / year) to be analyzed. Moreover, all the data will be reprocessed yearly to benefit from the improvement of analysis models. Due to these properties of CTA, standard analysis methods either are too slow or lack sensitivity at low energies. We need then to explore other methods. Following the recent advances of artificial neural networks, this thesis proposes a new deep learning approach to analyze CTA data, especially the data from the Large-Sized Telescope 1 (LST1), the first on-site prototype built.
The first contribution of this thesis is an original method to apply the deep learning techniques to any kind of pixel organization. It provides convolution and pooling operators, that were crucial for neural networks successes. This method respects the real neighborhood of the pixels, and avoids preprocessing and image size increase. This work demonstrates its interest over standard resampling methods for the hexagonal pixel images of the LST1. The second and main contribution of this thesis is a physically inspired deep multi-task architecture to perform full event reconstruction. It benefits from an attention mechanism enhancing its robustness to initialization conditions. Its evaluation on the simulated data used to prepare analysis algorithms shows that γ-PhysNet has a significantly better sensitivity and spatial resolution than a standard method relying on the Hillas parameter extraction and a multivariate method, such as the Random Forest. In particular, it achieves very interesting sensitivity below 200 GeV, and could enhance the study of transient phenomena. γ-PhysNet is also 800 times faster than the state-of-the-art method. Besides, using a visual explanation method, this thesis presents a preliminary analysis of the model proposed to understand better its behavior. Finally, the first observation data from the "Crab" produced by the LST1, still in commissioning phase, are analyzed with γ-PhysNet. This very preliminary analysis highlights the need for a better adaptation to real data. Finally, the last contribution is a framework to ease the deep learning procedure with CTA data, and to ensure the traceability and reproducibility of the experiments.

Keywords : deep neural networks, multi-task learning, indexed convolution, gamma astronomy