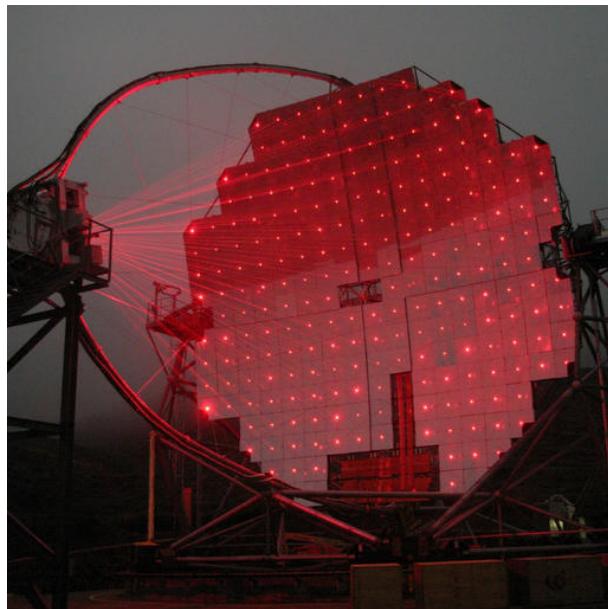


## Étape de pré-traitement pour télescope via machine learning



Thèse de Bachelor présentée par

**Yannis Perrin**

pour l'obtention du titre Bachelor of Science HES-SO en

**Informatique et systèmes de communication avec orientation  
Développement Logiciel**

**Août, 2024**

Professeur-e HES responsable

**Andres Upegui**

Mandant

**UNIGE**

Légende et source de l'illustration de couverture : Calibration automatique des miroirs de l'IACT MAGIC pendant la nuit. Source : commons.wikimedia.org par Robert Wagner, 2004

## TABLE DES MATIÈRES

<b>Remerciements . . . . .</b>	<b>v</b>
<b>Résumé . . . . .</b>	<b>vii</b>
<b>Liste de acronymes . . . . .</b>	<b>viii</b>
<b>Liste des illustrations . . . . .</b>	<b>x</b>
<b>Liste des tableaux . . . . .</b>	<b>xi</b>
<b>Introduction . . . . .</b>	<b>1</b>
<b>1 Chapitre 1 : Phénomène physique . . . . .</b>	<b>3</b>
1.1 La lumière Cherenkov . . . . .	3
1.2 Méthodes de détection . . . . .	5
1.3 Astronomie gamma existante . . . . .	5
a INTEGRAL . . . . .	5
b Fermi-LAT . . . . .	6
c MAGIC . . . . .	7
d H.E.S.S. . . . .	8
e VERITAS . . . . .	8
f HAWC . . . . .	8
1.4 Astronomie gamma future . . . . .	10
a CTAO . . . . .	10
<b>2 Chapitre 2 : Historique du projet . . . . .</b>	<b>11</b>
2.1 Projet initial . . . . .	11
a NUSES . . . . .	11
b Terzina . . . . .	12
2.2 Proposition de projet par l'UNIGE . . . . .	14
2.3 Délais de production . . . . .	14
2.4 Adaptation du projet . . . . .	15
a CTLearn . . . . .	15
b CTAO Data levels . . . . .	16
c Flux logiciel . . . . .	17
<b>3 Chapitre 3 : Méthodologie . . . . .</b>	<b>18</b>
3.1 Machine learning . . . . .	18
3.2 Réseaux neuronaux . . . . .	18
a Les perceptrons . . . . .	19
b Multi Layer Perceptron . . . . .	20
c Apprentissage . . . . .	20
d Réseau de neurones convolutif . . . . .	21
e Réseau de neurones récurrents . . . . .	22
f Auto-encodeurs . . . . .	23

g	Couches . . . . .	23
3.3	Données . . . . .	24
a	Pe Extractor . . . . .	25
b	Fichiers de simulation Corsika . . . . .	26
<b>4</b>	<b>Chapitre 4 : Résultats . . . . .</b>	<b>27</b>
4.1	Adaptation du simulateur Pe Extractor . . . . .	27
a	Création d'un dataset . . . . .	29
4.2	Premiers tests de réseaux neuronaux . . . . .	30
a	Détection booléenne de photon-électron . . . . .	30
b	Estimation calorimétrique de photon-électron par régression . . . . .	31
c	Métriques de performance des réseaux . . . . .	32
d	Entraînement avec Sample Weights . . . . .	34
e	Tests d'architecture RNN . . . . .	36
4.3	Fichiers de simulation Corsika . . . . .	37
a	Création d'un dataset . . . . .	42
b	Entraînement des réseaux de neurones . . . . .	44
4.4	Améliorations possibles . . . . .	45
<b>Conclusion . . . . .</b>	<b>46</b>	
<b>Annexes . . . . .</b>	<b>48</b>	
<b>Bibliographie . . . . .</b>	<b>48</b>	

## REMERCIEMENTS

*Je tiens à remercier ma famille pour son support pendant toute la durée de mes études universitaires. Je remercie aussi M. Andre Upegui Posada, Matthieu Heller et Tjark Miener et les membres de l'équipe de l'UNIGE pour leur collaboration technique tout au long de ce projet.*

# ÉTAPE DE PRÉ-TRAITEMENT POUR TÉLESCOPE VIA MACHINE

## LEARNING

## DÉVELOPPEMENT LOGICIEL

**Descriptif :** La lumière Cherenkov est un phénomène physique similaire à un boom supersonique dans le domaine électromagnétique. Lorsque des particules chargées entrent dans notre atmosphère à la vitesse de la lumière, elles interagissent avec les particules de celle-ci. Ceci commence une réaction en chaîne produisant une pluie de rayonnements électromagnétiques et/ou de particules élémentaires telles que des protons par exemple.

L'UNIGE travaille sur différents télescopes étudiant ce phénomène et souhaite améliorer la détection d'évènements intéressants physiquement pour réduire la quantité de données à stocker, transmettre et traiter. Pour cela, il est imaginé d'ajouter une étape de pré-traitement pour chaque pixel d'une caméra d'un télescope pour y filtrer le bruit des capteurs et effectuer une estimation calorifique du nombre de photons détectés.

L'étape de pré-traitement utilisera le modèle de machine learning le plus capable à discerner la ou les impulsions de photons à partir de la sortie analogique du capteur.

**Travail demandé :** Le travail demandé consiste à :

- Établir une série de tests et de métriques permettant de comparer différents modèles.
- Tester différents modèles de réseau de neurones (CNN/RNN/KAN).
- Intégrer l'étape de pré-traitement dans le pipeline du logiciel de classification de pluies Cherenkov : CTLearn.
- Exploration de l'utilisation du pré-traitement pour améliorer les performances de la stéréoscopie de CTLearn.

Candidat-e :

**YANNIS PERRIN**

Filière d'études : ISC

Professeur-e(s) responsable(s) :

**ANDRES UPEGUI**

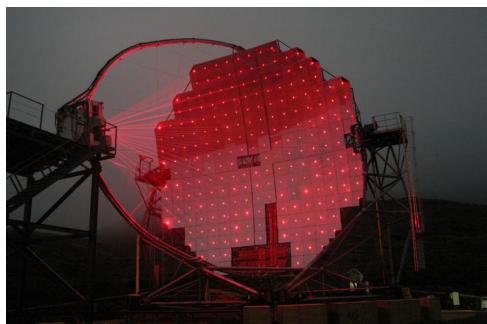
En collaboration avec : UNIGE

Travail de bachelor soumis à une convention de stage en entreprise : non

Travail soumis à un contrat de confidentialité : non

## RÉSUMÉ

La lumière Cherenkov est un phénomène physique similaire à un boom supersonique dans le domaine électromagnétique. Lorsque des particules chargées entrent dans notre atmosphère à la vitesse de la lumière, elles entament une réaction en chaîne qui produit une pluie de lumière atmosphérique. L'observation de ces pluies nous permet de mieux comprendre les mécanismes de la physique des particules. Dans cette optique, l'UNIGE travaille sur la mission pionnière du télescope spatial Terzina. Ce télescope a pour but de détecter et d'étudier les neutrinos de manière indirecte en observant les pluies de lumière atmosphérique produites par ceux-ci à partir de leur interaction avec le limbe de la Terre. Lors de sa conception, il s'est avéré que le télescope recueillera plus de données qu'il sera capable de transmettre au sol. Le but initial de ce projet consistait à créer un modèle de réseau de neurones embarquable à bord du satellite, afin de réduire l'utilisation de la bande passante ; pour cela, le réseau neuronal effectuera une analyse des données des capteurs afin de détecter la quantité de photons provenant d'une pluie atmosphérique afin d'aider à cibler des événements scientifiquement plus importants. Suite à des délais de production de certains composants essentiels au fonctionnement du réseau neuronal à bord du satellite, le projet a été adapté pour des télescopes au sol. Ceux-ci rencontrent des problèmes similaires de haute utilisation de stockage et de ressources de calcul à cause de la quantité astronomique de données recueillies. Un réseau de neurones similaire pourrait donc leur être utile. Lors de ce travail, j'ai commencé par étudier le phénomène physique et le fonctionnement des capteurs. J'ai ensuite pris en main les différents simulateurs de données. Avec ces données, j'ai testé différentes architectures de réseau neuronaux pour trouver un modèle performant.



Candidat-e :

**YANNIS PERRIN**

Filière d'études : ISC

Professeur-e(s) responsable(s) :

**ANDRES UPEGUI****En collaboration avec : UNIGE**Travail de bachelor soumis à une convention de stage  
en entreprise : non

Travail soumis à un contrat de confidentialité : non

## LISTE DE ACRONYMES

**ASIC** Application Specific Integrated Circuits. 13, 14, 15, 30

**CERN** Conseil Européen pour la Recherche Nucléaire. 26

**CNN** Convolutional Neural Network. 23, 33, 34

**CTAO** Cherenkov Telescope Array Observatory. 10, 15, 16

**EM** Électromagnétique. 3, 4, 5

**FPGA** Field Programmable Gate Arrays. 13, 14, 15

**HEPIA** Haute École du paysage, d'ingénierie et d'architecture de Genève. 11, 14

**IACT** Imagin Athmospheric (or Air) Cherenkov Telescope. 7, 9, 10

**INFN** Istituto Nazionale di Fisica Nucleare. 11

**LE** Low Energy. 5, 12

**LEO** Low Earth Orbit. 12

**LST** Large-Sized Telescope. 10, 15, 26, 37, 42

**MST** Medium-Sized Telescope. 10

**NASA** National Aeronautics and Space Administration. 6, 8

**pe** Photon-Électron. 25, 29, 30, 31, 32, 33, 35, 36, 43, 44, 45, 46

**RNN** Recurrent Neural Network. 23, 36

**SiPM** Silicon Photomultiplier. 12, 13

**SST** Small-Sized Telescope. 10

**UHE** Ultra High Energy. 12

**UNIGE** Université de Genève. 1, 11, 14, 15, 18, 25, 26, 37

**VHE** Very High Energy. 7, 8

## LISTE DES ILLUSTRATIONS

1.1	Advanced Test Reactor core, Idaho National Laboratory . . . . .	3
1.2	Diagramme de pluies EM et hadroniques . . . . .	4
1.3	Création de paires . . . . .	5
1.4	Rayonnement continu de freinage . . . . .	5
1.5	Illustration du satellite INTEGRAL . . . . .	6
1.6	Illustration du Fermi Gamma-ray Space Telescope en orbite . . . . .	7
1.7	Photo des deux télescopes MAGIC . . . . .	7
1.8	Photo de l'installation H.E.S.S . . . . .	8
1.9	Fred Lawrence Whipple Observatory . . . . .	9
1.10	The HAWC Observatory . . . . .	9
1.11	Rendu 3d de CTAO-North . . . . .	10
2.1	Mission NUSES . . . . .	11
2.2	Orbite héliosynchrone de Terzina . . . . .	12
2.3	Vue de la configuration optique de Terzina . . . . .	13
2.4	Flux des données CTAO . . . . .	16
2.5	Flux du logiciel CTLearn . . . . .	17
3.1	Illustration d'un perceptron . . . . .	19
3.2	Comparaison de données séparables ou non linéairement . . . . .	19
3.3	Exemple d'un perceptron multi-couches . . . . .	20
3.4	Exemple d'une descente de gradient en 3D . . . . .	21
3.5	Illustration d'une couche convulsive de réseau neuronal . . . . .	22
3.6	Illustration de réseau de neurones récurrents . . . . .	22
3.7	Schéma d'auto-encodeur . . . . .	23
3.8	Illustration du principe de Pooling . . . . .	24
3.9	Exemple de signal simulé . . . . .	26
4.1	Exemple de données générées par "pe_extractor" . . . . .	28
4.2	Exemple de séparation des données par fenêtre coulissante . . . . .	29

4.3	Diagramme d'architecture du premier CNN à classification simple . . . . .	30
4.4	Exemple de visualisation d'une inférence d'un réseau . . . . .	31
4.5	Diagramme d'architecture du CNN à multiples sorties . . . . .	32
4.6	Matrice de confusion . . . . .	33
4.7	Différence de métriques entre deux CNN . . . . .	34
4.8	Perte de performance en entraînant avec des Sample Weights . . . . .	35
4.9	Diagramme d'architecture du RNN simple . . . . .	36
4.10	Comparaison des métriques entre CNN et RNN . . . . .	37
4.11	Structure des fichiers binaires de simulations Corsika . . . . .	39
4.12	Exemple de données générées par simulations Corsika . . . . .	40
4.13	Données Corsika normalisées . . . . .	41
4.14	Signaux concaténés de chaque capteur d'une pluie Cherenkov . . . . .	42
4.15	Comparaison entre un dataset uniformisé et non uniformisé générés sur 100 pluies Cherenkov . . . . .	43
4.16	Distribution du dataset généré pour les entraînements . . . . .	43
4.17	Métriques d'entraînement des modèles sur les données Corsika . . . . .	44
4.18	Exemple de données Corsika difficilement discernables . . . . .	45

## **LISTE DES TABLEAUX**

1.1	Techniques de détection en fonction de la classification énergétique des rayons gamma . . . . .	6
1.2	Complementary detection characteristics of imaging air Cherenkov telescopes (IACTs) and extensive air shower arrays. . . . .	9

## INTRODUCTION

Dans le cadre de mon travail de bachelor, j'ai été choisi pour travailler avec le groupe "High-Energy Multi-Messenger" de l'Université de Genève (UNIGE). Ce groupe effectue des recherches expérimentales sur les particules provenant d'astres lointains à l'aide de satellites et de télescopes. Leurs activités se concentrent majoritairement sur les rayons gamma, les rayons cosmiques et les neutrinos. Mon travail consiste à aider cette équipe pour traiter des données concernant les rayons gamma et cosmiques à travers le phénomène physique de la radiation Cherenkov.

Ce phénomène se produit lorsqu'une particule chargée se déplace à une vitesse supérieure à celle de la lumière dans un médium diélectrique transparent. L'exemple le plus notable de ce phénomène est la lumière bleutée produite dans l'eau autour d'un réacteur nucléaire. Cette lumière est due aux rayons gamma produits par la réaction nucléaire. Ces rayons, à cause des immenses forces électromagnétiques, se déplacent momentanément plus rapidement que la vitesse de la lumière dans l'eau. On peut comparer ce phénomène à un boom supersonique mais pour de la lumière.

Les chercheurs de l'[UNIGE](#) s'intéressent cependant moins aux rayons gamma produits par des installations humaines qu'aux rayons et particules cosmiques provenant de l'espace. Ces rayons et particules extraterrestres produisent aussi une radiation de Cherenkov mais dans le médium de notre atmosphère. Lorsque le phénomène Cherenkov débute dans l'atmosphère, on appelle cela la "lumière Cherenkov". Ce rayonnement va créer un cône de particules chargées qui sera détecté au sol à l'aide de télescopes.

Aujourd'hui, ce phénomène est déjà étudié par de nombreux télescopes terrestres mais son observation est limitée par divers bruits. Pour pallier à ceux-ci, il est prévu d'étudier ce phénomène depuis un télescope en orbite afin de les éviter au maximum. Cependant, le satellite envisagé possède des limitations techniques, notamment sur la quantité d'informations qu'il peut envoyer ou recevoir. Mon implication dans ce projet est de présenter un modèle de machine learning qui, en effectuant une étape de pré-traitement, aidera à cibler les événements détectés et, possiblement, à réduire la quantité de données à transmettre.

La première partie de ce rapport portera sur une explication du phénomène physique et de la façon dont il est détecté physiquement. Une deuxième partie sera consacrée au déroulement du

projet ainsi qu'aux intérêts scientifiques que celui-ci présente. Une troisième partie expliquera de plus près les technologies et méthodes utilisées au cours de ce projet. Le dernier chapitre présentera la réalisation du projet ainsi que les résultats obtenus suivi d'une conclusion.

## CHAPITRE 1 : PHÉNOMÈNE PHYSIQUE

### 1.1. LA LUMIÈRE CHERENKOV

L'effet Cherenkov a lieu lorsqu'une particule chargée électriquement traverse un milieu diélectrique transparent en excédant la vitesse maximale de la lumière dans ce même milieu. Lors de son passage, cette particule va exciter les autres particules du milieu et produire une réaction en chaîne. Cette dernière va donc créer un cône de particules tel que des électrons, positrons ou photons et même des particules élémentaires nommées hadrons. Cet effet est similaire au bang supersonique lorsque des objets dépassent le mur du son mais dans le spectre électromagnétique. Le cas le plus connu de cet effet est celui qui peut être observé autour des réacteurs nucléaires, dû à la lumière bleue qu'il engendre.

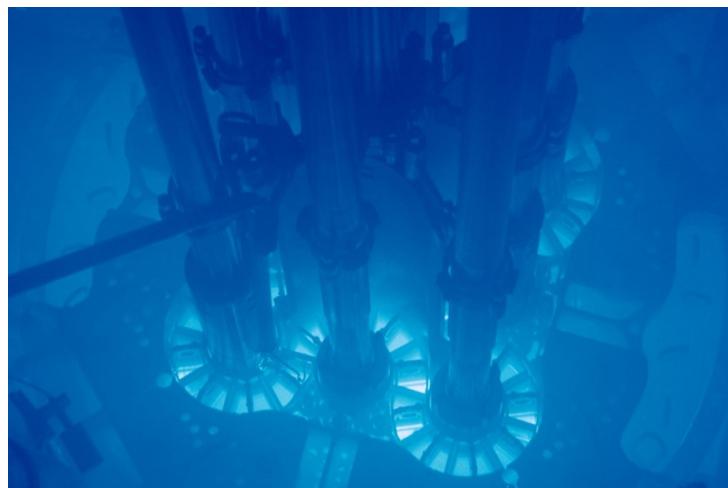


ILLUSTRATION 1.1 – Advanced Test Reactor core, Idaho National Laboratory. Source : [Lab09]

L'intensité de la lumière est communément quantifiée en électron-volt ou eV. Cette unité représente l'énergie cinétique gagnée lorsqu'un électron voit son énergie potentielle augmentée d'un Volt dans le vide.

La lumière Cherenkov atmosphérique peut produire deux types de pluies différentes : les pluies purement Électromagnétique (EM) composées uniquement d'électrons, positrons et photons ainsi que les pluies hadroniques qui elles sont composées de hadrons chargés électriquement, comme des protons.

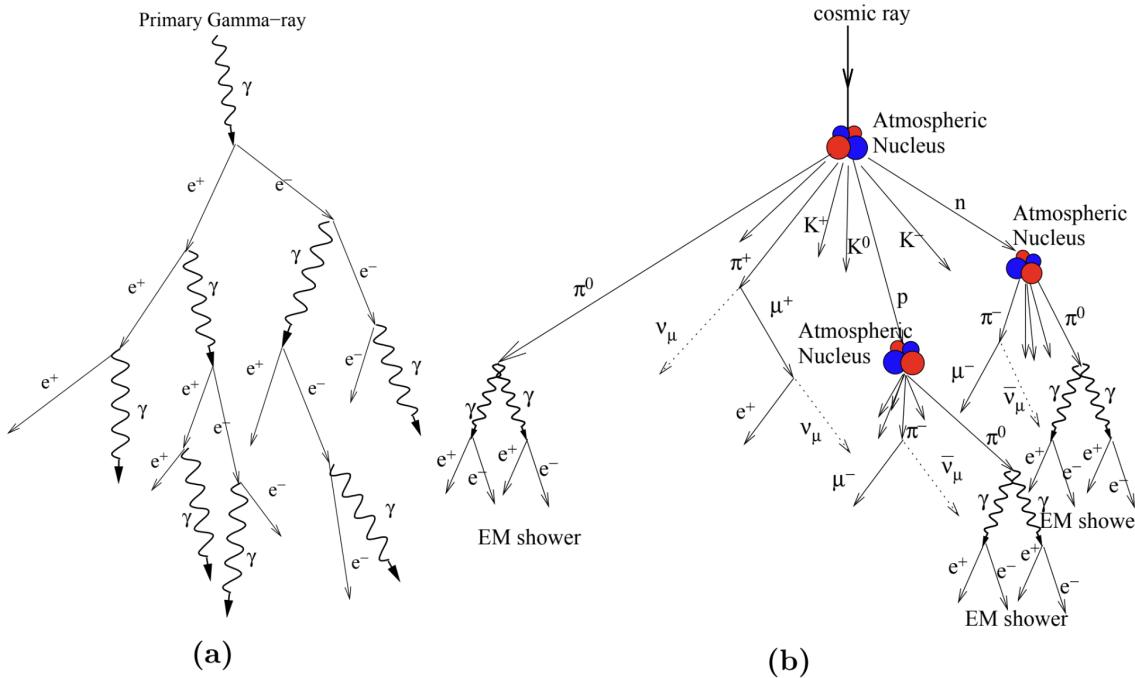
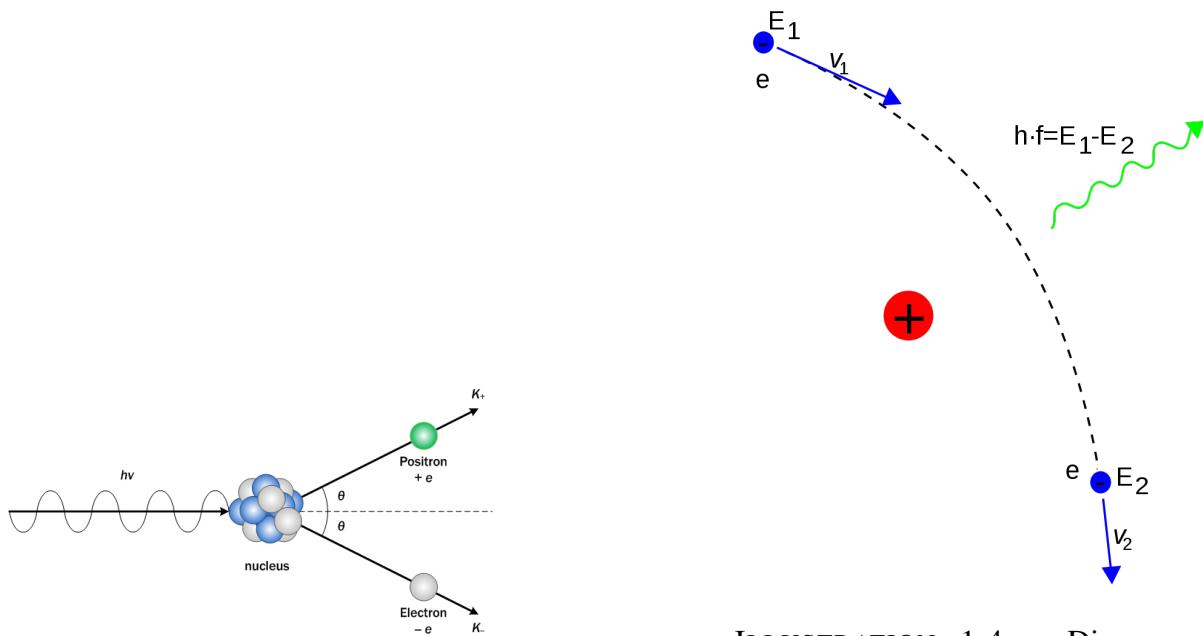


ILLUSTRATION 1.2 – Diagramme de pluies électromagnétiques pures (a) et pluies hadroniques (b). Source : I. Oya Vallejo. “Observation of active galactic nuclei with the Magic telescope”. UCM. PhD thesis. 2010.

Les pluies hadroniques sont généralement plus puissantes que des pluies EM et ont voit aussi qu'elles se décomposent au fur et à mesure en pluies EM. Pour les pluies EM, la production d'électrons et de positrons se fait via deux principes physiques : la production de paires et par rayonnement continu de freinage. La production de paires se produit lorsqu'une particule chargée à haute énergie interagit aux abords du noyau d'une autre particule. Suite à cette interaction un électron et un positron sont produits. Ces deux particules vont ensuite créer à leur tour des photons lorsqu'elles ralentissent autour d'autres noyaux. Ce ralentissement a comme conséquence de produire un photon.



L'intérêt scientifique se porte cependant plus sur les pluies EM pures à cause des rayons gamma qui débutent la réaction en chaîne. Comparés aux particules chargées comme des protons, les rayons gamma eux ne sont pas déviés par les champs électriques ou magnétiques pendant leur trajet à travers les astres. En détectant ces pluies purement EM, on peut en déduire leur provenance beaucoup plus simplement que pour des pluies hadroniques où il faut prendre en compte les différentes interactions astronomiques et atmosphériques.

## 1.2. MÉTHODES DE DÉTECTION

Les rayons gamma varient grandement en intensité de quelques MeV jusqu'à des dizaines de PeV et plus encore jusqu'au maximum 100 EeV. [Mie23] En fonction de l'énergie d'un rayon gamma, certaines techniques sont plus optimisées que d'autres, ici on retrouve les différentes techniques utilisées aujourd'hui :

## 1.3. ASTRONOMIE GAMMA EXISTANTE

### a. INTEGRAL

En l'an 2002, l'agence spatiale européenne a lancé le satellite INTEGRAL qui est le premier à avoir observé les spectres EM visible, x-ray et gamma en simultané. Concernant les rayons gamma, ce satellite est équipé de deux capteurs englobant la plage de rayons gamma Low

Notation	Gamme d'énergie	Technique de détection	Emplacement
LE	< 30 MeV	Photoélectrique, Compton	Espace
HE	30 MeV à 100 TeV	Création de paires	Espace
VHE	100 GeV à 30 TeV	Cherenkov (atmosphérique)	Terrestre
UHE	30 TeV à 30 PeV	Cherenkov (eau)	Terrestre
EHE	> 30 PeV	Fluorescence, Hybride	Terrestre

TABLEAU 1.1 – Techniques de détection en fonction de la classification énergétique des rayons gamma. Source : [Mie23] p. 17

Energy (LE) de 15keV à 10MeV [All22]

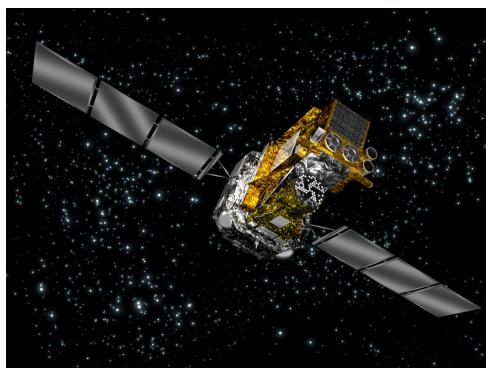


ILLUSTRATION 1.5 – Illustration du satellite INTEGRAL. Source : [Age02b]

## b. Fermi-LAT

Le satellite Fermi a été développé par la National Aeronautics and Space Administration (NASA) en collaboration avec des institutions françaises, allemandes, japonaises, italiennes et suédoises. Il a été envoyé en orbite en 2008 et il est en opération depuis lors. [AA24] Il transporte deux instruments de mesures : le Large Area Telescope (20 MeV à > 300 GeV) et le Gamma-ray Burst Monitor (8 keV à 40 MeV).

Le détecteur fonctionne via la détection de production par paires à l'intérieur du détecteur après une collision d'un photon. Les trajectoires et l'énergie dégagée par cet impact ressemble aux expériences des accélérateurs de particules.

L'observatoire fonctionne normalement en un mode de découverte qui scanne tout le spectre gamma du ciel. Il est capable de scanner l'entièreté du ciel en seulement deux orbites. Il a par exemple découvert les bulles de Fermi qui sont d'immenses sources de rayons gamma dans notre Voie lactée. Celles-ci proviendraient de grandes décharges d'énergie émises par le trou noir central de la galaxie il y a de cela plusieurs millions d'années.



ILLUSTRATION 1.6 – Illustration du Fermi Gamma-ray Space Telescope en orbite. Source : [AA23]

### c. MAGIC

MAGIC est une installation de deux télescopes **I**magin **A**thmospheric (or Air) **C**herenkov **T**elescope (IACT) complétée en 2009 à l'observatoire "Observatorio del Roque de los Muchachos" situé à La Palma aux îles Canaries à une altitude d'environ 2200m. [Gronda]

Les télescopes ont été conçus pour étudier les rayons gamma **V**ery **H**igh **E**nergy (VHE) de 30GeV à 100 TeV grâce à leurs miroirs de 17m de diamètre qui réfléchissent la lumière dans une grille hexagonale de 1039 tubes photomultiplicateurs, ce qui leur permet de visualiser le ciel avec un champ de vision du ciel de 3,5°.



ILLUSTRATION 1.7 – Photo des deux télescopes MAGIC. Source : [Grondb]

## d. H.E.S.S.

Le **High Energy Stereoscopic System** était initialement une installation de 4 télescopes de 12m formant un carré de 120m de côté rendu opérationnel en 2004 dans la région de Khomas en Namibie. En 2012, un 5ème télescope de 28m et 580 tonnes a été ajouté au centre de ce carré, augmentant la sensibilité en la résolution angulaire du système. [Hof12]

Les quatre télescopes aux extrémités ont un champ de vision de  $5^\circ$  et sont équipés de caméras composées de 960 tubes photomultiplicateurs. Le télescope central possède aussi une caméra composée de 2048 photomultiplicateurs avec un champ de vision de  $3,2^\circ$ .



ILLUSTRATION 1.8 – Photo de l'installation H.E.S.S. Source : [Kle12]

## e. VERITAS

Le **Very Energetic Radiation Imaging Telescope Array System** est un projet financé par les USA, le Canada et l'Allemagne installé au Fred Lawrence Whipple Observatory dans l'Arizona. Il est composé de quatre télescopes de 12m d'ouverture avec des caméras chacune composées de 499 tubes photomultiplicateurs. [Ver04] L'installation a pour but d'étudier les rayons gamma **VHE** entre 50GeV et 50TeV et a notamment été utilisée pour complémenter la mission Fermi de la NASA.

## f. HAWC

HAWC pour **High altitude Water Cherenkov gamma-ray observatory** est un observatoire situé sur le flanc du volcan Sierra Negra au Mexique, à une altitude de 4100 mètres. Comme son nom l'indique, il observe la lumière Cherenkov dans de l'eau au lieu de l'atmosphère. Ceci permet d'étudier des rayons gamma de plus haute intensité, de 100GeV à



ILLUSTRATION 1.9 – Fred Lawrence Whipple Observatory. Source : [Ver04]

100TeV. [Col04]

Au lieu d'utiliser des photomultiplicateurs comme les télescopes précédents, HAWC utilise des compteurs de particules basés sur des scintillateurs. Cette différence de fonctionnement comparé aux IACT lui permet de meilleures performances dans certains domaines tel le champ de vision mais le rend plus sensible au bruit de fond par exemple :

	Télescope Cherenkov	Détecteur de Pluie Atmosphérique
<b>Gamme d'énergie</b>	Basse (<200 GeV)	Haute (>10TeV)
<b>Rejet de bruit de fond</b>	Excellent (>99.7%)	Modéré (>50%)
<b>Champ de vision</b>	Petit (<2°)	Large (>45°)
<b>Disponibilité</b>	Basse (5%-10%)	Haute (>90%)

TABLEAU 1.2 – Complementary detection characteristics of imaging air Cherenkov telescopes (IACTs) and extensive air shower arrays, traduit. Source : [Col04]



ILLUSTRATION 1.10 – The HAWC Observatory. Source : J. Goodman, Nov. 2016 [Col04]

## 1.4. ASTRONOMIE GAMMA FUTURE

### a. CTAO

Le Cherenkov Telescope Array Observatory (CTAO) est le plus grand projet d'observatoire IACT au monde. Il est prévu de construire 60 télescopes répartis sur 2 sites, le premier (CTAO-North) dans l'hémisphère nord aux îles Canaries sur l'île La Palma et le deuxième (CTAO-South) à Paranal, Chili.

L'observatoire est conçu pour étudier les rayons gamma de 20GeV à 300TeV avec plus de 60 télescopes de différentes tailles, Small-Sized Telescope (SST), Medium-Sized Telescope (MST) et Large-Sized Telescope (LST) qui peut atteindre 45m de haut Pour cela chacun des types de télescopes a été conçu pour répondre au mieux à certaines gammes d'énergie. Pour la gamme principale de l'installation (150GeV à 5TeV), ce sont 23 MST qui sont prévu d'être construire. 37 SST sont prévus et ils seront optimisés pour détecter les énergies  $> 5\text{TeV}$  et jusqu'à 4 LST sont prévus pour détecter les gammes d'énergie  $< 150\text{ GeV}$ .



ILLUSTRATION 1.11 – Rendu 3d de CTAO-North. Source : Gabriel Pérez Díaz, IAC [CTA17]

## CHAPITRE 2 : HISTORIQUE DU PROJET

### 2.1. PROJET INITIAL

En novembre 2023, l'**UNIGE** a contacté l'**Haute École du paysage, d'ingénierie et d'architecture de Genève (HEPIA)** afin de créer un projet commun sur la recherche d'un modèle de réseau neuronal à très basse consommation et embarquable pour le projet du télescope spatial Terzina. Le but de ce modèle serait d'aider à la décision d'événements scientifiquement intéressants détectés par le télescope afin de réduire la quantité d'informations à transmettre au sol. Il est prévu d'envoyer le télescope Terzina à bord du satellite NUSES.

#### a. NUSES

Le projet **NeUtrino and Seismic Electromagnetic Signals** est une mission spatiale pionnière qui a vu le jour via la collaboration entre différentes universités et organisations gouvernementales. [Tri23] Notamment l'**UNIGE**, l'Istituto Nazionale di Fisica Nucleare (INFN) et la NASA. Son but est d'envoyer un satellite dans l'espace avec deux instruments scientifiques à bord, Zirè et Terzina, qui mèneront des expériences une fois en orbite.

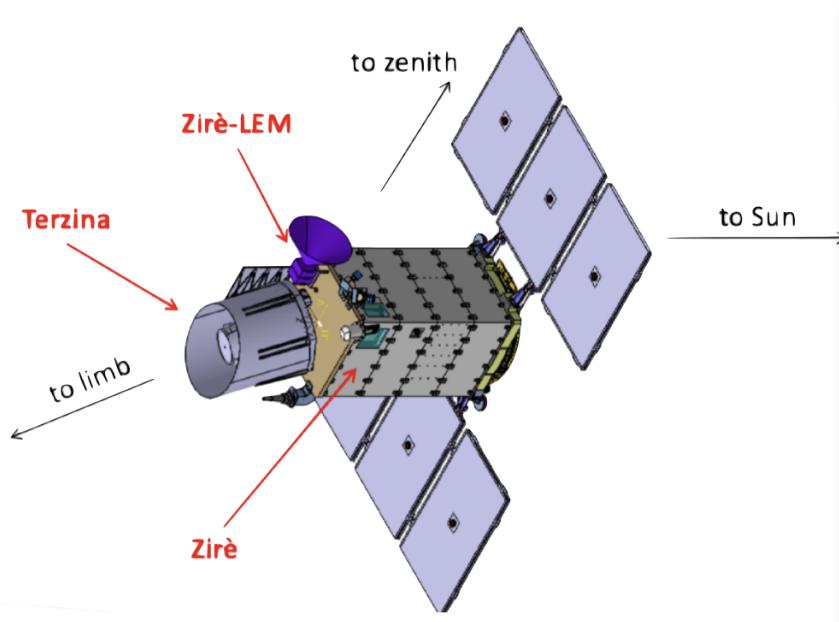


ILLUSTRATION 2.1 – Mission NUSES. Source : [Tri23]

**Zirè** a pour but d'observer le rayonnement cosmique LE (<250 MeV) pour étudier de plus près la ceinture de Van Allen, la météo spatiale et les interactions entre les lithosphère, ionosphère et magnétosphère.

**Terzina** a pour but de tester de manière concrète les technologies qui pourraient être utilisées pour étudier des rayonnements cosmiques Ultra High Energy (UHE) > 100 PeV et de détecter des neutrinos via les pluies atmosphériques de lumière Cherenkov qu'ils produisent.

Le satellite sera placé en orbite terrestre basse de manière héliosynchrone. L'orbite terrestre basse ou **Low Earth Orbit (LEO)** est définie comme toutes les orbites plus proches que 1000 km au dessus de la surface de la terre.[Age20a] L'orbite héliosynchrone est une orbite presque polaire (naviguant du nord au sud ou inversement) où le satellite passe au dessus d'un même point à la même heure solaire. Cela implique que son orientation en rapport au soleil reste la même.

Pour Terzina, cela permet aux capteurs d'être toujours pointés en direction opposée au soleil pour éviter au maximum les rayons gamma qu'il émet.

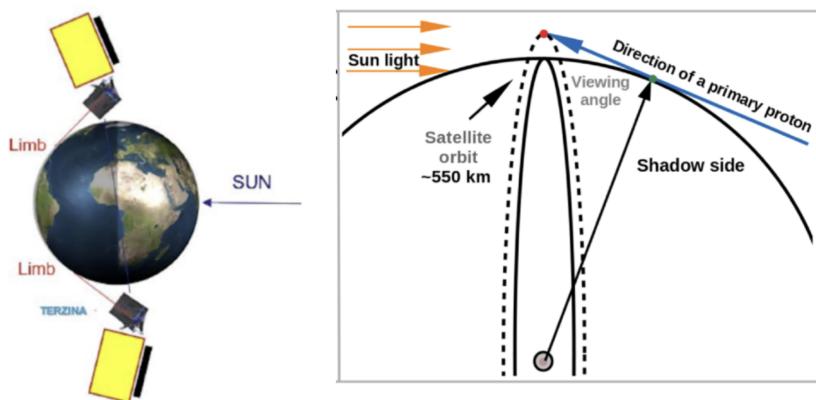


ILLUSTRATION 2.2 – Orbite héliosynchrone de Terzina. Source : [Tri23]

## b. Terzina

Comparé aux autres satellites qui ont étudiés des rayons gamma en orbite, Terzina est prévu d'être le premier télescope spatial qui détectera la lumière Cherenkov depuis l'espace. De plus, il permettra aussi d'étudier les performances de nouveaux capteurs Silicon Photomultiplier (SiPM) dans l'espace. Ceux-ci devraient détecter plus de photons et être plus robustes que leur contrepartie classique bien que ces nouveaux capteurs soient plus sensibles aux bruits de fond et à l'irradiation abondante hors de notre atmosphère.

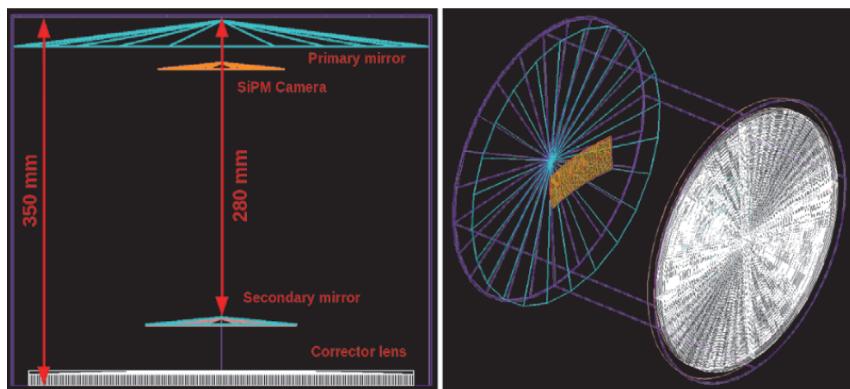


ILLUSTRATION 2.3 – En bleu les miroirs, en blanc la lentille, en orange la caméra et en violet les murs **Gauche :** Vue de dessus. **Droite :** Vue de côté. Source : [Bur23] p.2

Le télescope utilise une configuration à deux miroirs qui redirigent et concentrent les photons reçus sur une matrice rectangulaire de 10 modules SiPM, eux-mêmes possédant une résolution de 8x8 pixels, 640 pixels au total. [Bur23]

Cette forme rectangulaire a été décidée pour observer le limbe terrestre de manière optimale et elle est capable de détecter une coupe transversale de  $140 \times 360 \text{ km}^2$ . Derrière les 10 modules SiPM, il est prévu 10 Application Specific Integrated Circuits (ASIC) chacun possédant 64 canaux pour chacun des tubes photomultiplicateurs de la caméra. Ces ASIC sont conçus pour amplifier le signal de sortie puis le numériser avant d'être collectés par un Field Programmable Gate Arrays (FPGA) à bord du satellite.

En plus de leur rôle de numérisation, les ASIC ont aussi le rôle de déclencheur matériel. Ce rôle est important pour que le signal analogue des photomultiplicateurs ne soit converti en signal numérique que si nécessaire.

Le premier mécanisme de déclenchement, nommé "haut", arrive lorsqu'un pic d'énergie dépasse un seuil "haut" dans un seul canal d'un module SiPM ; chaque canal du module est numérisé et collecté par le FPGA.

Le deuxième mécanisme de déclenchement "bas" arrive lorsque deux pixels adjacents d'un même module SiPM dépassent ce seuil ou lorsque c'est l'un des 8 pixels voisins d'un autre module. Ensuite, les pixels voisins sont analysés pour déterminer si au moins deux d'entre eux ont aussi dépassé ce seuil "bas" ; dans ce cas l'FPGA va conserver cet événement. Tous ces tests se déroulent dans un laps de temps très court; de la numérisation du signal aux traitement et stockage sur le FPGA, il se passerait environ  $51.2\mu\text{s}$

## 2.2. PROPOSITION DE PROJET PAR L'UNIGE

La quantité de données récoltées par Terzina se révèle être trop importante pour être envoyée au sol, même après l'activation des déclencheurs intégrés au matériel. En effet, la capacité de bande passante de Terzina est limitée à 40Gbit par jour en transmission et seulement quelques Kbit par jour en réception. Ceci est dû à la configuration du satellite NUSES qui est aussi partagé entre les deux instruments Zirè et Terzina.

C'est au moment de la conception que l'UNIGE a eu l'idée d'ajouter un réseau de neurones à bord du [FPGA](#), capable de filtrer le bruit de chaque pixel pour n'en garder que les photons de pluies atmosphériques. Avec ce signal filtré, la décision d'envoyer ou non l'événement au sol serait plus simple. Ils ont donc proposé ce projet à l'[HEPIA](#) et j'ai été choisi pour travailler sur celui-ci.

Les ressources à bord du satellite étant coûteuses, le modèle avait comme contrainte d'être le plus petit possible afin d'utiliser le moins de place et de puissance de calcul. Il est prévu d'utiliser un modèle Keras pour le réseau de neurones, car il existe déjà des moyens de l'exporter vers des [FPGA](#). Le déploiement du modèle de réseau de neurones ne faisait pas partie intégrante de ce travail de semestre mais impliquait une restriction technique car la programmation du [FPGA](#) n'aurait été possible qu'avant le lancement.

En plus de cette utilisation en tant que filtre, l'équipe de l'UNIGE a théorisé l'idée que le réseau de neurones pourrait aussi compenser l'usure et l'irradiation des capteurs au fil du temps. Cette compensation pourrait se faire par un ajustement en vol des poids du réseau de neurones ou implicitement par l'architecture et l'entraînement du modèle directement.

## 2.3. DÉLAIS DE PRODUCTION

Malheureusement, en mars 2023, l'équipe s'occupant du développement des [ASIC](#) à annoncé des retards de production conséquents ne permettant pas de les installer sur Terzina. Il a donc été décidé de remplacer ces [ASIC](#) par un modèle déjà existant : CITIROC. Cependant, ces nouveaux [ASIC](#) ne sont pas capables de transmettre un signal comme les [ASIC](#) prévus jusqu'à présent et ne donnent que des informations sur le pic d'intensité atteint sur la durée d'échantillonnage.

Ce changement de matériel modifie les données en sortie du capteur, ne produisant plus de signal numérisé et rendant donc la capabilité d'un réseau de neurones difficile, voire impossible.

## 2.4. ADAPTATION DU PROJET

Voyant que l'ajout du réseau de neurones sur le **FPGA** du télescope n'est plus possible, l'**UNIGE** propose alors de poursuivre ce projet de recherche mais pour une utilisation sur des télescopes au sol. L'équipe de l'**UNIGE** participe également au groupe **CTAO** qui a pour but de créer deux observatoires dotés de trois types de télescopes différents.

Le réseau de neurones imaginé pour Terzina serait aussi utile pour ces télescopes. Les télescopes standard ne présentent pas les mêmes contraintes que la légèreté et la simplicité que Terzina doit respecter pour arriver en orbite. Sans ces contraintes, le **LST** atteint une résolution de 1'855 pixels et une fréquence d'acquisition de  $1GHz$ . Cette configuration collecte  $24Gbit$  de données par seconde avant les systèmes de déclencheurs matériels. [CTA24] De plus, la nouvelle génération de caméra sur laquelle l'**UNIGE** travaille, augmentera même la résolution jusqu'à 8'000 pixels.

L'ajout d'un réseau de neurones capable de réduire le bruit et d'estimer le nombre de photons détectés par pixel serait donc très utile en tant que point de décision supplémentaire pour garder ou non les différents événements déclenchés afin de réduire les ressources utilisées pour le stockage et le traitement de ces données.

De plus, il a été théorisé que le filtrage et l'estimation du nombre de photons que le réseau de neurones apporterait une amélioration du traitement des données en aval. Cela pourrait notamment améliorer la résolution angulaire (d'où provient la pluie de lumière Cherenkov) et une possible amélioration de la sensibilité du télescope aux événements moins énergétiques.

Pour tester l'amélioration de ces performances, il est prévu d'insérer ce réseau neuronal en tant qu'étape de pré-traitement des données pour chaque pixel avant de les transmettre à un logiciel de classification et d'analyse des pluies Cherenkov : **CTLearn**.

Mais, pour sa possible utilisation finale, ce réseau pourrait aussi être programmé dans des **ASIC** ou **FPGA** qui seraient intégrés dans les différents télescopes au sol.

### a. **CTLearn**

Cet outil, développé en coordination entre les universités de Genève et de Madrid, permet d'analyser des pluies Cherenkov détectées par les télescopes du **CTAO**. **CTLearn** va alors, pour chaque pluie essayer d'en déterminer son type : hadronique ou électromagnétique, sa provenance dans le ciel et effectuer des analyses sur l'énergie totale de cet événement. Ces

différentes informations sont ensuite utilisées par d'autres projets pour calculer d'autres valeurs d'intérêts scientifiques.

## b. CTAO Data levels

Pour faciliter la communication et les échanges entre les nombreuses équipes travaillant pour le CTAO, plusieurs niveaux de données ont été élaborés.

### Pipelines : from raw data to source catalogues

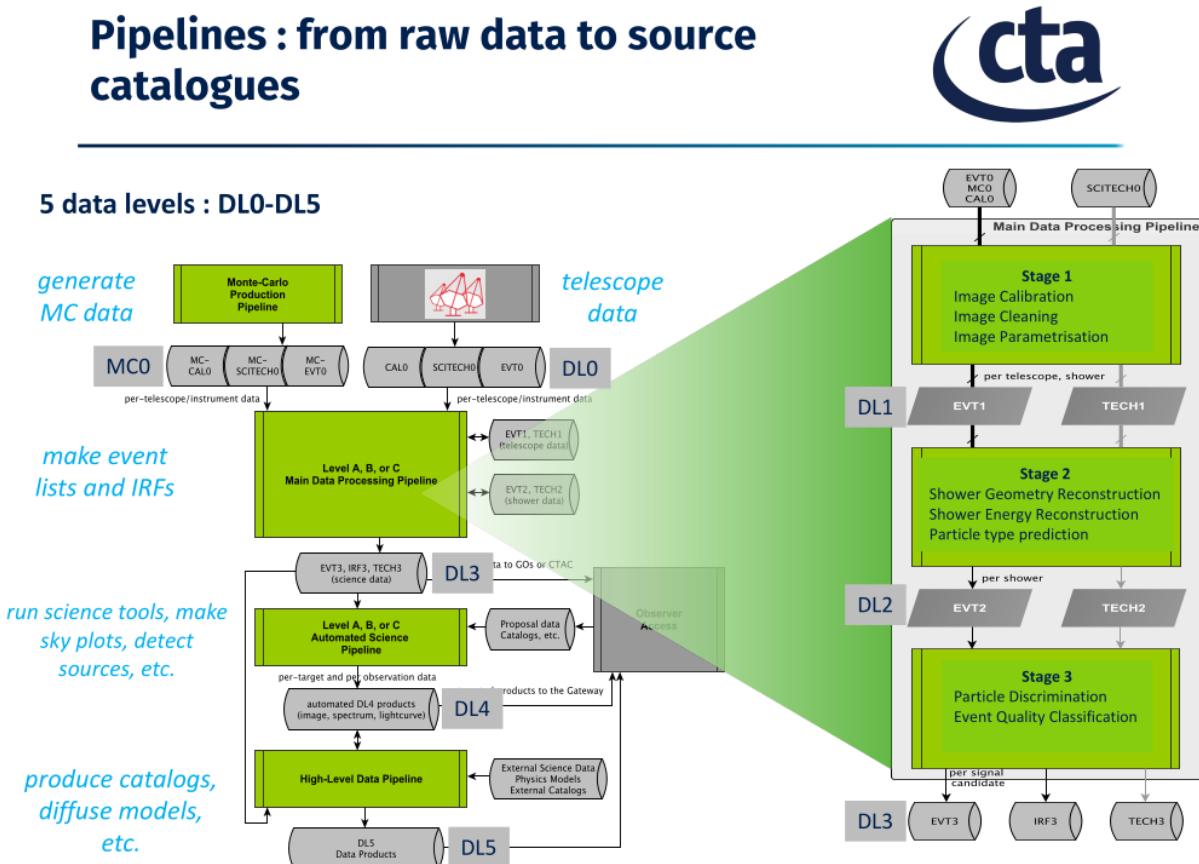


ILLUSTRATION 2.4 – Flux des données CTAO. Source : [CTA22], page 5

- MC0 : Données créées par des simulateurs Monte-Carlo
- DL0 : Données réelles récupérées par les télescopes
- DL1 : Images calibrées et nettoyées (par télescope/pluie Cherenkov)
- DL2 : Reconstruction de la pluie selon les images récupérées et prédiction de type hadronique/électromagnétique
- DL3 : Qualification et discrimination des pluies
- DL4 : Analyse des données de haut-niveau (spectres du ciel complet, etc.)
- DL5 : Agrégation des données pour en créer des catalogues.

### c. Flux logiciel

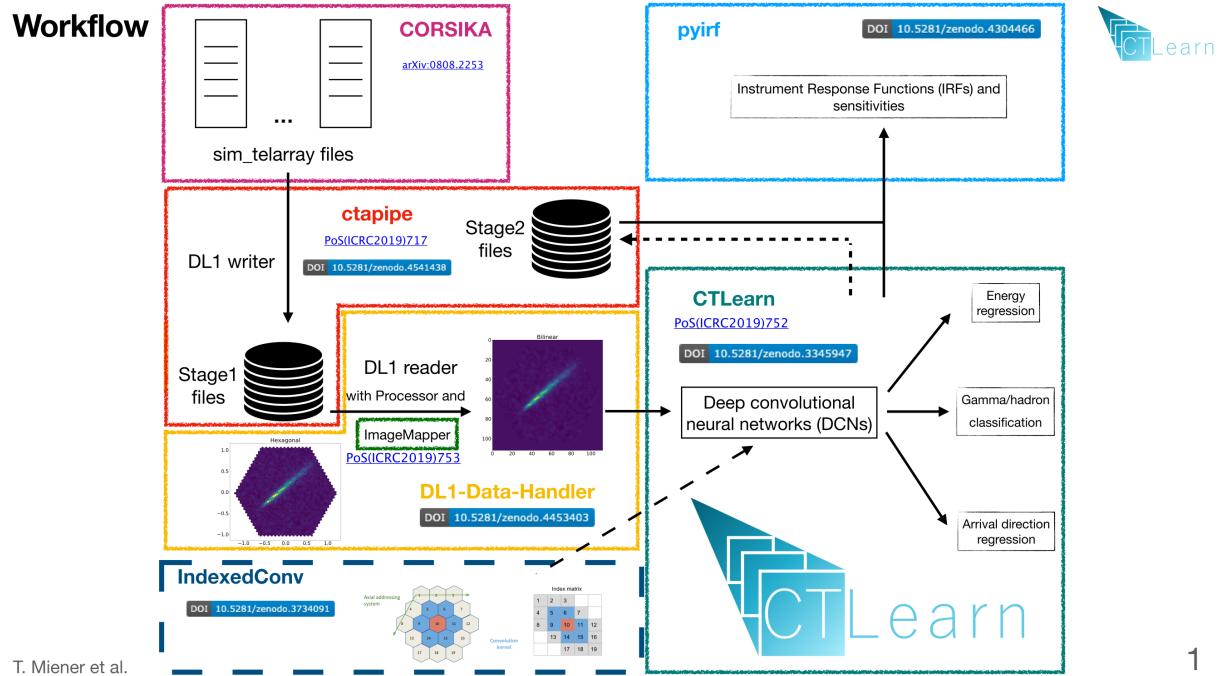


ILLUSTRATION 2.5 – Flux du logiciel CTLearn. Source : [Mie24b]

Les données utilisées par CTLearn sont les signaux numérisés de chaque capteur composant la caméra d'un télescope, données de niveau DL1. Ces données sont récupérées via les outils CTAPipe et DL1-Data-Handler. Ce projet de bachelor résiderait donc entre DL1-Data-Handler et CTLearn ou serait intégré à l'un d'entre eux comme étape de pré-traitement des signaux individuels.

## CHAPITRE 3 : MÉTHODOLOGIE

Dans ce projet, nous avons essayé de créer un programme capable d'analyser un signal temporel et d'en estimer la présence et la quantité de photons présent dans ce signal.

Pour cela, nous avons utilisé différents types de réseaux neuronaux qui sont eux-mêmes une technique de machine learning. Ce chapitre se porte donc sur le fonctionnement des technologies et outils utilisés pour ce projet.

### 3.1. MACHINE LEARNING

Le machine learning peut être n'importe quel forme de programme ou système capable d'apprendre ou d'améliorer ses performances en fonction des données qu'il traite.

Il existe deux types d'apprentissage pour le machine learning : supervisé et non supervisé. Le premier signifie que les données utilisées pour entraîner un système contient aussi les conclusions qu'il doit en tirer. Par exemple une image et un texte descriptif. Au fur et à mesure de l'analyse de ces différentes données, le système de machine learning va trouver des informations ou motifs auxquels il va attribuer les conclusions prédéfinies.

Pour l'apprentissage non supervisé, le système de machine learning utilise des données brutes qui n'ont pas de conclusions ou descriptions associées. Ce modèle va alors lui même essayer de trouver une manière de classifier ces données brutes ou de leur donner un score.

Dans le cadre de ce projet, l'équipe de l'[UNIGE](#) nous a fourni des données provenant de simulations des capteurs conçus pour les différents télescopes analysés. Lors de la création de ces simulations, les méta-données des photons simulés ont aussi été sauvegardées, ce qui nous donne accès à la "vérité" du moment exact où chaque photon a été détecté, formalisant le corps du projet comme une recherche d'architecture de réseau neuronal avec un apprentissage supervisé.

### 3.2. RÉSEAUX NEURONAUX

Alors, qu'est-ce qu'un réseau de neurones ou réseau neuronal exactement? Comme vu précédemment, c'est un modèle de machine learning qui est inspiré par le fonctionnement d'un cerveau humain.

Cette technique utilise des noeuds (neurones artificiels) interconnectés entre eux, créant des couches similaires à nos cerveaux. Chaque noeud peut être ou non connecté à un ou plusieurs

neurones supplémentaires permettant une infinité de configurations possibles. La configuration d'un réseau de neurones est aussi surnommée l'architecture du réseau.

### a. Les perceptrons

Chacun des noeuds qui composent un réseaux de neurones fonctionne comme une fonction mathématique, prenant des valeurs d'entrée et une valeur de sortie. Ces neurones sont aussi appelés perceptrons et suivent en général le fonctionnement suivant :

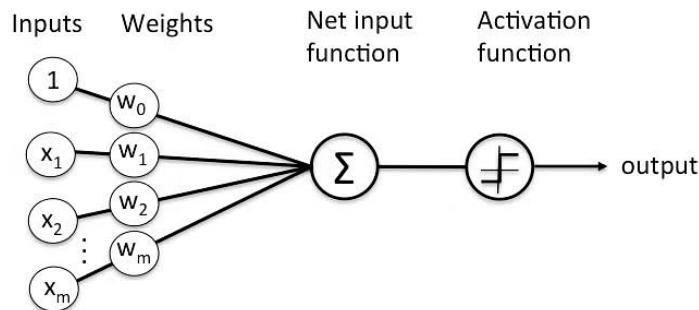


ILLUSTRATION 3.1 – Illustration d'un perceptron. Source : [Ban23]

Les paramètres d'un perceptron sont : un biais, et  $x$  données d'entrées avec des poids associés. Toutes ces valeurs pondérées sont ensuite additionnées ensemble et passées à une fonction d'activation qui donnera la valeur finale de sortie au perceptron. En général, la fonction d'activation est une sigmoïde, mais pourrait être n'importe quelle fonction mathématique qui s'adapterait mieux au problème que l'on essaye de résoudre.

Avec un seul perceptron, il est possible d'ajuster ces poids de manière à lui faire apprendre une motif de séparation linéaire en fonction des données d'entrée.

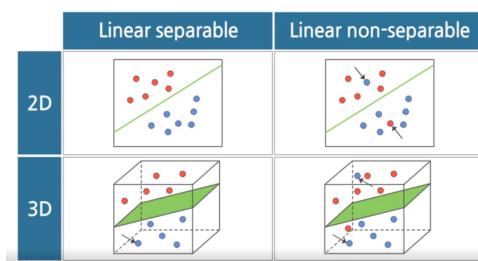


ILLUSTRATION 3.2 – Comparaison de données séparables ou non linéairement. Source : [Rit23]

Il a donc été rapidement imaginé d'interconnecter ces perceptrons pour permettre d'apprendre des problèmes non linéaires. Créant l'architecture la plus simple le "Multi Layer

Perceptron" ou perceptron multi-couches.

## b. Multi Layer Perceptron

Cette architecture de réseaux de neurones consiste simplement à créer 3 types de couches avec un nombre variable de perceptrons pour chacune d'entre elles :

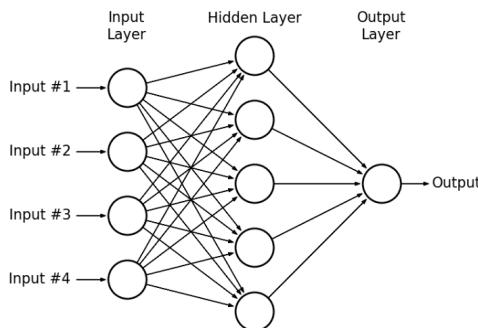


ILLUSTRATION 3.3 – Exemple d'un perceptron multi-couches. Source : [Zah15]

- 1 couche d'entrée : Cette couche représente seulement les valeurs d'entrées de manière organisée, par exemple une image de 10x10 pixels aurait une couche de taille 100.
- $x$  nombre de couches cachées : Ici un nombre de couches indéterminées peuvent être ajoutées. Chacune d'entre elles peuvent avoir un nombre de neurones différents dépendant du problème à traiter.
- 1 couche de sortie : Cette couche peut aussi avoir une taille variable dépendant de ce que le modèle doit accomplir. Par exemple, si le modèle a comme but de détecter si une image est un chat ou non, cette couche pourrait n'avoir qu'un neurone avec une fonction d'activation sigmoïde.

Entre chacune de ces couches, chaque neurone est connecté avec chacun des autres de la couche suivante, créant des couches surnommées "denses".

## c. Apprentissage

Peu importe le nombre de couches ou de neurones qui composent un réseau, il faut bien que celui-ci apprenne à résoudre le problème pour lequel il a été conçu. Cet apprentissage se repose sur les bases mathématiques que le perceptron utilise. Lorsque l'on calcule le résultat d'un perceptron, celui-ci va calculer une ou plusieurs valeurs en sortie. Lors d'un apprentissage supervisé, les valeurs de sortie attendues sont connues. On peut calculer une valeur d'erreur avec

une fonction mathématique. Nous avons donc volonté dans ce cas de minimiser cette fonction d'erreur. Et il est possible de le faire en modifiant les poids des valeurs d'entrées. La manière de trouver comment modifier ces poids s'appelle la descente de gradient.

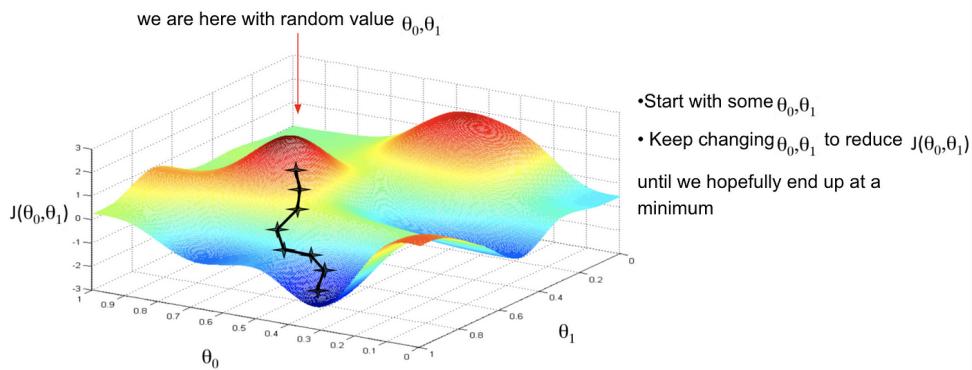


ILLUSTRATION 3.4 – Exemple d'une descente de gradient en 3D. Source : [Liu22]

Il est possible de trouver comment modifier les valeurs de chaque poids ou biais de chaque neurones afin de minimiser la fonction d'erreur et cela s'appelle la "backward propagation error" ou "backpropagation".

Un piège fréquent lors de l'entraînement de réseaux neuronaux est l'overfitting. Le processus d'apprentissage fonctionne parfois trop "bien" sur les données utilisées, jusqu'au point que le réseau de neurones y trouve des motifs qui ne sont pas ceux que l'on attend.

Par exemple, pour un réseau essayant de différencier des images de chaussures d'autres vêtements. Si, toutes les chaussures sont vertes dans les données d'entraînement, le réseau pourrait en déduire que c'est la couleur qui permet de les différencier. Cependant, lorsque le modèle se verra présenter une chaussure rouge ne faisant pas partie des données d'entraînement, celui-ci ne la reconnaîtra pas à cause de son mauvais entraînement.

## d. Réseau de neurones convolutif

Comme évoqué précédemment, il existe une infinité de configurations pour un réseau de neurones, alors nous ne regarderons que en détail les différentes configurations qui ont été envisagées dans ce projet.

L'une des premières architectures de réseau neuronal qui a été envisagée pour ce projet à été le réseau de neurones convolutif. Celui-ci consiste à certaines couches cachées d'une manière spécifique qui effectuera le même calcul qu'une convolution.

L'opération de convolution en mathématique permet généralement d'extraire des

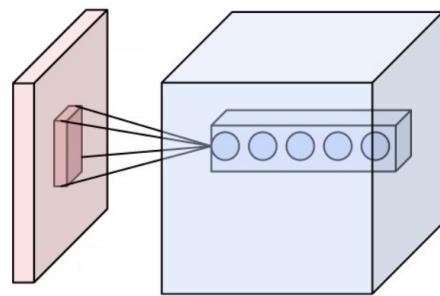


ILLUSTRATION 3.5 – Illustration d'une couche convective de réseau neuronal. Source : [Aph15]

informations d'une image comme le taux de variation d'intensité entre chaque pixel lors d'une détection de contours par exemple. Ici, la différence est que le noyau utilisé pour la convolution n'est pas un noyau provenant d'algorithme comme Sobel ou Canny mais ce sera l'apprentissage du réseau qui va établir un noyau le plus optimisé pour trouver les caractéristiques recherchées dans les données d'entrée.

### e. Réseau de neurones récurrents

Un autre type de réseau neuronal est le réseau de neurones récurrents. Cette architecture est adaptée à des problèmes comprenant des données séquentielles ou temporelles généralement comme de la détection automatique de parole ou de texte.

Cette architecture diffère des autres réseaux de neurones car elle contient un mécanisme de "mémoire" intégré dans le modèle. La mémoire est généralement implémentée en connectant la sortie de chaque neurone comme une entrée supplémentaire au prochain neurone de cette même couche.

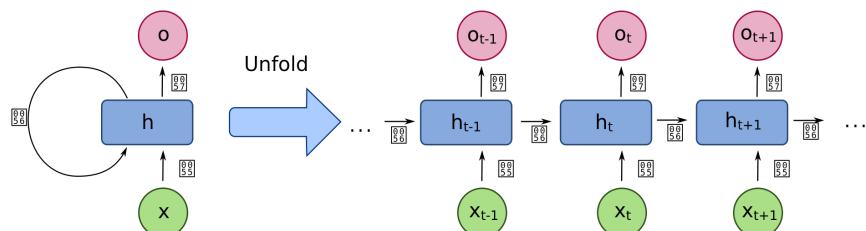


ILLUSTRATION 3.6 – Illustration de réseau de neurones récurrents. Source : [fde17]

Cette "mémoire" n'existe que de manière temporaire à chaque exécution du réseau.

Certaines autres configurations tel que les LSTM pour "Long and Short Term Memory" sont aussi capables de mémoriser des informations à plus long terme.

## f. Auto-encodeurs

Lors de la première partie de ce projet se focalisant sur les problématiques du télescope Terzina, un autre type de réseau avait été envisagé pour sa capacité à compresser des données : les auto-encodeurs.

L'auto-encodeur est une architecture de réseaux de neurones particulière qui comprend deux parties distinctes d'encodage et de décodage. Ces deux étapes peuvent aussi être perçues comme une compression et décompression successive. [Dav23]

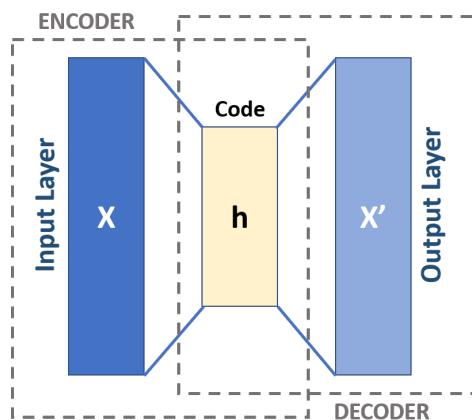


ILLUSTRATION 3.7 – Schéma d'auto-encodeur. Source : [Mas19]

En entraînant le réseau de neurones, celui-ci va apprendre à réduire les informations en entrée jusqu'à un minimum pour ensuite essayer de reproduire au mieux les données originales à partir de cette représentation réduite. Ceci résulte en général en une perte de précision des données lors de la compression.

## g. Couches

Les configurations vues jusqu'à maintenant peuvent aussi être combinées entre elles pour former de plus grands réseaux de neurones. Les réseaux neuronaux peuvent donc avoir une partie de réseau Convolutional Neural Network (CNN) et Recurrent Neural Network (RNN), etc. Chacune des couches peut ainsi être catégorisée par rapport à son utilité. Il existe aussi plusieurs couches spécifiques capables de travail spécifique. En voici quelques-unes utilisées au cours de ce travail ainsi que leur utilité :

## 1. BatchNormalisation

La couche "BatchNormalisation", en général utilisée en premier dans un réseau, sert à normaliser les données qui lui sont données. Cela sert à généraliser le reste du traitement du réseau, par exemple si l'on traitait des images avec des expositions différentes.

## 2. Flatten

Lors de l'utilisation de couches de convolution, il est possible d'en effectuer plusieurs en même temps et de les traiter parallèlement. On peut ensuite utiliser une couche "Flatten" pour rassembler ces données en une seule dimension.

## 3. Pooling

Ce type de couches dans un réseau neuronal permet de réduire le nombre de dimensions qui lui sont données en entrée. Ses utilités sont de réduire la quantité de calculs pour la suite du réseau et de diminuer l'overfitting lors de l'entraînement.

Les deux manières les plus communes pour effectuer un pooling est de garder la moyenne ou le maximum de chaque dimension :

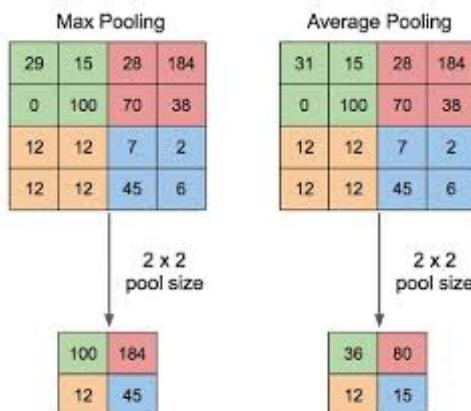


ILLUSTRATION 3.8 – Illustration du principe de Pooling. Source : [Yan19]

## 4. Dropout

Cette couche a la même utilité que le Pooling, en réduisant le nombre d'entrées qui sont passés à la suite du réseau afin d'en diminuer l'overfitting. Cependant, ici, seul un pourcentage des noeuds seront gardés au lieu de transformer les dimensions du réseau.

### 3.3. DONNÉES

Une autre grande partie de ce projet a été la gestion des données utilisées pour tester les différents réseaux de neurones.

### a. Pe Extractor

Le premier simulateur qui a été fourni provenait d'un projet existant de l'**UNIGE** s'appelant "pe\_extractor". Ce projet Python contient un générateur de signal avec du bruit NSB dans lequel des **Photon-Électron (pe)** sont insérés. Le concept de photon-électron ou **pe** est l'interaction physique lorsqu'un photon est détecté par un capteur et converti en charge électromagnétique analogique. Cette quantité électromagnétique est appelée photon-électron.

Le bruit NSB pour un télescope est le bruit de fond minimum lorsqu'il pointe le ciel pendant la nuit. Même dans ces meilleures conditions, de la lumière provenant des villes et de l'atmosphère est détectée par le télescope. De manière aléatoire et selon une fréquence donnée, le générateur ajoute au signal l'amplitude simulée d'un photon-électron. Cette amplitude est définie par un fichier d'impulsion qui dépend du capteur utilisé.

Le générateur fournit deux tableaux en sortie : le signal discrétilisé et des bacs contenant la vérité du nombre de **pe** ayant été injectés sur cette période.

Voici les paramètres importants du générateur et leur explication :

1. "pe\_rate\_mhz" : Définit la fréquence moyenne à laquelle un **pe** est injecté dans le signal. L'injection de **pe** est aléatoire en suivant une distribution Poisson.
2. "sampling\_rate\_mhz" : Définit la vitesse d'échantillonnage du signal. Pour Terzina, elle est de 200MHz.
3. "n\_sample" et "n\_sample\_init" : Le générateur a un mécanisme d'initialisation pour le bruit électrique qui peut prendre un certain nombre de cycles pour commencer à créer des informations cohérentes au niveau physique, ces deux paramètres peuvent donc contrôler un nombre d'échantillons à écarter au début de la génération.
4. "bin\_size\_ns" : Ce paramètre gère la période de comptabilisation des **pe** insérés dans le signal de sortie.
5. "shift\_proba\_bin" : Permet de décaler la comptabilisation de l'insertion de signal par un certain nombre de bacs. Ce système existe car l'impulsion a un temps de montée avant d'atteindre son pic, ce qui permet de décaler les bacs de vérité pour s'aligner avec le pic de l'impulsion.
6. "sigma\_smooth\_pe\_ns" : Ce paramètre sert à répartir le compte des **pe** insérés sur les bacs voisins en ns pour éviter une vérité entière.

## 7. "noise\_lsb" : Est une quantité de bruit électronique.

Normalement, les paramètres sont tels que, le nombre de bacs simulés est plus grand que le nombre d'échantillons générés pour le signal. Une première adaptation a été programmée pour regrouper ces bacs sur la même fréquence d'échantillonnage du signal :

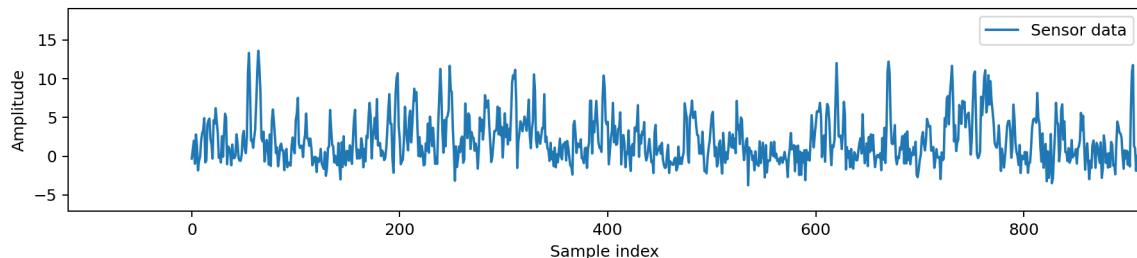


ILLUSTRATION 3.9 – Exemple de signal simulé.

## b. Fichiers de simulation Corsika

Le premier simulateur de signal provenant de "pe\_extractor" ne simule que des interférences de type NSB. Pour se rapprocher des données réelles des caméras, il a été choisi de récupérer des fichiers de simulation de pluies Cherenkov pour le LST.

Ces fichiers de simulations contiennent énormément de données, de la configuration du télescope jusqu'à l'impact des photons provenant des pluies Cherenkov. Ceux-ci sont stockés sur le partage de fichiers du cluster Yggdrasil de l'[UNIGE](#) sous un format de fichier ROOT.

ROOT est un framework logiciel conçu par le Conseil Européen pour la Recherche Nucléaire ([CERN](#)) pour l'analyse de données et la gestion d'entrées et sorties. Ces types de fichiers sont souvent utilisés pour stocker de manière structurée des données provenant d'expériences scientifiques en tant qu'arbres et tableaux pour réduire l'espace de stockage utilisé par un fichier.

Ces différentes données ne sont pas utilisables directement et doivent être réinterprétées par le logiciel "pyeventio\_example", lui aussi fourni par l'équipe de l'[UNIGE](#). Ce projet met à disposition un exécutable nommé "runana". Celui-ci permet de créer un fichier binaire contenant des métadonnées, le signal de chaque capteur composant la caméra du télescope et les instants auxquels les photons provenant de la pluie Cherenkov ont impacté les capteurs.

## CHAPITRE 4 : RÉSULTATS

Ce chapitre porte sur les différents essais qui ont été effectués et les résultats de ceux-ci. Le projet ayant été adapté en cours de route, les sous-chapitres sont organisés de manière chronologique.

### 4.1. ADAPTATION DU SIMULATEUR PE EXTRACTOR

La première étape du projet a été de maîtriser et formatter les données du premier simulateur "pe\_extractor". Comme évoqué lors du chapitre précédent, les bacs contenant les photons détectés sont générés à une fréquence plus élevée que celle du signal. J'ai donc commencé par regrouper les différents bacs générés sur la même période que le signal :

```

1 # generator variables
2 n_sample = 200000
3 n_sample_init = 0
4 batch_size = 1
5 shift_proba_bin = 30
6 sigma_smooth_pe_ns = 0
7 bin_size_ns = 0.20
8 sampling_rate_mhz = 1000 #200 MHz is the sampling rate of Terzina
                           #1000 LST + MST
9
10 pe_rate_mhz = 150 # 1 to 150 MHz
11 noise_lsb=4. # 3.5 to 5.5
12 amplitude_gain=16.
13 relative_gain_std=0.05
14
15 # computed variables
16 sampling_period_s = 1 / (sampling_rate_mhz * 1e6)
17 bin_per_sample = ceil(((sampling_period_s) * 1e9) / bin_size_ns)
18
19 # parameters omitted for brevity
20 gen = generator_nsb(...)
21 data = next(gen)
22
23 # data contains in index 0 the configurable batches of the discrete signal
24 # and in index 1 the batches of truth
25 # here we configured only 1 batch, which explains the [1][0]
26 summed_bins = np.sum(data[1][0].reshape(-1, bin_per_sample), axis=1)

```

LISTING 4.1 – Regroupement des bacs de photons-électrons, signal\_sense.ipynb

Ce qui nous donne le couple de données suivant :

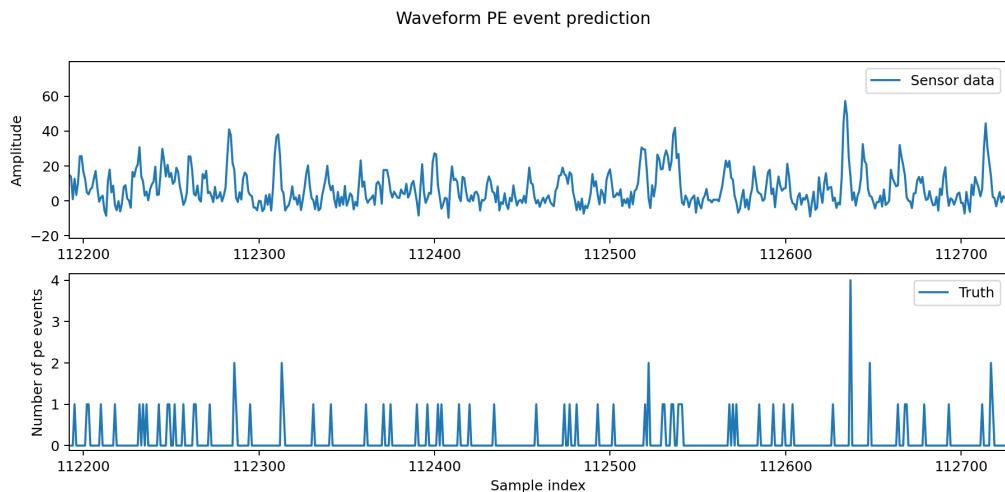


ILLUSTRATION 4.1 – Exemple de données générées par "pe\_extractor".

Les configurations des paramètres du générateur sont restées majoritairement les mêmes au cours du projet ; seuls ceux commentés ont évolués au cours du projet en fonction du télescope visé. Il en va de même pour le fichier d’impulsion correspondant au télescope.

## a. Création d'un dataset

Jusqu'à présent, les données produites sont deux signaux de 200'000 échantillons. L'un contient l'amplitude électrique du capteur et l'autre le compte de pe pendant la période d'échantillonnage. Cependant, il ne serait pas facile d'entraîner un réseau sur ce genre de données car il devrait avoir 200'000 neurones d'entrée. Il est nécessaire de découper ces données générées en plusieurs morceaux pour que le modèle puisse s'entraîner sur chaque cas individuellement. Pour cela, on peut utiliser une technique connue appelée "sliding window" ou fenêtre coulissante en français. Cela va diviser notre signal de 200'000 échantillons en plusieurs parties, comme si l'on regardait ces données à travers une fenêtre que l'on déplace d'un bout à l'autre :

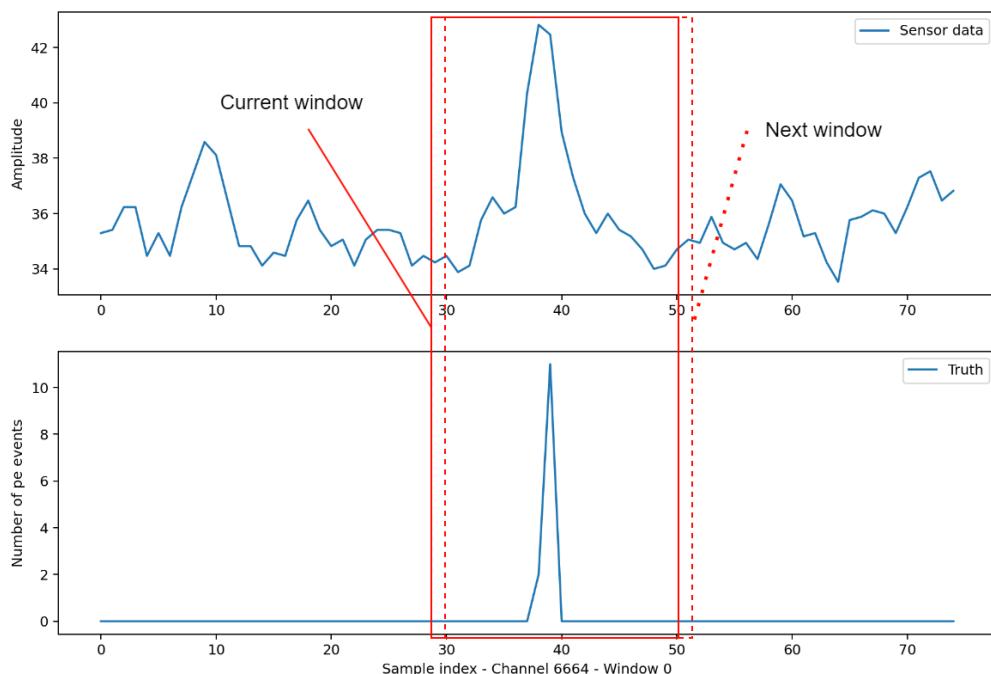


ILLUSTRATION 4.2 – Exemple de séparation des données par fenêtre coulissante.

Cette transformation étant déjà implémentée dans la librairie Numpy, je l'ai utilisée sur les deux signaux en même temps afin de ne pas perdre la vérité générée par le simulateur.

À cet instant du projet, la plus grosse contrainte du réseau de neurones est qu'il doit fonctionner en temps réel, avec le moins d'utilisation de ressources, à bord de Terzina. C'est

pour cela qu'au début des tests, plusieurs tailles de fenêtres ont été testées, 7, 11, 21 jusqu'à un maximum de 51 ; au delà, cela était jugé excessif. De plus, la conception des ASIC prévoyait de ne numériser qu'environ une vingtaine d'échantillons lors d'un déclenchement.

## 4.2. PREMIERS TESTS DE RÉSEAUX NEURONAUX

### a. Détection booléenne de photon-électron

Comme premier réseau neuronal, j'ai essayé de détecter la présence ou non d'au moins un pe exactement au centre de chaque fenêtre.

Le résultat attendu de ce réseau est une seule estimation en sortie de la présence ou non d'un photon à l'échantillon central. Pour cela, les fenêtres de vérité ont été converties en un tableau de taille 1 avec la valeur 1 ou 0 dépendant de la présence d'au moins un pe. J'ai ensuite entraîné le réseau neuronal suivant sur ces données :

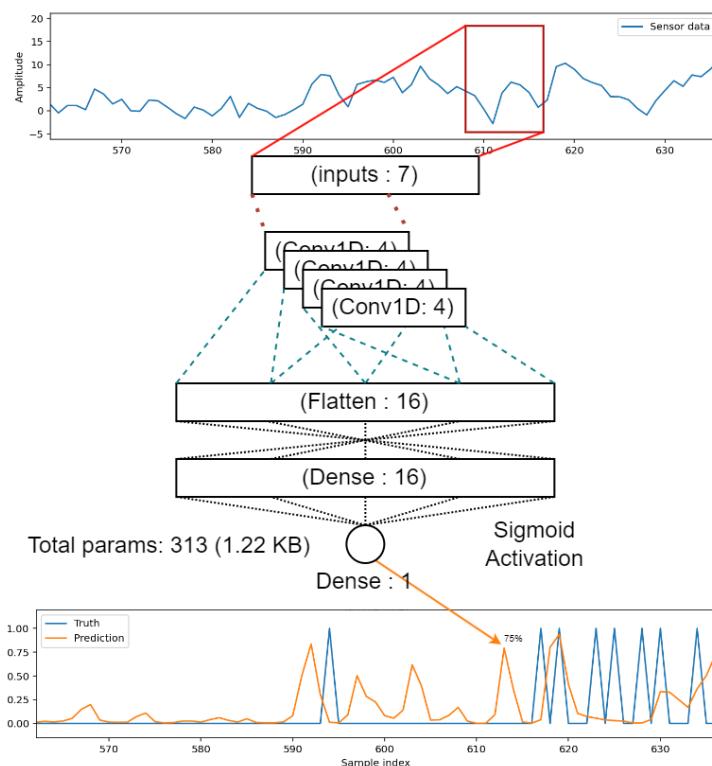


ILLUSTRATION 4.3 – Diagramme d'architecture du premier CNN à classification simple.

Les résultats observés provenant de cette architecture n'étaient pas bons. La raison principale de ces mauvais résultats est l'un des pièges les plus communs du machine learning : de mauvaises données. Cette première erreur provenait de la transformation supplémentaire

pour créer des valeurs attendues booléennes. Lors de cette transformation, une simple erreur d'indice avait récupéré le premier échantillon de la fenêtre au lieu de l'échantillon central. Cela a donc faussé toutes les données sur lesquelles le réseau s'était entraîné.

Cette première expérience m'a poussé à développer des vues au "cas par cas" où chaque inférence du modèle peut être examinée en détail. Cette vue affiche les données d'entrée, la vérité attendue et le résultat du modèle, ce qui permet de visualiser son comportement après entraînement.

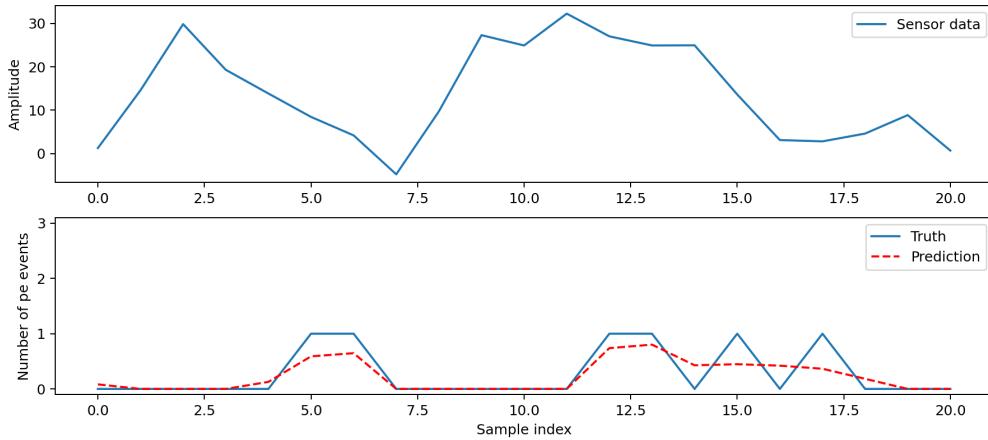


ILLUSTRATION 4.4 – Exemple de visualisation d'une inférence d'un réseau.

Cependant, même après avoir corrigé le dataset d'entraînement, les résultats -bien que légèrement améliorés- restaient non fiables à cause de nombreux faux positifs et faux négatifs.

## b. Estimation calorimétrique de photon-électron par régression

Pour avoir un meilleur point de décision qu'une simple valeur booléenne à bord de Terzina et pour améliorer les performances du premier réseau, j'ai adapté celui-ci pour avoir une fenêtre de sortie estimant une quantité de photons présents à chaque échantillon.

En premier, il faut modifier la couche de sortie pour qu'elle ait le même nombre de neurones que la couche d'entrée. En plus de cela, il faut changer la fonction d'activation utilisée par les neurones de sortie. Par défaut, la fonction utilisée est la sigmoïde. Celle-ci ne retourne que des valeurs dans l'intervalle  $[0, 1]$ , tandis que la fonction "ReLU" retourne des valeurs dans l'intervalle  $[0, +\infty[$ , ce qui est parfait pour compter des pe.

Pour l'entraînement de ce réseau, il a été choisi d'utiliser la fonction d'erreur "MeanSquaredError" et l'optimiseur Adam, car ce sont ceux qui, après quelques tests manuels,

présentaient les meilleures performances d'entraînement. Le dataset d'entraînement comportait 160'000 fenêtres et 40'000 réservées à la validation (un répartition 80/20), ces fenêtres étaient générées de manière aléatoire mais avec les mêmes configurations.

Ce nouveau modèle est donc capable d'estimer le nombre de pe pour chaque échantillon. Celui-ci a immédiatement mieux performé que le précédent :

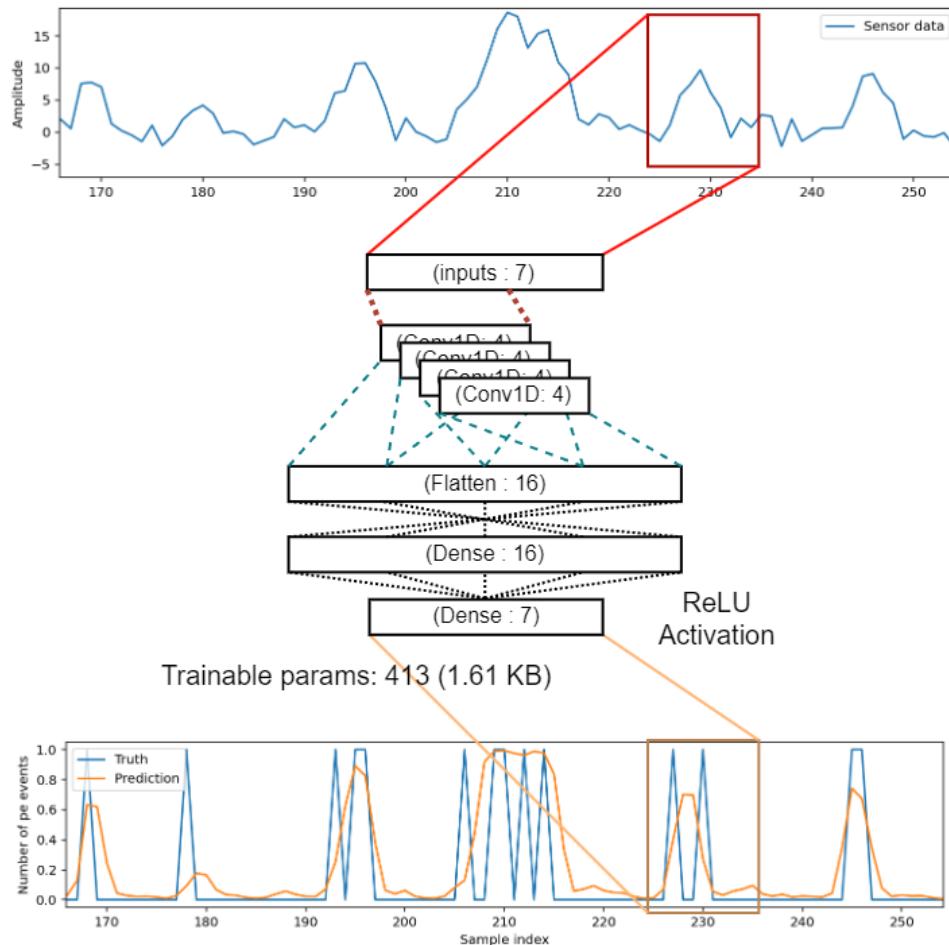


ILLUSTRATION 4.5 – Diagramme d'architecture du CNN à multiples sorties.

### c. Métriques de performance des réseaux

Arrivé à cette étape, il est devenu nécessaire de quantifier les écarts de performances entre chaque architecture testée. Pour des modèles effectuant de la classification, chaque prédiction peut être comptabilisée dans une matrice de confusion comme celle-ci :

		Predicted condition	
		Positive (PP)	Negative (PN)
Actual condition	Total population = P + N	Positive (P)	True positive (TP) False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

ILLUSTRATION 4.6 – Matrice de confusion. Source : [Wik19]

Dans notre cas, nous n'effectuons pas une classification mais une régression. Pour pouvoir remplir une matrice de confusion, il est nécessaire de définir ce qu'est un vrai positif, un faux positif, etc. J'ai donc défini que, si l'estimation du modèle dépasse une valeur de 0.5, cette estimation est considérée comme positive. Ensuite, si la vérité contient effectivement un `pe` ou plus à cette position, cette estimation est considérée comme vrai positif ; à l'inverse, si aucun `pe` n'est présent, alors cela est considéré comme un faux positif.

Les autres calculs de performances qui ont été retenus pour ce travail sont les suivants :

1. Précision : La précision représente le nombre de vrais positifs parmi tous les positifs identifiés par le modèle. Une précision de 1 signifie que tous les éléments positifs identifiés par le modèle l'étaient bien, alors que zéro indique l'inverse.
2. Rappel ou Recall : Le rappel est calculé en divisant le nombre de vrais positifs par le nombre de positifs qui existent dans les données. Un rappel de 1 veut dire qu'aucun positif n'a été manqué alors que 0 veut dire qu'aucun n'a été identifié.
3. Exactitude ou Accuracy : L'exactitude mesure le nombre de vrai résultats positifs ou négatifs.

Les implémentations de ces fonctions ont été faites pour fonctionner directement avec le framework Tensorflow pour générer automatiquement des logs lors de l'entraînement des modèles.

Avec ces métriques, le dernier modèle CNN affiché est comparé avec le meilleur CNN trouvé :



ILLUSTRATION 4.7 – Différence de métriques entre deux CNN (en rose le premier CNN à régression et en vert le meilleur CNN testé).

Le meilleur CNN trouvé travaille sur une fenêtre de 21 échantillons au lieu de 7, contient une couche de BatchNormalisation, effectue des convolutions sur 11 échantillons au lieu de 4 et a une couche dense de 32 neurones au lieu de 16.

Cependant, on peut constater que même ce CNN peine à performer avec une précision de seulement 30% et un Recall de 70%.

## d. Entraînement avec Sample Weights

Lors des tests avec différentes architectures de CNN, un problème plus subtil a été trouvé. Lors de l’entraînement d’un réseau, il est nécessaire d’avoir un dataset d’entraînement uniforme sur les différents cas qu’il verra lors de son utilisation. Par exemple, si un réseau classifiant des habits n’est entraîné qu’avec un dataset contenant des chaussures à 99% et que le 1% restant se rapporte à d’autres types d’habits, celui-ci aura alors du mal à apprendre de ce 1%.

Dans des cas moins graves qu'une répartition 99/1, comme par exemple : une répartition 60/40, ce genre de disparités des données peut être mitigé en utilisant des poids pour chaque cycle d'apprentissage. Les données apparaissant le plus souvent se verront attribuer un poids "léger" lors de l'apprentissage, car chaque rencontre avec une donnée appartenant aux 60% est moins importante qu'une rencontre avec des données composant les 40% restants, qui se voient attribuer un poids plus "lourd".

Dans notre cas, cela s'est représenté sous la forme des différentes valeurs des `pe` détectés. Le nombre d'échantillons ne contenant aucun `pe` était parfois en surabondance de plus de 3 fois comparé aux autres valeurs. J'ai alors créé une fonction calculant l'importance d'une fenêtre en fonction du nombre total de `pe` qui y sont présents.

Malheureusement ces nouveaux poids ont eux l'effet inverse lors de l'entraînement des réseaux :

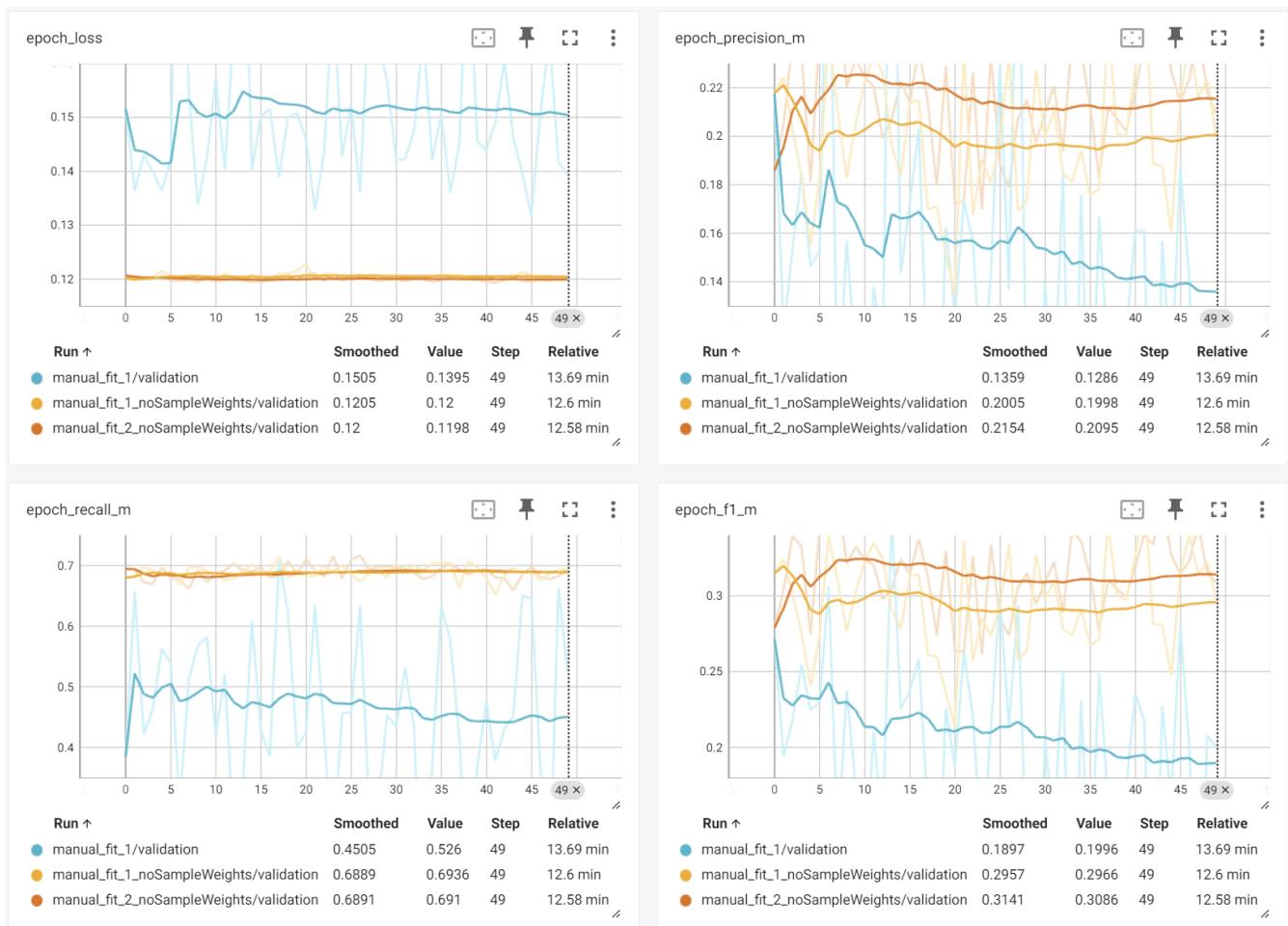


ILLUSTRATION 4.8 – Perte de performance en entraînant avec des Sample Weights (en orange/jaune deux réseaux CNN entraînés sans Sample Weights, en bleu un modèle entraîné avec Sample Weights).

J'estime que cela est dû à l'application des Sample Weights sur une fenêtre complète. Une fenêtre peut contenir une grande proportion d'échantillons où aucun `pe` n'est présent. Cela signifie que tous ces résultats sont quand même majorés par rapport au Sample Weight de la fenêtre et cela fausse donc l'apprentissage du réseau.

### e. Tests d'architecture RNN

Pour essayer d'augmenter la précision ainsi que le Recall des derniers modèles testés, j'ai créé une architecture de `RNN` simple ressemblant à ceci :

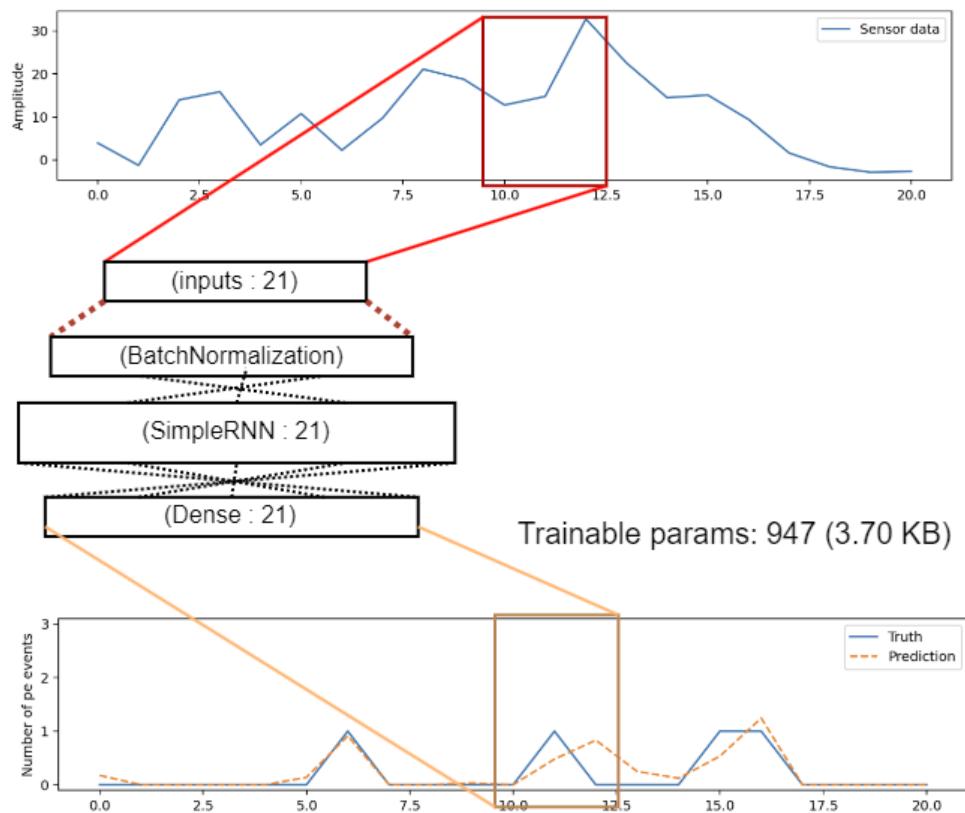


ILLUSTRATION 4.9 – Diagramme d'architecture du RNN simple.

Dès les premiers cycles d'entraînement, ce modèle `RNN` s'est directement départagé des autres en achevant très rapidement les meilleures performances sur les données simulées par "pe\_extractor" avec une précision de presque 60% et un Recall de 70% :

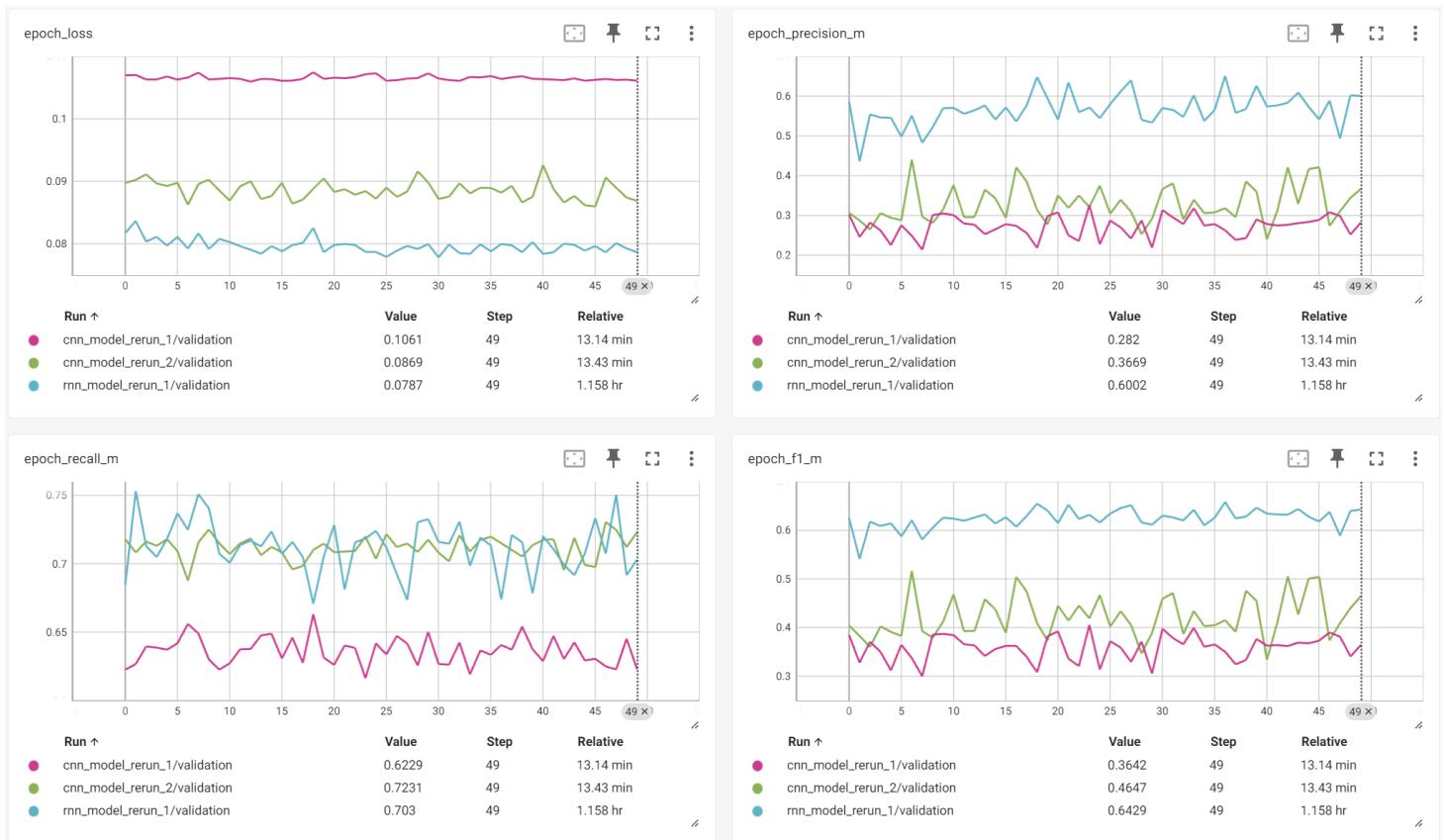


ILLUSTRATION 4.10 – Comparaison des métriques entre CNN et RNN (en rose et vert les CNN et en bleu le RNN simple).

### 4.3. FICHIERS DE SIMULATION CORSIKA

Après avoir testé de nombreuses autres configurations sur les données du précédent simulateur, j'ai pensé que les différents plateaux atteints lors des entraînements pouvaient être causés par les données elles-même.

J'ai pensé qu'en utilisant des données provenant de simulations plus complexes et avec des vraies pluies Cherenkov, cela pourrait aider les réseaux à y trouver des motifs qui n'existaient pas dans l'ancienne simulation.

L'équipe de l'**UNIGE** m'a alors fourni des fichiers de simulations pour la prochaine caméra du **LST**. Pour pouvoir utiliser ces fichiers, il a d'abord fallu recréer les signaux avec l'outil complémentaire : "pyeventio\_example". Sur le cluster Yggdrasil, j'ai compilé et utilisé l'exécutable "runana" de manière distribuée grâce à un script bash :

```

1 #!/bin/sh
2
3 # manage cluster modules
4 module purge

```

```

5 module load GCC/11.2.0 OpenMPI/4.1.1 ROOT/6.24.06
6
7 # select events range to generate
8 ev_start=0
9 ev_stop=1000
10
11 # select the simulation file
12 inRootFiles="/srv/beegfs/scratch/shares/heller/Leonid/mono-lst-sipm-pmma-3ns-v1_triggerless/
   gamma/0000/corsika_0000ID.root"
13 rndseed="1312312"
14
15 for ((evID=ev_start; evID<=ev_stop; evID++))
16 do
17     # location of output file
18     binFileOut="/home/users/p/perrinya/scratch/bin_data/gamma_ev"$evID"_out.bin"
19     # recreate complete waveform with truths
20     srun --time "00:00:30" /home/users/p/perrinya/pyeventio-example/runana 333 "
   $inRootFiles" "$evID" "$binFileOut" "$rndseed" &
21 done

```

LISTING 4.2 – Script de génération des signaux à partir de fichier de simulations, data/slurm-run.sh

Les fichiers binaires créés utilisent bien plus de place que les fichiers ROOT. Par exemple, le fichier "corsika\_0000ID.root", comportant les informations nécessaires à reconstruire environ 500'000 pluies Cherenkov d'une durée d'environ 75ns chacune, ne pèse que 1.2GB. En comparaison, les 1'000 premières pluies sous forme de fichiers binaires prennent plus de 2.38GB de stockage. Si l'on venait à stocker la totalité des événements d'un seul fichier ROOT en fichiers binaires, ceux-ci prendraient une taille de :  $500'000/1000 * 2.38GB = 1.19TB$

Cependant, la création de ces fichiers n'est que la première étape pour créer un jeu de données utilisable par nos réseaux de neurones. Les fichiers produits sont structurés de la manière suivante :

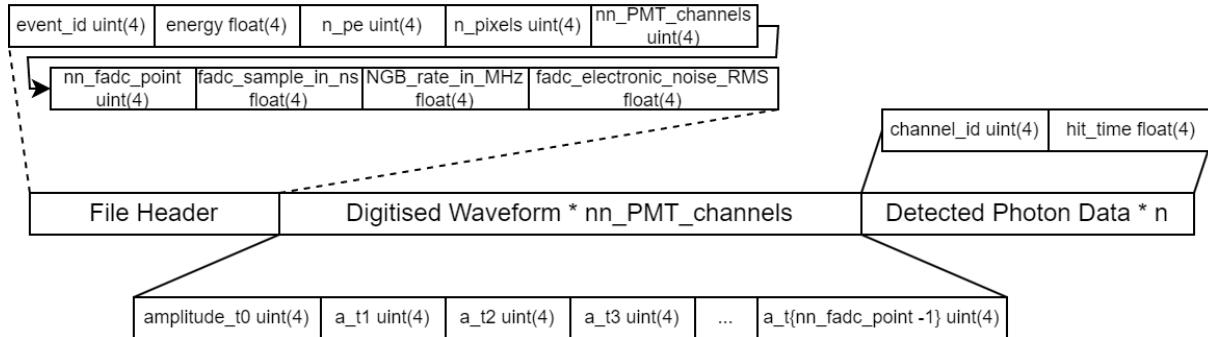


ILLUSTRATION 4.11 – Structure des fichiers binaires de simulations Corsika.

J'ai donc dû créer un lecteur capable de parser ces fichiers binaires et de recréer une vérité sur la même fréquence que le signal d'entrée. Pour cela, la lecture du header et du signal se fait simplement en lisant et stockant des variables dans des tableaux en mémoire. Cependant, quelques calculs et vérifications doivent être effectués pour les photons détectés par la caméra. En premier, il faut convertir l'instant de l'impact en numéro d'échantillon en divisant ce temps par la période d'acquisition connue dans l'entête du fichier. Puis, il faut vérifier si ce photon a bien été capturé pendant la durée du signal, car celui-ci peut avoir été détecté avant ou après l'événement, son temps d'impact peut donc être négatif ou supérieur à 75ns, qui correspond la fin de la capture du signal.

Au final, les données récupérées dans ces fichiers binaires sont similaires au premier simulateur :

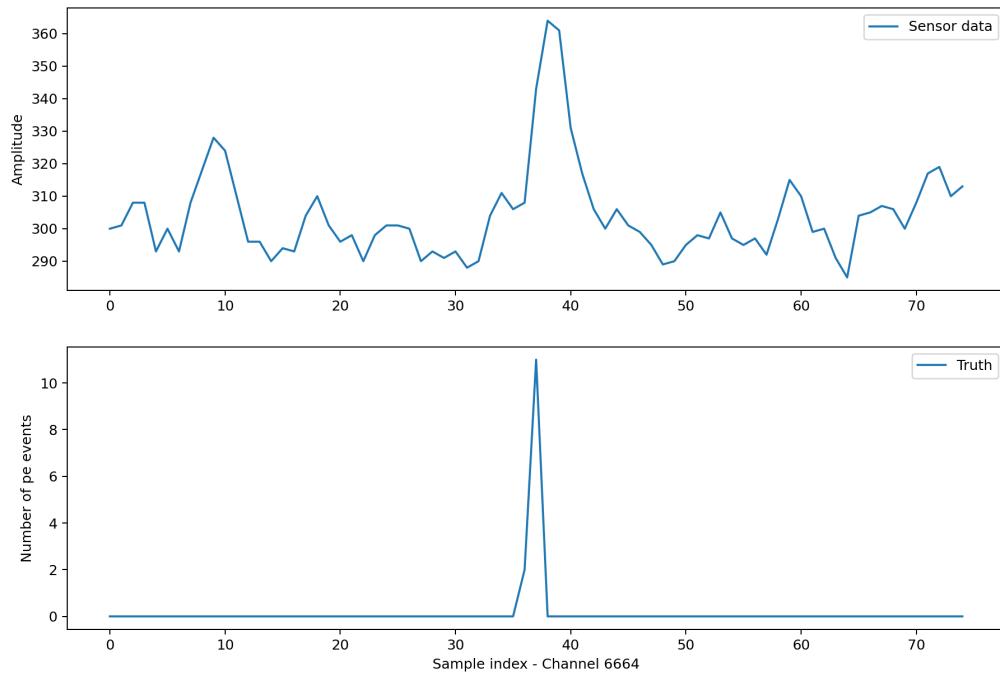


ILLUSTRATION 4.12 – Exemple de données générées par simulations Corsika.

Cependant, même dans cet état, les données ne sont pas encore utilisables. Lorsque l'on regarde l'amplitude du signal contenu dans ces fichiers binaires, ceux-ci sont encodés sur des entiers. Pour réduire ces valeurs avant d'essayer d'entraîner des réseaux neuronaux, il m'a été conseillé de normaliser ces données en les divisant par 8,5 en rapport avec les capteurs simulés.

De plus, nous avons découvert que les temps d'impacts des photons détectés étaient un peu en avance par rapport au signal. Nous avons donc ajouté un délai de 2 échantillons lors du parsing du fichier, pour que les pics de vérité arrivent en même temps ou avec un léger retard pour que le réseau de neurones puisse utiliser les échantillons précédents comme information utile.

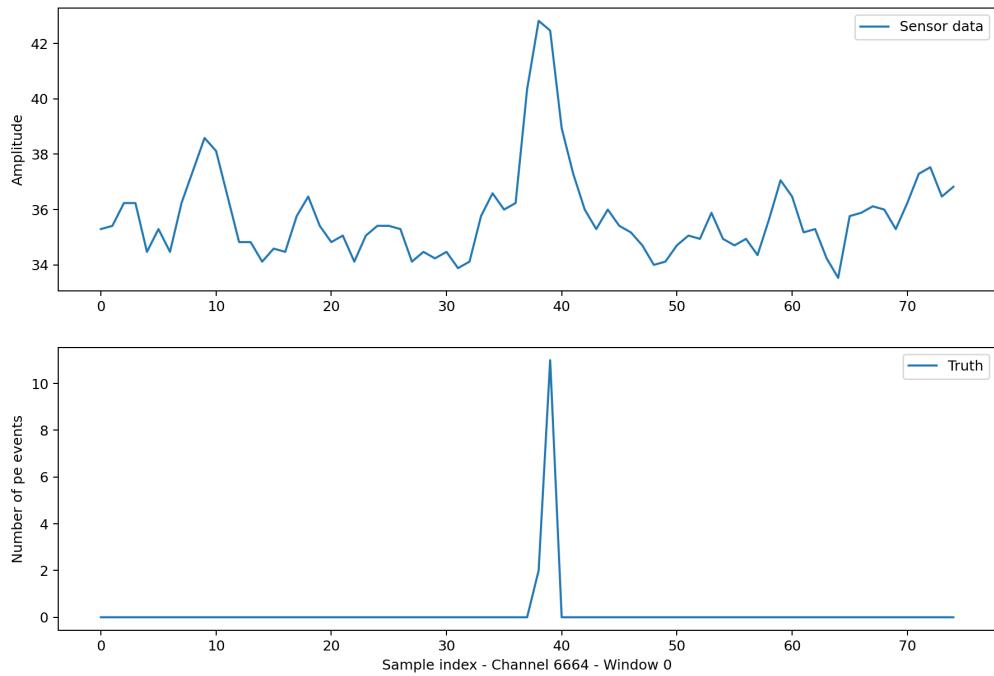


ILLUSTRATION 4.13 – Données Corsika normalisées.

## a. Création d'un dataset

Ici aussi, comme pour le simulateur "pe\_extractor", il a été nécessaire de créer un dataset à partir des données brutes. Cependant, pour ces fichiers de simulations, les signaux reconstitués à partir des fichiers binaires ne forment pas un seul long signal mais 7'855 signaux de 75ns chacun. Il est donc nécessaire de prendre chaque signal à part individuellement pour créer des fenêtres capables d'entraîner les réseaux de neurones.

Lors de la création de ce nouveau dataset, un problème d'uniformité encore plus grave que le simulateur "pe\_extractor" a été constatée. Les pluies Cherenkov sont des événements très brefs et peu énergétiques. Cela est d'autant plus vrai pour le **LST** qui est le télescope le plus sensible de la nouvelle génération.

Tous les signaux concaténés d'une pluie Cherenkov ressemblent à ceci :

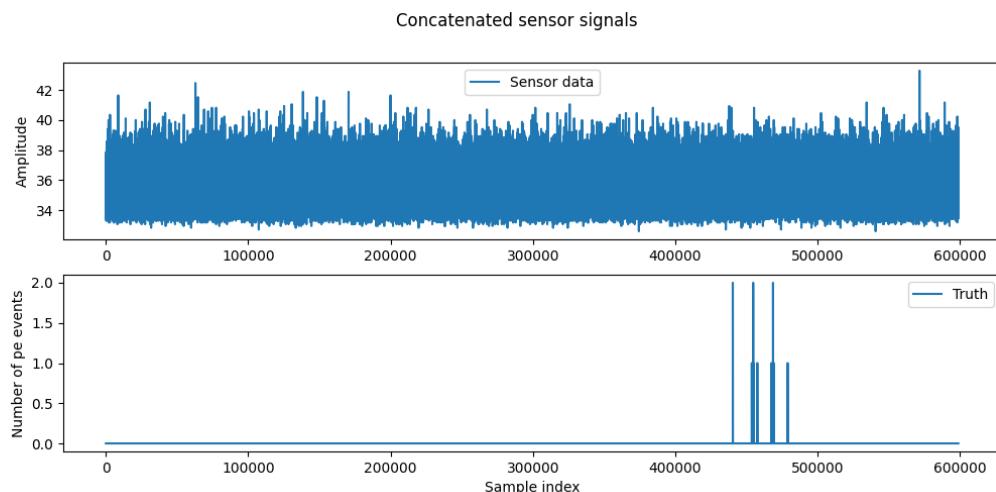


ILLUSTRATION 4.14 – Signaux concaténés de chaque capteur d'une pluie Cherenkov.

Pour éviter ce problème, alors que les Sample Weights n'avaient pas l'air de donner de bons résultats précédemment, j'ai ajouté, lors de la création d'un dataset, une limite sur le nombre de fenêtres contenant un même nombre de pe. Afin d'éviter que ce ne soit toujours les  $x$  premières fenêtres d'un événement qui soient sélectionnées, chaque fenêtre est choisie de manière aléatoire. Avec cette limitation, le dataset généré est beaucoup plus uniforme :

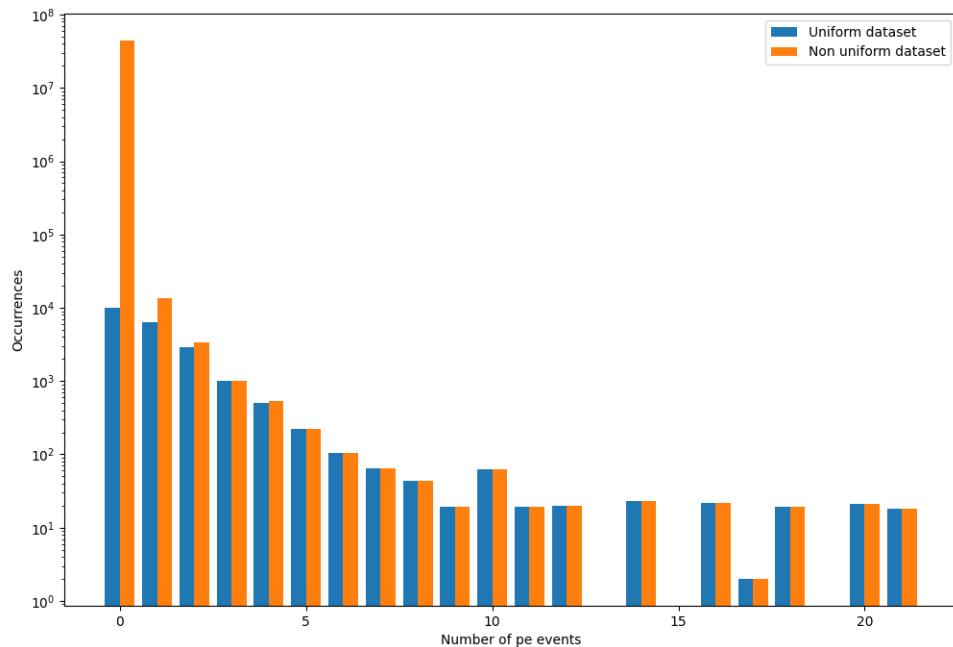


ILLUSTRATION 4.15 – Comparaison entre un dataset uniformisé et non uniformisé générés sur 100 pluies Cherenkov.

Cela réduit la taille du dataset considérablement, ce qui permet de le stocker et de le garder en mémoire vive plus facilement. Le dataset créé pour les entraînements contient lui les 1'000 premières pluies simulées du fichier Corsika :

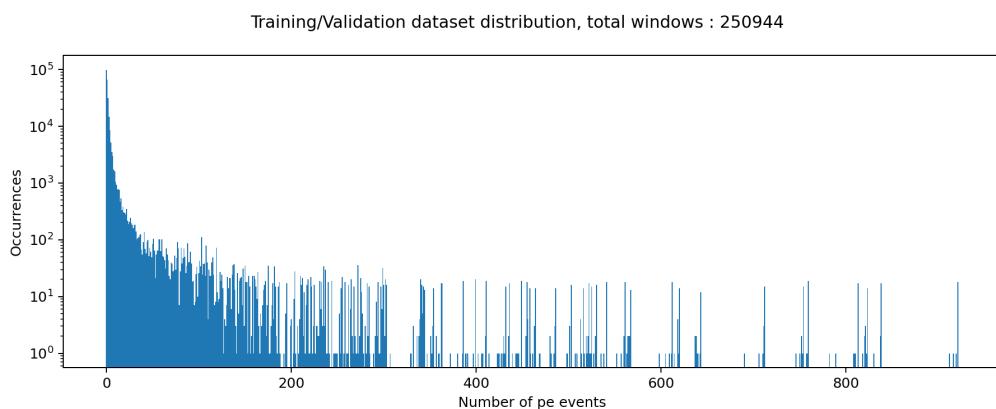


ILLUSTRATION 4.16 – Distribution du dataset généré pour les entraînements.

Pour éviter de devoir à chaque fois recharger et traiter ces 1'000 événements, j'ai aussi créé un système de sauvegarde et chargement d'un dataset.

## b. Entraînement des réseaux de neurones

Avec ce nouveau dataset, j'ai repris les différentes architectures de réseaux qui avaient donné les meilleurs résultats sur l'ancien simulateur. J'ai ensuite relancé des entraînements de zéro et ai observé les résultats suivants :



ILLUSTRATION 4.17 – Métriques d'entraînement des modèles sur les données Corsika.

Les résultats de ces entraînements ont régressés par rapport au simulateur "pe\_extractor". Après analyse, je pense que cela est dû aux données du simulateur et aux différentes limitations physiques des capteurs. Lorsque l'on regarde de près certaines fenêtres du dataset, on constate que les petites quantités de `pe` présentes dans le signal ne participent que très peu à l'amplitude de celui-ci, rendant l'extraction de l'information difficile.

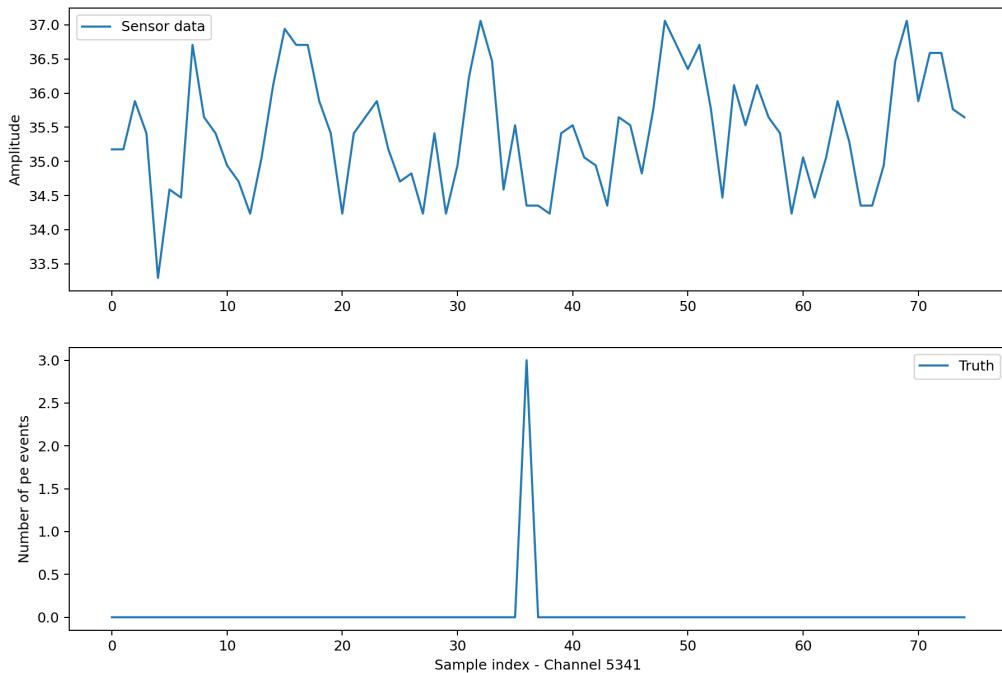


ILLUSTRATION 4.18 – Exemple de données Corsika difficilement discernables.

Ici, le pic d'amplitude produit par la présence des 3 pe présents dans le signal est difficilement discernable du bruit alentour. La forme du pic pourrait être discernable par un réseau de neurones mais la faible résolution temporelle disponible dans ces échantillons me fait douter de la possibilité d'en tirer des informations concluantes sur des petites quantités de pe.

#### 4.4. AMÉLIORATIONS POSSIBLES

C'est malheureusement à ce point que je n'ai pas réussi à explorer de techniques supplémentaires pour pallier aux problèmes rencontrés lors de l'utilisation de données plus proches des capteurs réels. Cependant, il me reste encore une idée qui pourrait possiblement réduire les problématiques rencontrées.

Lors de pluies atmosphériques Cherenkov, plusieurs capteurs aux alentours sont souvent aussi touchés. Le bruit électronique de chaque capteur est majoritairement propre à lui-même. Même si une partie de l'électronique produit du "CrossTalk" qui est du bruit électrique se propageant autour de chaque pièce matérielle, celui-ci pourrait être amoindri en analysant non plus un pixel seul mais un groupe de pixels voisins.

## CONCLUSION

Étant passionné par l’astronautique et intéressé par tous les sujets adjacents tels que l’astrophysique ou même l’aéronautique, j’ai été ravi d’être choisi pour participer à un projet concret touchant à ces mêmes domaines.

Pour rappel, l’objectif de ce projet de bachelor consistait à trouver un modèle de réseau neuronal capable d’interpréter le contenu de signaux produits par des télescopes afin d’identifier les possibles photons provenant de pluies atmosphériques Cherenkov. Le modèle trouvé aurait servi à deux usages : le premier à réduire le stockage et le traitement des données en participant au tri d’événements scientifiquement intéressants, le deuxième à possiblement améliorer les performances d’analyse de plus haut niveau en filtrant le bruit contenu dans les signaux.

Pour réaliser ce projet, j’ai d’abord dû appréhender le phénomène physique de la radiation Cherenkov et ses implications pour le domaine de l’astrophysique. Il m’a fallu aussi comprendre le fonctionnement des différents télescopes existants et futurs afin de discerner ce qui était attendu de ce projet. J’y ai appliqué les bases en machine learning que j’ai appris lors de mon cursus, et ai réussi à éviter plusieurs pièges du domaine. J’ai aussi dû prendre en main plusieurs outils simulant des données scientifiques et les adapter pour entraîner des modèles de machine learning. Malgré les diverses techniques de machine learning testées, aucun des modèles n’a présenté des performances permettant à celui-ci d’être intégré au sein d’autres projets comme imaginé.

Cependant, la détection de *pe* d’un signal n’est pas encore à considérer comme impossible. Il reste encore des techniques de machine learning plus complexes à investiguer. De plus et comme évoqué au chapitre précédent, si mon soupçon que les données ne contiennent pas assez de détails s’avérait exact, il est envisageable d’observer les données d’une manière différente pour minimiser les effets de bruits. Par exemple, en observant des groupes de pixels voisins. Il se pourrait aussi qu’il est impossible de discerner la présence de petites quantités de *pe* dans des pluies de très basse énergie.

Dans l’ensemble, je suis satisfait du travail que j’ai pu fournir au cours de ces derniers mois. Grâce à cette expérience, j’ai pu améliorer mes connaissances en machine learning et en recherches scientifiques. Dans un monde se tournant de plus en plus vers les intelligences artificielles et le machine learning, il est certain que les apprentissages et les conclusions que

j'ai acquis au cours de ce travail me seront d'une grande utilité durant toute ma carrière professionnelle.

## ANNEXES

### ANNEXE 1

Code source et fichiers de travail disponibles sur le repository git :

<https://github.com/TheCakeIsNotALie/signal-sense>

## BIBLIOGRAPHIE

- [AA23] National AERONAUTICS et Space ADMINISTRATION. *Fermi Gamma-ray Space Telescope*. <https://science.nasa.gov/mission/fermi>. Accessed : (15.04.2024). 2023.
- [AA24] National AERONAUTICS et Space ADMINISTRATION. *Fermi Gamma-ray Space Telescope*. <https://fermi.gsfc.nasa.gov/>. Accessed : (15.04.2024). 2024.
- [Age02a] European Space AGENCY. *Integral overview*. [https://www.esa.int/Science\\_Exploration/Space\\_Science/Integral\\_overview](https://www.esa.int/Science_Exploration/Space_Science/Integral_overview). Accessed : (14.04.2024). 2002.
- [Age02b] European Space AGENCY. *Integral Rendition*. [https://www.esa.int/ESA\\_Multimedia/Images/2001/11/Integral](https://www.esa.int/ESA_Multimedia/Images/2001/11/Integral). Accessed : (14.04.2024). 2002.
- [Age20a] European Space AGENCY. *Low Earth orbit*. [https://www.esa.int/ESA\\_Multimedia/Images/2020/03/Low\\_Earth\\_orbit](https://www.esa.int/ESA_Multimedia/Images/2020/03/Low_Earth_orbit). Accessed : (15.04.2024). 2020.
- [Age20b] European Space AGENCY. *Polar and Sun-synchronous orbit*. [https://www.esa.int/ESA\\_Multimedia/Images/2020/03/Polar\\_and\\_Sun-synchronous\\_orbit](https://www.esa.int/ESA_Multimedia/Images/2020/03/Polar_and_Sun-synchronous_orbit). Accessed : (15.04.2024). 2020.
- [All22] Lorella Angelini Jesse ALLEN. *Integral Specifications*. <https://heasarc.gsfc.nasa.gov/docs/integral/integral.html>. Accessed : (14.04.2024). 2022.
- [Aph15] Wikimedia Commons APHEX34. *Convolutional layer*. [https://commons.wikimedia.org/w/index.php?title=File:Conv\\_layer.png&oldid=867907596](https://commons.wikimedia.org/w/index.php?title=File:Conv_layer.png&oldid=867907596). Accessed : (19.04.2024). 2015.
- [Ban23] Mayank BANOULA. *What is Perceptron : A Beginners Guide for Perceptron*. <https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron>. Accessed : (18.08.2024). 2023.
- [Ber24] Konrad BERNLÖHR. *Atmospheric Cherenkov light*. <https://www.mpi-hd.mpg.de/hfm/CosmicRay/ChLight/Cherenkov.html>. Accessed : (21.02.2024). 2024.

- [Bur23] Leonid BURMISTROV. “Terzina on board NUSES : A pathfinder for EAS Cherenkov Light Detection from space”. In : *EPJ Web of Conferences* 283 (2023). Sous la dir. d'I. DE MITRI et al., p. 06006. ISSN : 2100-014X. DOI : 10.1051/epjconf/202328306006. URL : <http://dx.doi.org/10.1051/epjconf/202328306006>.
- [Col04] The HAWC COLLABORATION. *The High-Altitude Water Cherenkov Gamma-Ray Observatory*. <https://www.hawc-observatory.org>. Accessed : (16.04.2024). 2004.
- [CTA17] IAC CTAO Gabriel Pérez Díaz. *CTAO Northern Hemisphere Array Rendering*. [https://www.flickr.com/photos/cta\\_observatory/32835056736/](https://www.flickr.com/photos/cta_observatory/32835056736/). Accessed : (18.04.2024). 2017.
- [CTA22] CTAO. *5 Concepts You Should Know About Gradient Descent and Cost Function*. [https://indico.cern.ch/event/1096028/contributions/4761499/attachments/2404761/4113415/CTA-GDB\\_2022.pdf](https://indico.cern.ch/event/1096028/contributions/4761499/attachments/2404761/4113415/CTA-GDB_2022.pdf). Accessed : (18.08.2024). 2022.
- [CTA24] CTAO. *LST Specifications*. [https://www.ctao.org/wp-content/uploads/Specifications\\_LST\\_2024.pdf](https://www.ctao.org/wp-content/uploads/Specifications_LST_2024.pdf). Accessed : (16.08.2024). 2024.
- [CTAnd] CTAO. *Cherenkov Telescope Array Observatory*. <https://www.ctao.org/>. Accessed : (18.04.2024). n.d.
- [Dav23] IBM DAVE BERGMANN Cole Stryker. “What is an autoencoder ?” In : (2023). Accessed : (19.04.2024).
- [fde17] Wikimedia Commons FDELOCHE. *Recurrent neural network unfold*. [https://commons.wikimedia.org/w/index.php?title=File:Recurrent\\_neural\\_network\\_unfold.svg&oldid=854998628](https://commons.wikimedia.org/w/index.php?title=File:Recurrent_neural_network_unfold.svg&oldid=854998628). Accessed : (19.04.2024). 2017.
- [Gronda] MAGIC GROUP. *The MAGIC Telescopes*. <https://magic.mpp.mpg.de/>. Accessed : (16.04.2024). n.d.
- [Grondb] MAGIC GROUP. *The MAGIC Telescopes Photo*. [https://magic.mpp.mpg.de/gallery/pictures/tn/IMG\\_2520.JPG.html](https://magic.mpp.mpg.de/gallery/pictures/tn/IMG_2520.JPG.html). Accessed : (16.04.2024). n.d.
- [Hof12] Werner HOFMANN. *The H.E.S.S. Telescopes*. <https://www.mpi-hd.mpg.de/>. Accessed : (16.04.2024). 2012.

- [Kle12] H.E.S.S. collaboration KLEPSER DESY. *The H.E.S.S. II five-telescope gamma-ray experiment in Namibia.* [https://commons.wikimedia.org/wiki/File:HESS\\_II\\_gamma\\_ray\\_experiment\\_five\\_telescope\\_array.jpg](https://commons.wikimedia.org/wiki/File:HESS_II_gamma_ray_experiment_five_telescope_array.jpg). Accessed : (16.04.2024). 2012.
- [Lab09] Argonne National LABORATORY. *Advanced Test Reactor core, Idaho National Laboratory.* [www.flickr.com/photos/35734278@N05/3954062594/](http://www.flickr.com/photos/35734278@N05/3954062594/). Accessed : (10.04.2024). 2009.
- [Liu22] Clare LIU. *5 Concepts You Should Know About Gradient Descent and Cost Function.* <https://www.kdnuggets.com/2020/05/5-concepts-gradient-descent-cost-function.html>. Accessed : (18.08.2024). 2022.
- [Mas19] Michela MASSI. *Autoencoder schema.* [https://commons.wikimedia.org/w/index.php?title=File:Autoencoder\\_schema.png&oldid=808656740](https://commons.wikimedia.org/w/index.php?title=File:Autoencoder_schema.png&oldid=808656740). Accessed : (19.04.2024). 2019.
- [Mie23] Tjark MIENER. “Indirect Dark Matter Searches in the Gamma-ray Band and Development of New Analysis Techniques for Ground-based Gamma-ray Astronomy”. Thèse de doct. 2023.
- [Mie24a] Tjark MIENER. *Getting started with DL1DH+CTLearn.* [https://commons.wikimedia.org/w/index.php?title=File:Conv\\_layer.png&oldid=867907596](https://commons.wikimedia.org/w/index.php?title=File:Conv_layer.png&oldid=867907596). Accessed : (19.04.2024). 2024.
- [Mie24b] Tjark MIENER. *Getting started with DL1DH+CTLearn.* Support de présentation. 2024.
- [Min24] MIN-KYONG. *Pair Production and Annihilation.* <http://electrons.wikidot.com/pair-production-and-annihilation>. Accessed : (10.04.2024). 2024.
- [Ren20] Yves RENIER. *PE Extractor.* <https://github.com/cta-sst-1m/pe-extractor>. Accessed : (18.04.2024). 2020.
- [Rit23] F. Zanchetta RITA FIORESI. *Deep Learning and Geometric Deep Learning : an introduction for mathematicians and physicists - Scientific Figure on ResearchGate.* <https://www.researchgate.net/figure/Linearly-separable-and->

- linearly - non - separable - sets - githubborg \_ fig4\_370634232. Accessed : (16.08.2024). 2023.
- [Tri23] Caterina TRIMARELLI. *The NUSES space mission*. [https://indico.cern.ch/event/1199289/contributions/5446979/attachments/2703059/4692617/NUSES\\_TAUP.pdf](https://indico.cern.ch/event/1199289/contributions/5446979/attachments/2703059/4692617/NUSES_TAUP.pdf). Accessed : (14.04.2024). 2023.
- [Ver04] VERITAS. *Very Energetic Radiation Imagin Telescope Array System*. <https://veritas.sao.arizona.edu/>. Accessed : (16.04.2024). 2004.
- [Wik19] WIKIPÉDIA. *Confusion matrix*. [https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix). Accessed : (19.08.2024). 2019.
- [Wik23] l'encyclopédie libre. WIKIPÉDIA. *Rayonnement continu de freinage*. [http://fr.wikipedia.org/w/index.php?title=Rayonnement\\_continu\\_de\\_freinage&oldid=205544775](http://fr.wikipedia.org/w/index.php?title=Rayonnement_continu_de_freinage&oldid=205544775). Accessed : (10.04.2024). 2023.
- [Yan19] Muhamad YANI. *Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail*. [https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max\\_fig2\\_333593451](https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max_fig2_333593451). Accessed : (19.08.2024). 2019.
- [Zah15] Mohamed ZAHRAN. *ASSESSMENT OF ARTIFICIAL NEURAL NETWORK FOR BATHYMETRY ESTIMATION USING HIGH RESOLUTION SATELLITE IMAGERY IN SHALLOW LAKES : CASE STUDY EL BURULLUS LAKE*. - *Scientific Figure on ResearchGate*. [https://www.researchgate.net/figure/A-hypothetical-example-of-Multilayer-Perceptron-Network\\_fig4\\_303875065](https://www.researchgate.net/figure/A-hypothetical-example-of-Multilayer-Perceptron-Network_fig4_303875065). Accessed : (18.08.2024). 2015.