# Novel Methods for an established System: Moonlight Observations and Deep Learning Data Analysis with H.E.S.S.

**Neuartige Methoden für ein etabliertes System: Mondlicht Beobachtungen und Datenanalyse durch Maschinelles Lernen mit H.E.S.S**

Der Naturwissenschaftlichen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg
zur Erlangung des Doktorgrades Dr. rer. nat.

Vorgelegt von
Matthias Büchele
aus Nürnberg

Als Dissertation genehmigt von der Naturwissenschaftlichen Fakultät der Friedrich-Alexander-Universität Erlangen-Nürnberg

| | |
|---|---|
| Tag der mündlichen Prüfung: | 07. Mai 2021 |
| Vorsitzender des Promotionsorgans: | Prof. Dr. Wolfgang Achtziger |
| Gutachter: | Prof. Dr. Stefan Funk |
| | Prof. Dr. Ulrich Katz |

# Abstract

The High Energetic Stereoscopic System (H.E.S.S.) experiment has been successfully taking and analyzing gamma-ray data for more than a decade. It was designed to be at the technical frontier and since then many efforts were taken to keep that level of excellence. Nevertheless, there are still borders that are yet to be crossed. This work is focused on two of those constraints and provides the cornerstone for the system and its collaboration to not fall behind. The first one is the ability to extend the observation schedule by allowing safe operations of the telescopes during moonlight. Due to the detector system being a very sensitive and delicate system, additional light in the sky or the vicinity has always been a red flag. In the end, this is one of the main reasons why the site is located at an extremely remote place in the Khomas Highland at the heart of Namibia. The main science goal kept shifting from monitoring and scanning the gamma-ray night sky more and more to studying targets of opportunity in the last years, where the system reacts to an alert from another experiment when a source in the sky shows exceptional activity. Since those alerts might come in at any time, the question whether observations under moderate moonlight are possible became of interest. The developed model for the brightness of the night sky provides those answers together with different strategies to extend the system's duty cycle by an additional 60% to 88%. The second main part goes the other direction: Instead of taking more data, the possibility of analyzing more of the data already available using machine learning techniques is investigated. Many events are recorded and written to disc but due to strict quality selection cuts never end up in an analysis. With the modern available hardware it is possible to apply deep learning algorithms that are capable of analyzing data that otherwise would have been discarded. Results on the Active Galactic Nucleus PKS 2155-304 are presented, where the signal to noise ratio of the analysis could be improved from 11.6 to 12.4. In this sense, both covered topics share the same goal of further pushing the limits of this well established system, to stay at the edge of modern astroparticle research.

# Kurzfassung

Das *High Energetic Stereoscopic System* (H.E.S.S.) Experiment beobachtet und analysiert Gammastrahlung erfolgreich seit mehr als einem Jahrzehnt. Es wurde an der Grenze des technisch Möglichen konstruiert und viele Bemühungen wurden jeher unternommen, um diesen Grad an Exzellenz zu erhalten. Nichtsdestotrotz gibt es immer noch Bereiche in die man noch nicht vorgedrungen ist. Diese Arbeit konzentriert sich auf zwei dieser Beschränkungen und legt den Grundstein für das System und die damit verbundene Kollaboration um nicht abgehängt zu werden. Der erste Punkt ist die Fähigkeit den Beobachtungszeitplan durch das Erlauben von Beobachtungen während Mondlichts zu erweitern. Da die Detektoren sehr empfindlich sind war zusätzliches Licht im Himmel oder der Umgebung immer ein rotes Tuch. Dies ist nicht zuletzt ein Grund dafür, warum sich der Standort an einem so abgelegenem Ort im Khomas Hochgebirge im Herzen Namibias befindet. Der wissenschaftliche Schwerpunkt hat sich in den letzten Jahren zunehmend verschoben. Von systematischen Beobachtungen und dem Abtasten des Gammastrahlen-Himmels hin zur Untersuchung von Gelegenheitszielen, bei denen das System auf den Alarm von anderen Experimenten reagiert, wenn eine Quelle im Himmel außergewöhnliche Aktivität zeigt. Da diese Alarme zeitlich nicht vorhersagbar sind stellte sich die Frage, ob Beobachtungen während mäßigem Mondlicht möglich sind. Das hierfür entwickelte Modell der Helligkeit des Nachthimmels liefert diese Antworten zusammen mit verschiedenen Strategien die den Betriebszyklus des Systems um zusätzliche 60% bis 88% erhöhen. Der zweite Teil der Arbeit geht die andere Richtung: Anstatt mehr Daten zu nehmen wird die Möglichkeit untersucht mehr der bereits vorhandenen Daten mittels maschinellem Lernen auszuwerten. Aufgrund von strengen Qualitätskriterien in den Standardanalysen werden viele Ereignisse zwar aufgezeichnet, aber nie in einer Auswertung verwendet. Mit der modernen verfügbaren Hardware ist es möglich Algorithmen mit tiefen maschinellen Lernen zu verwenden, die in der Lage sind Daten zu analysieren welche normalerweise verworfen würden. Ergebnisse des aktiven Galaxienkern PKS 2155-304 werden gezeigt, bei denen das Signalrauschverhältnis der Auswertung von 11.6 auf 12.4 verbessert werden konnte. In diesem Sinne verfolgen beide präsentierten Themen das gemeinsame Ziel die Grenzen dieses etablierten Systems zu erweitern um an der Front moderner Forschung im Bereich der Astroteilchenphysik bleiben zu können.

# Contents

# Chapter 1.

# Introduction to Astroparticle Physics

## 1.1. Victor Hess discovers Cosmic Rays

It is not by chance, that the collaboration I was thankfully part of during the work on this thesis and their telescope system in Namibia, which will be introduced later in detail, are named H.E.S.S. (High Energetic Stereoscopic System). At the beginning of the last century, there was no particle physics, not to mention astroparticle physics. In fact, it was just a few years earlier in 1896, that radioactivity had been discovered through Antoine Henri Becquerel and was therefore a hot research topic in those days. It is amazing what development science and mankind as such went through in those last 100 years, as the existence of those research topics seems to be taken for granted nowadays. The H.E.S.S. telescope system was officially inaugurated in 2004, 40 years after the passing of Austrian physicist Victor Franz Hess, with the acronym chosen to honor his life work.

V. Hess undertook his famous balloon flights in 1911 and 1912, where he originally wanted to prove that the penetrating, ionizing radiation, which was measurable in closed containers was coming from radioactive decays in the earth's crust. [Hess, 1912] A decrease in rate with increasing height over ground due to more absorption from the air was postulated, but for example Theodor Wulf could not prove this with enough statistical evidence while climbing the tallest building of his time (the Eiffel Tower in Paris) with his measuring device [Wulf, 1910]. Wulf's result even hinted at an *increasing* rate with height, but the significance was not high enough and the influence of the Eiffel Tower's steel structure was not fully understood. Today we know, that the radioactive elements within the metal structure did definitely influence his results.
Hess on the other hand had the possibility to measure radiation during 7 balloon flights and thus covered a wider range in altitude difference than for example Wulf before him and was not constraint to climbing structures. He managed to seal both of his electroscopes to eliminate any bias from change in air pressure. His measurement results are summarized in figure 1.1.

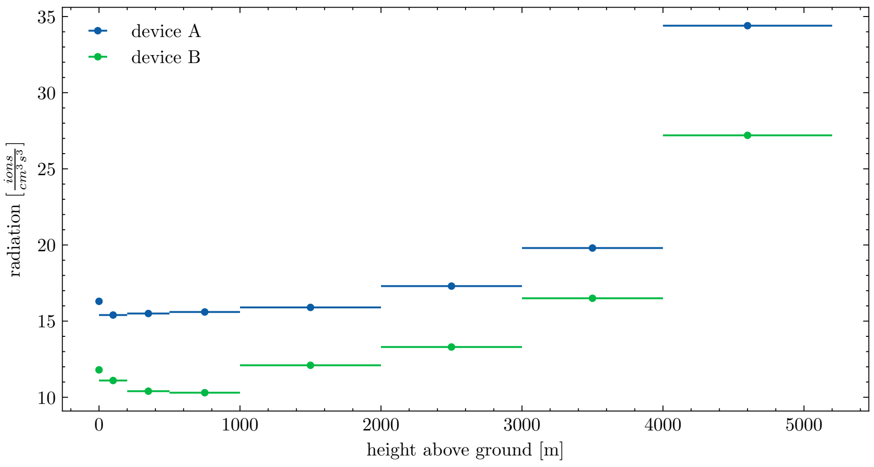


Figure 1.1.: The summary of Victor Hess's balloon flights. Horizontal bars show the height bins. Although the data points are calculated as averages over several flights with varying conditions, the trend is still obvious. (plotted after "*Tabelle der Mittelwerte*" on page 1089 of Hess [1912])

Hess' conclusion after the last and highest flight up to over 5000 meters above sea level was the following [Hess, 1912]:

> Die Ergebnisse der vorliegenden Beobachtungen scheinen am ehesten durch die Annahme erklärt werden zu können, daß eine Strahlung von sehr hoher Durchdringungskraft von oben her in unsere Atmosphäre eindringt und auch noch in der untersten Schicht einen Teil der in geschlossenen Gefäßen beobachteten Ionisation hervorruft. [...] Da ich im Ballon weder bei Nacht noch bei einer Sonnenfinsternis eine Veränderung der Strahlung fand, so kann man wohl kaum die Sonne als Ursache dieser hypothetischen Strahlung ansehen, wenigstens solange man nur an eine direkte $\gamma$-Strahlung mit geradliniger Fortpflanzung denkt.
>
> The results of the presented observations seem to be most likely explained by the assumption that a radiation with strong penetration power enters our atmosphere from above and even in the lowest layer causes a part of the observed ionization in closed vessels. [...] Since I did not find a change of the radiation in the balloon neither at night nor during an eclipse, one can barely see the sun as the origin of this hypothetical radiation, at least as long as one only thinks of a direct $\gamma$-radiation with straight propagation.

This publication is often referred to as the cradle of astroparticle physics. It was widely accepted as it could proof that indeed there is radiation coming from the ground and the decreasing rate is noticeable in the first data points, but quickly gets overcompensated by the newly detected strong cosmic radiation from above. Since then, many research fields evolved around this radiation, its composition and origin. The last piece is the one with the most still open questions. What are the objects and mechanisms in the universe, that are able to produce or accelerate particles to the enormous energies measured locally today?

## 1.2. Spectrum and Composition of Cosmic Rays

The energy spectrum of cosmic rays has been relatively well-studied over an astonishing range. In good approximation, it can be described as a rapidly falling power law in energy $\frac{dN}{dE} \sim E^{-\alpha}$ with the index $\alpha \approx 2.8$. Looking closer however, there is more structure to the spectrum. At the so called "knee" around $10^{15}$ eV, there is a significant steepening from $\alpha = 2.7$ to $\alpha = 3.0$ and a second steepening at about $5 \cdot 10^{17}$eV ($\alpha = 3.3$) is followed by a harder spectral index ($\alpha = 2.7$) above $5 \cdot 10^{18}$ eV, which form the "second knee" and the "ankle" as they can be seen in figure 1.2.

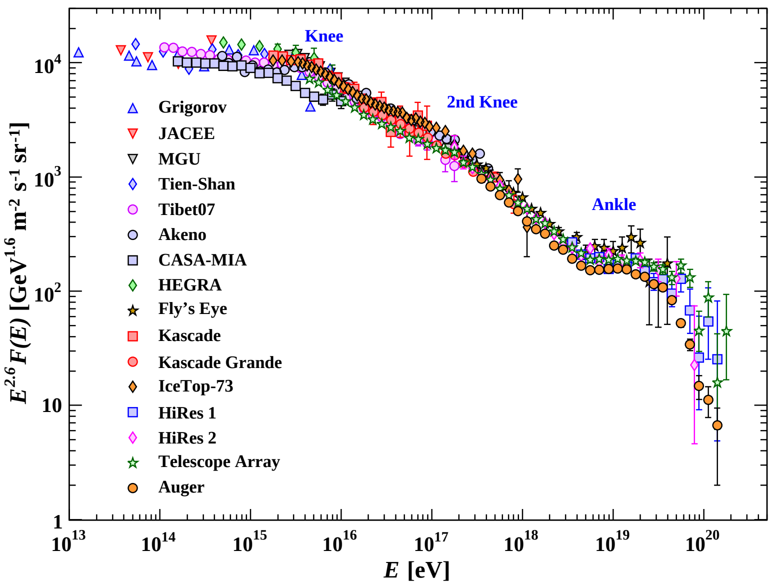


Figure 1.2.: The cosmic ray spectrum. The data points have been scaled with a factor of $E^{2.6}$ to better visualize the changes in power law indexes. Figure from Patrignani, C. et al. (Particle Data Group) [2016]

These changes in the spectral shape are thought to be connected with changes in the composition and sources of the particles. Below the "knee", most cosmic rays are believed to origin from super nova remnants inside our galaxy and the decrease in flux beyond the "knee" can be described with particles escaping the magnetic field of our galaxy as their gyroradii $r_g$ increase.

$$r_g = \frac{p_\perp}{|q|B} \tag{1.1}$$

Assuming e.g. a proton with charge 1 $e$ and neglecting the rest mass one can write:

$$r_g \approx \frac{E/c}{e\ B}. \tag{1.2}$$

Using a magnetic field strength of our galaxy of $3\,\mu$G [McMillan, 2011] and the energy of the "knee" $E = 5 \cdot 10^{18}$ eV, the gyroradius turns out to be

$$r_g \approx \frac{5 \cdot 10^{18}\,\mathrm{eV}/c}{e\ 3\,\mu\mathrm{G}} \approx 1800\,\mathrm{pc}, \tag{1.3}$$

which easily exceeds the average height of the milky way of ($\approx 300\,$pc) and is in the order of 1/10 of the galactic radius ($\approx 20 - 30$ kpc).
Cosmic rays are composed mainly of protons at lower energies and show a trend to a more prominent abundance of heavier nuclei towards the ankle [Abraham et al., 2010], or as Bertram Schwarzschild put it in the headline of his Physics today article [Schwarzschild, 2010]:

> The highest-energy cosmic rays may be iron nuclei. Or perhaps, at energies far beyond what terrestrial accelerators can produce, protons just look fat.

## 1.3. Gamma-Rays in the Context of Cosmic Ray Research

The answer to the question *where do Cosmic Rays come from?* is twofold: What are the sources and/or mechanisms that produce the observed spectrum of particles and where do we find them?
To generate the bulk of observed energy density of cosmic rays, a power of $\approx 2 \cdot 10^{41}$erg s$^{-1}$ is necessary. Known objects to generate this amount of energy are e.g. super novae, if there are 1-3 per century with an average kinetic energy of $2 \cdot 10^{51}$ erg and at least 10 % of the energy is converted into the production of super relativistic particles [Bykov et al., 2018]. But even though super novae are the biggest explosions possible in the universe, they are thermal processes and the ultra high energy particles observed in cosmic rays can only be

explained with non-thermal acceleration mechanisms. In 1949, Enrico Fermi proposed an acceleration mechanism using randomly moving "magnetic mirrors" [Fermi, 1949].

If a particle collides head on with such a mirror, it would gain energy and similar lose some of its velocity in a tailing collision. If these mirrors are thought as moving interstellar magnetized clouds, the probability of up-scattering would be higher than that of decelerating collisions and the particles would gain kinetic energy on average. This random process is now called second-order Fermi acceleration, because the mean energy gain per bounce depends on the relativistic mirror velocity squared $\beta^2 << 1$. This process yields a power law for the spectrum, but without a fixed value for the index. It is also rather inefficient and particles would need in the order of $10^6$ years to reach the ultra high energies observed.

The much more efficient first order Fermi acceleration was proposed years later in the 1970s. It considers particles being trapped within a shock front, where they diffuse from upstream to downstream and vice versa. When viewed in an appropriate reference frame every encounter is head-on with a mean fractional increase in energy. The resulting spectrum can be shown to be

$$\frac{dN}{dE} \propto E^{-2} \tag{1.4}$$

for a strong non-relativistic shock. Although this simple picture returns a value for the power law index, it is not quite at the experimentally measured sloped of $\alpha \approx 2.8$. More sophisticated considerations have to be taken into account to fully explain the observed cosmic ray spectrum. See for example Bell [2013] for detailed explanations or Achterberg et al. [2001] for a description of ultra-relativistic shocks. Such shock fronts are believed to be found at the expanding shells of super nova remnants or jets of active galactic nuclei.

The problem: Cosmic rays are charged particles. Even though precise measurements would allow to draw a lot of conclusions about the production mechanisms and the source properties, one can not link the particles arriving at earth to a location in the sky, since they have been arbitrarily deflected by galactic magnetic fields on their journey through the universe to us. At least below an energy of several EeV, at which the gyroradius becomes greater than the diameter of the galaxy. See the publication on "Correlation of the Highest-Energy Cosmic Rays with Nearby Extragalactic Objects" by The Pierre Auger Collaboration et al. [2007]. This results in a mostly homogeneous diffuse distribution of arrival directions.

Neutral particles created right at the source of such galactic accelerators however do travel (aside of maybe gravitational lensing effects) in straight lines and if detected at earth can be traced back to point to their origin: Photons (gamma-rays) and neutrinos. Neutrinos show incredible low cross sections, which makes them perfect messengers as they do barely interact

with anything on the way from their source to earth. Unfortunately it is the same property that makes them very hard to detect.
This work focuses on gamma-rays. There are two main production scenarios which will be described briefly below, followed by a deeper explanation of the detection method and the introduction to the H.E.S.S. gamma-ray telescopes in the next chapter.

## Hadronic Scenario

If one looks at protons as representatives of cosmic ray particles, and their possible interactions during or just after the aforementioned acceleration to ultra high energies, they are likely to undergo deep inelastic scattering with other protons or nuclei in the interstellar medium, resulting in the creation of mesons. In the simplest picture these would be pions from proton-proton interactions:

$$p + p \rightarrow p + p + \pi^0 + \pi^\pm + \cdots \tag{1.5}$$

Those secondary particles then typically decay via the following channels

$$\pi^0 \rightarrow \gamma + \gamma \tag{1.6}$$

$$\pi^+ \rightarrow \mu^+ + \nu_\mu \rightarrow e^+ + \nu_e + \bar{\nu}_\mu + \nu_\mu \tag{1.7}$$

$$\pi^- \rightarrow \mu^- + \bar{\nu}_\mu \rightarrow e^- + \bar{\nu}_e + \nu_\mu + \bar{\nu}_\mu \tag{1.8}$$

to produce the neutral messengers that point back to their origin. Measuring the gamma-ray flux, the lateral morphology and spectral shape, the host object can be studied.

## Leptonic Scenario

The other possible scenario to create a gamma-ray emission related to cosmic rays is the so called leptonic scenario. In this case, electrons or positrons produce gamma-rays via up-scattering of ambient photon fields, like the cosmic microwave background or local infrared and optical radiation fields. This process is known as the inverse Compton effect.

Often, both scenarios are plausible explanations to the gamma-ray origin, but in either case, the measured flux can be used to study astrophysical objects like the young shell-type supernova remnant RX J0852.0-4622 (also know as *Vela Junior*). Its morphology in gamma-rays has first been resolved by the H.E.S.S. collaboration [Aharonian et al., 2007a], showing a clear shell structure of the shock front. In the early publication the hadronic scenario was favored, but newer studies [H.E.S.S. Collaboration et al., 2018] still can not exclude the leptonic scenario completely.

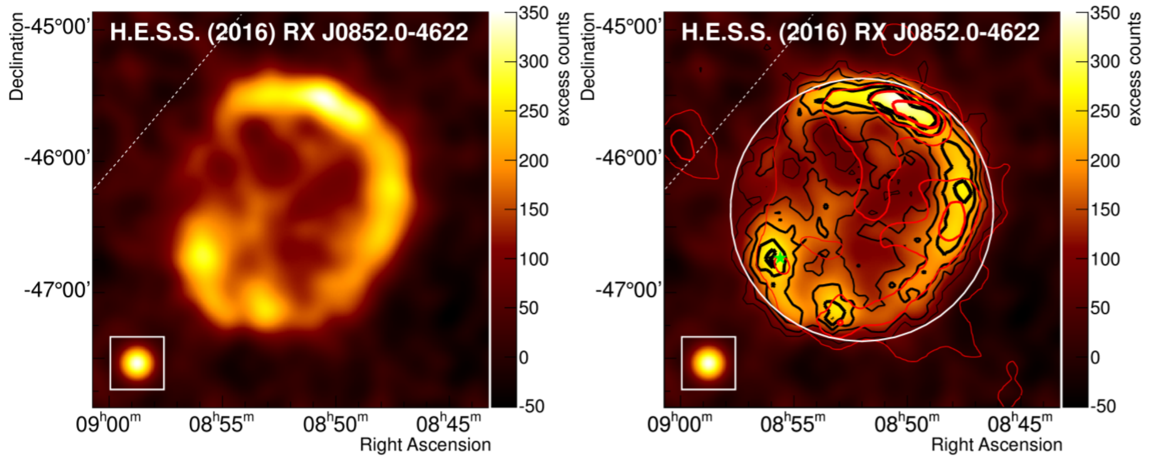


Figure 1.3.: Left: exposure-corrected excess map for RXJ0852.0-4622. The white dashed line shows the position of the Galactic plane; the inset shows the PSF of the analysis at the same scale for comparison.
Right: same as in the left panel, but additionally the boundary of the ON region is shown as a white circle and the significance contours at 3, 5, 7, 9, and 11 $\sigma$ are shown in black with increasing line width for increasing significance. In addition, X-ray contours from the ROSAT All Sky Survey for energies larger than 1.3 keV are shown in red. The X-ray contours were derived at 25, 50, 75, and 100 counts. Figure and caption adopted from H.E.S.S. Collaboration et al. [2018]

# Chapter 2.

# Introducing H.E.S.S.

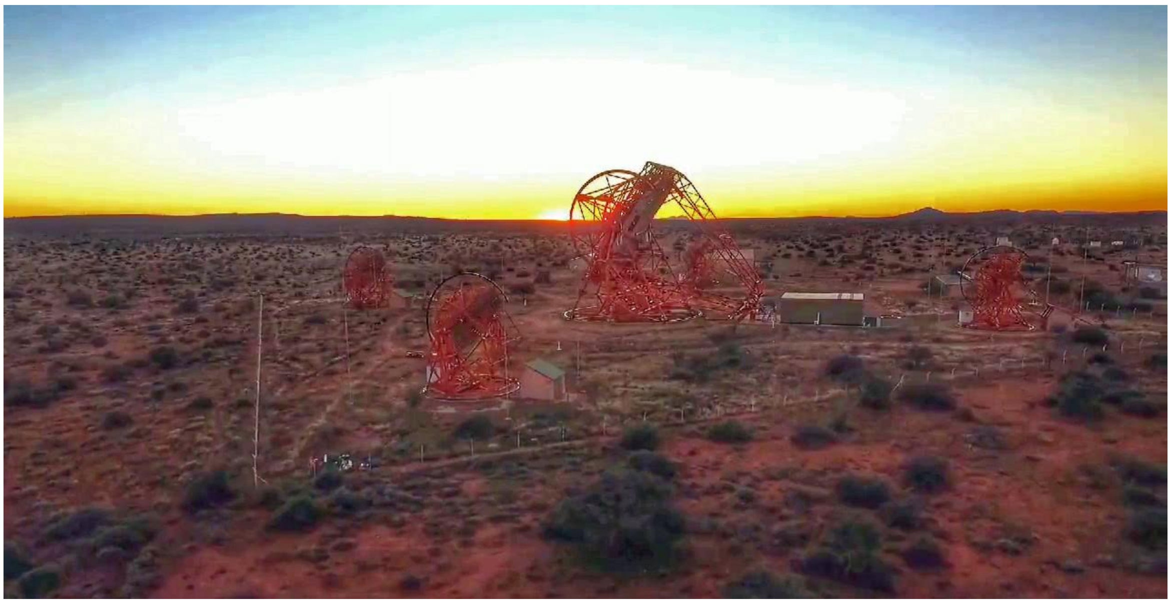


Figure 2.1.: The H.E.S.S. array at sunrise. Image by Matthias Büchele, May 2018.

The High Energy Stereoscopic System (H.E.S.S.) is an array of five imaging air Cherenkov telescopes (IACTs) located at the Khomas Highlands in Namibia ( $23°16'0''$S, $16°30'0''$E). The first four telescopes are in operation since 2003 (H.E.S.S. Phase-I), while the fifth was added to the array in 2012 (H.E.S.S. Phase-II). The four 12 m telescopes are arranged in a square with side length of 120 m and have a total mirror area of 107 $m^2$ each [Aharonian et al., 2004]. The bigger Cherenkov telescope 5 (CT5) with a total mirror area of 614 $m^2$ was placed in the center of the four other telescopes. Figure 2.1 shows the telescopes during sunrise. For security measures they are parked pointing south, away from the sun.
Both parts of this work only use data of the four small telescopes.

## 2.1. From Gamma Rays to Events on disc, the IACT principle

As the name *imaging air Cherenkov telescope* implies, air is an important component to operate this kind of device. But it does not mean the pneumatics involved in the drive system, rather than the atmosphere above the telescope. The Earths atmosphere is only transparent to parts of the electromagnetic spectrum. For example visible light, but not X-rays or gamma-rays (see Fig. 2.2). Although that is a good thing for all life on Earth, it is bad for science as this type of radiation is not directly accessible at ground level.

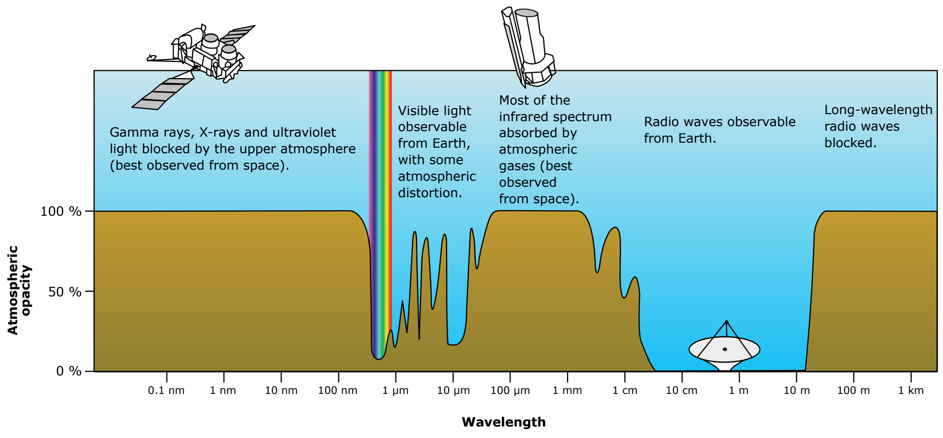


Figure 2.2.: Illustration of the atmospheric opacity. Image credits: NASA, public domain.

To observe such short wavelengths and thus high energies one has to go above the atmosphere (or at least very high in altitude). This was and is done in the past and the present via balloons and satellites, but as the spectrum of cosmic radiation decreases rapidly in flux with increasing energies (see chapter 1.2), these methods become less efficient and economic. Flux is defined as the number of particles (or photons) per area, energy, time and solid angle. Since most experiments aim for high resolution in energy and solid angle, and it is not always feasible to do integrated measurements over a certain amount of time, the only parameter to tweak is the effective detector area. But as with many space-born experiments, size is relative. The *Large Area Telescope* on board of the *Fermi Gamma-ray Space Telescope* for example has an effective detector area of roughly $1\,\mathrm{m}^2$. This is where the enormous mirror dish areas of the H.E.S.S. telescopes come to play.

### 2.1.1. Particle Showers in the Atmosphere

Since direct measurements of gamma-rays is not possible at ground level and huge detectors cannot be brought to space, an indirect method is used. When cosmic radiation enters the atmosphere there are two types of interactions which will happen depending on the nature of the particle: hadronic and electromagnetic.

For a detailed explanation the reader is advised to see Matthews [2004]. In the following a brief summary will be given.

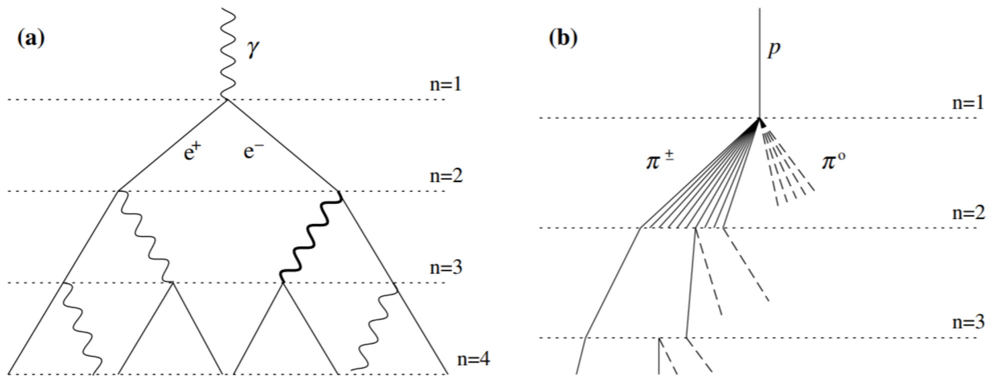


Figure 2.3.: Schematic views of (a) an electromagnetic cascade and (b) a hadronic shower. In the hadron shower, dashed lines indicate neutral pions which do not re-interact, but quickly decay, yielding electromagnetic subshowers (not shown). Not all pion lines are shown after the n = 2 level. Neither diagram is to scale. Figure and caption from Matthews [2004]

A high energetic photon will transform into an electron-positron pair via pair production on average after a charatceristic distance proportional to the radiation length in the medium. These still very energetic particles emit Bremsstrahlung - new photons, which themselves create $e^-e^+$ pairs. The process repeats until the energy per particle reaches a critical limit which is too low for either of the two processes.

Protons, as the main representative of hadronic particles in cosmic radiation, have additional interaction channels. In this case the most likely process is the creation of pions, which themselves induce electromagnetic subshowers (Fig. 2.3).

Even at TeV energies the total path length until the cascade stops is small compared to the atmospheric depth, which rules out direct measurements of the secondary particles to gain insight in the primary particles properties as direction of origin, particle type and energy.

Here the $C$ in $IACT$ comes to play: The Cherenkov effect. It is named for Soviet physicist Pavel Cherenkov, who shared the 1958 Nobel Prize in Physics for its discovery. The speed of light in vacuum $c$ is a universal constant. Nothing can travel faster. But in dielectric media such as water or air, the speed of light is decreased by a factor known as the refractive index $n$. Ultra relativistic particles such as those described in the air showers above can exceed this speed and therefore travel faster through a medium than light propagates in that medium, leading to a trailing light cone with a specific opening angle $\cos\theta_c = \frac{1}{n\beta}$ with $\beta = \frac{v}{c}$ (see figure 2.4). This effect is similar to super sonic objects that are followed by a shock wave audible as sonic boom.

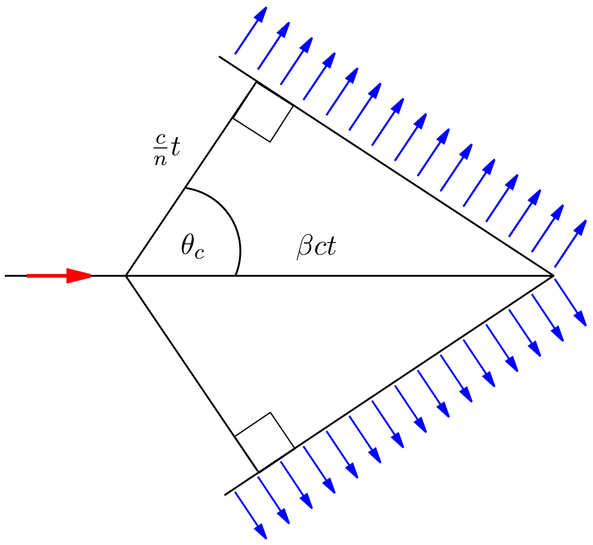


Figure 2.4.: The geometry of the Cherenkov radiation shown for the ideal case of no dispersion. The blue arrows show the wave front formed of constructive interference. Image credits: Arpad Horvath under the CC-BY 2.5 license. [*]

This Cherenkov radiation is in first order approximation visible blue light. The reason why human eyes don't see blue flashes all over the nightsky is its timescale. A particle shower is fully developed within a few nanoseconds, but the "integration" time of human vision is much larger, resulting in the Cherenkov light from such cascades not being perceptible. Nevertheless it is possible to take images of those showers with dedicated fast camera hardware, which will be explained in the following section.

### 2.1.2. Camera Hardware

The H.E.S.S. Phase I cameras consist of 960 pixels made out of photomultiplier tubes (PMTs), sensitive to single photons with a total field of view of about 5° in diameter. The deadtime is around 450 µs. The H.E.S.S. Phase II camera on the fifth telescope (CT5) was built in a similar fashion with 2048 PMTs but since it was designed several years later, more advanced readout electronics were available and used, allowing to lower the deadtime down to 15 µs.

Starting in 2015 the four cameras of the phase I telescopes have been upgraded to counter the increasing number of hardware failures due to aging and most of all reduce the deadtime (down to 6 µs) to keep up with the higher triggerrate of CT5 which is at roughly 1.5 kHz compared to the 200-300 Hz of the small telescopes [Giavitto et al., 2015]. The signals of the individual PMTs are amplified by two separate channels per pixel with different gains to

[*]Adapted from https://en.wikipedia.org/wiki/Cherenkov_radiation#/media/File:Cherenkov.svg, Creative Common license: https://creativecommons.org/licenses/by-sa/2.5/legalcode

have a wider range. The low gain channel sacrifices resolution for the benefit of still being linear without clipping strong signals. With the high gain channel on the other hand it is possible to resolve single photo electron peaks from the electronic and dark noise.
The final signal is taken from either of the channels or a linear interpolation if it falls into the region of overlapping linearity of the two channels and at this point is given in "units" of $ADC$ counts, which stands for the analog to digital converter values obtained from the corresponding hardware.

$$A = (1-\epsilon)\cdot ADC^{HG} + \epsilon \cdot ADC^{LG} \tag{2.1}$$

with

$$\epsilon = (ADC^{HG} - 150)/(200-150) \tag{2.2}$$

To trigger an event, multiple stages are required. If a large enough cluster of pixels in a camera reaches the set threshold, the camera sends a signal to the central trigger, which will initiate a full array readout only if at least two telescopes were hit within a short period of time. This on the one hand assures stereoscopic data, which can be reconstructed with more precision, and on the other hand lowers the number of accidental triggers through background.

### 2.1.3. Image Calibration and Cleaning

To convert from $ADC$ counts to an equivalent of photo electrons (p.e.) on the PMT cathode the following formulas are used:

$$A^{HG} = \frac{ADC^{HG} - P^{HG}}{\gamma_e^{ADC}} \cdot FF \tag{2.3}$$

$$A^{LG} = \frac{ADC^{LG} - P^{LG}}{\gamma_e^{ADC}} \cdot FF \cdot \frac{HG}{LG} \tag{2.4}$$

With $P^{HG}$ and $P^{LG}$ being the pedestal positions for both channels measured in $ADC$ counts. These are the baselines of signal in the absence of Cherenkov events, which is dominated by night sky background with some addition of pure electronic noise (the *dark* pedestal).
$\gamma_e^{ADC}$ is the absolute gain of the high gain channel in units of $ADC$ counts per p.e. . It is calculated from special *single photo electron* (singlePE) runs, where the cameras are parked within the shelters to block any stray light and then illuminated by a very faint pulsed laser diode to trigger events where only few (best case single) photo electrons are induced in the PMTs. The distance of those peaks to each other and to the pedestal baseline is directly

related to the gain factor which can therefore be calculated via a fit. For details see the H.E.S.S. publication on calibration [Aharonian et al., 2004].
$FF$ is the *flat fielding* coefficient, which accounts for gain and quantum efficiency variation within the camera pixels. It is obtained from another special run type where the camera is uniformly illuminated with a diffuse light source and the deviations of every pixel response to the mean of all pixels within a camera is calculated and stored. The inverse of that deviation is than multiplied as $FF$ factor to the pixels to guarantee a flat camera response.
The last piece is $\frac{HG}{LG}$, which denotes the gain ratio of the high to low gain channel. This is needed, since the low gain channel cannot resolve single p.e. peaks and therefore the gain has to be calculated relative to the other channel. It is obtained from data, where the channel outputs in the overlapping working regime of both gain channels are compared.

After all pixel values are converted to p.e. the cleaning step is applied. It suppresses the recorded noise in all pixels but the ones containing Cherenkov signal by applying a two thresholded tailcut. In the standard configuration, the so-called *5-10 cleaning* is used: A pixel is flagged as signal if its value is above 5 (10) p.e. and there is at least on neighboring pixel with more than 10 (5) p.e., otherwise it is set to zero. This method provides strong cleaning capabilities while still managing to not cut away too much of the edge of a signal patch, since pixel values as low as 5 p.e. are kept there.

## 2.2. Reconstruction of Event Properties

The main analysis takes place offline. There is a real time analysis (RTA) running on site, but it is trimmed to work fast and therefore does not provide a true and full calibration, but is rather used for the shift crew to be alerted if for example a source in the field of view shows an extraordinary high gamma-ray flux. In this case of a flare, it is not of first priority to calculate the flux with highest precision, but to be noticed and follow up the observation. Or on the other side if H.E.S.S. does follow an external alert to observe a target of opportunity, the shifters can estimate for how long it makes sense to stay on that target before returning to the regular schedule.
The next section will discuss the procedures to analyze the data.

### 2.2.1. Overview of Analysis Techniques

The IACT technique for gamma-ray detection is strongly background dominated. For a rule of thumb there are 1000 hadronic induced triggered images from cosmic rays for every gamma-ray images in the data. The first important step in every analysis chain is therefore

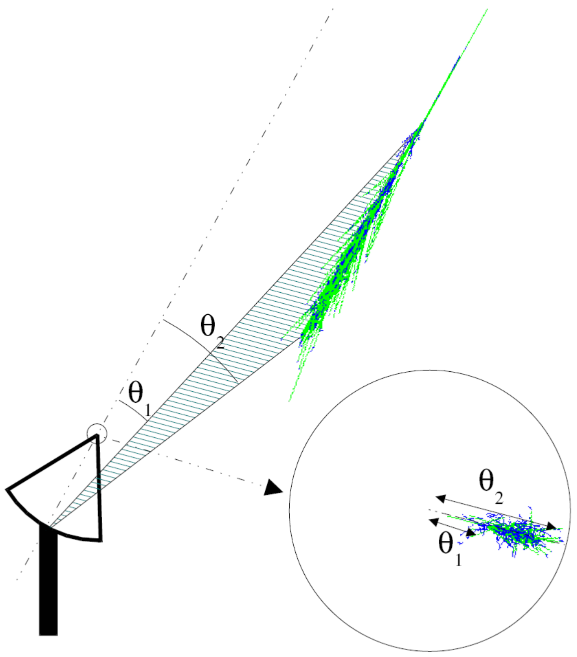


Figure 2.5.: The schematics and geometry of imaging an Cherenkov shower with a IACT. Figure from de Naurois [2012]

to suppress this background*. Looking back at the shower schematics in figure 2.3 at page 11, the main difference in the two shower developments is the presence of multiple sub-showers in the hadronic case. This leads to an overall more lateral spread light distribution at the ground level. Electromagnetic showers develop fairly straight downwards, close to the original trajectory of the primary particle. It is impossible for a shower to hit all four (five) telescopes head on and therefore the image of such a shower will always be a view from the side, resulting in the characteristic elliptical shape of the image in the camera plane. See figures 2.5 and 2.6.

In figure 2.6 the images of multiple cameras are overlayed to illustrate the advantage of stereoscopic imaging. The origin of the shower seen in the camera plane is located somewhere along the major shower axis. As illustrated in Fig. 2.5 this point is an angular distance of $\theta_1$ away from the top of the shower images. Remember that the top of the shower always corresponds to the central end in the camera image. Unluckily this distance $\theta_1$ is not known per se, since it relates to the height in the atmosphere the first interaction took place. Although this could be modeled and estimated, it is much easier to use stereoscopy where the origin is simply the intersection of all shower axis in the image.

*Except one is interested in using a device built for gamma-ray imaging to measure a proton spectrum. Cheers David, see thesis of Jankowsky (2020).

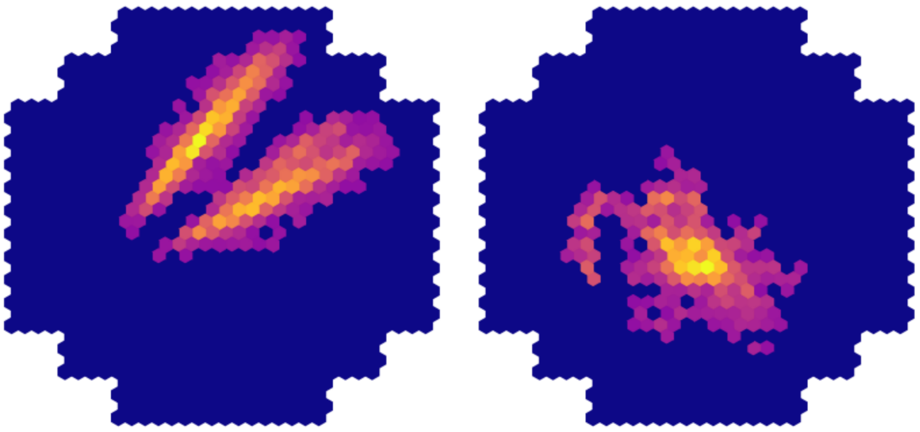


Figure 2.6.: Examples of Cherenkov shower images from simulations after cleaning. Left: gamma-ray induced electromagnetic shower with characteristic elliptical shape. The images of two cameras are overlayed to illustrate the stereoscopy. Right: Hadron induced shower with typical irregular shaped due to electromagnetic sub-showers

It has to be noted, that the shower images displayed here are hand picked extreme cases to represent their corresponding classes. In reality, gamma-ray signals do not always look that elliptical and hadronic showers are not always that irregular. The spectrum in between is continuous, that is why having a strong classifier to automatically separate the types is very important.

### The H.E.S.S. standard analysis

The standard analysis implementation is based on describing the image with its moments, which are named after M. Hillas, who proposed them in one of the most cited publications in gamma-ray astronomy [Hillas, 1985]. The Hillas parameters are (as illustrated in Fig. 2.7):

- *length* and *width* of an ellipse describing the geometry of the shower. Used mainly for event classification.
- The *size* of the image. The name of this parameter is misleading and does not relate to the ellipse's area, but is rather calculated as the total sum of all pixel amplitudes contributing to the event. It has therefore units of p.e. and is used for the energy reconstruction.
- *nominal distance* of the image's center of gravity to the center of the camera. Used for data quality selection cuts.
- orientation $\alpha$ of the ellipse relative to the camera center. Used for direction reconstruction in the stereoscopic analysis as described above.

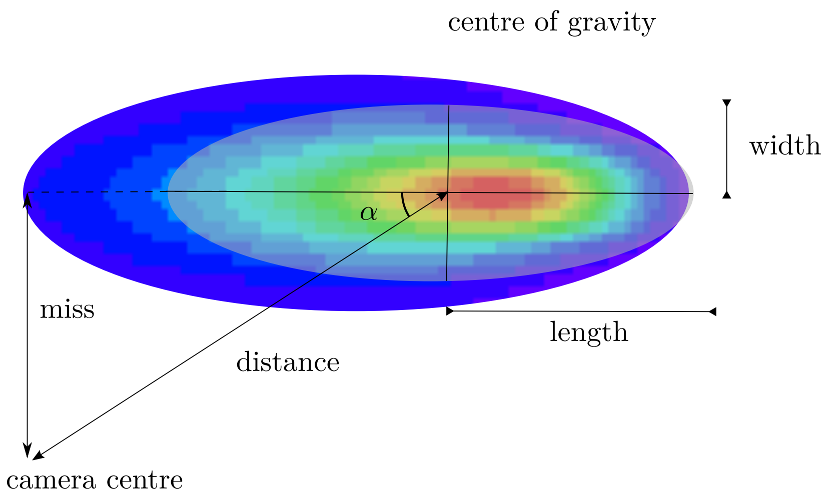


Figure 2.7.: The most important image moments as described by Hillas [1985]. Figure with best thanks from Veh [2018].

Those parameters can easily be calculated [de Naurois, 2012] considering every image as a set of pixels with coordinates $(x_i, y_i)$ and amplitudes $a_i$. It is noted with respect to chapter 4 on deep learning, that these calculations are independent of camera and pixel geometry and composition. All it needs are the coordinates of each pixel with respect to a camera reference frame.

The first and second moments are:

$$\langle x\rangle = \frac{\sum_i x_i a_i}{\sum_i a_i}, \quad \langle y\rangle = \frac{\sum_i y_i a_i}{\sum_i a_i} \tag{2.5}$$

$$\langle x^2\rangle = \frac{\sum_i x_i^2 a_i}{\sum_i a_i}, \quad \langle y^2\rangle = \frac{\sum_i y_i^2 a_i}{\sum_i a_i}, \quad \langle xy\rangle = \frac{\sum_i x_i y_i a_i}{\sum_i a_i} \tag{2.6}$$

with the variances and covariances:

$$\sigma_{x^2} = \langle x^2\rangle - \langle x\rangle^2, \quad \sigma_{y^2} = \langle y^2\rangle - \langle y\rangle^2, \quad \sigma_{xy} = \langle xy\rangle - \langle x\rangle\langle y\rangle \tag{2.7}$$

using intermediate variables:

$$\chi = \sigma_{x^2} - \sigma_{y^2}, z = \sqrt{\chi^2 + 4\sigma_{xy}} \tag{2.8}$$

$$b = \sqrt{\frac{(1+\chi/z)\langle x\rangle^2 + (1+\chi/z)\langle y\rangle^2 - 2\sigma_{xy}\langle x\rangle\langle y\rangle}{2}} \tag{2.9}$$

The final Hillas parameters then read:

$$d = \sqrt{\langle x\rangle^2 + \langle y\rangle^2} \tag{2.10}$$

$$l = \frac{1}{2}(\sigma_{x^2} + \sigma_{y^2} + z) \tag{2.11}$$

$$w = \frac{1}{2}(\sigma_{x^2} + \sigma_{y^2} - z) \tag{2.12}$$

$$\alpha = \arcsin(\frac{b}{d}) \tag{2.13}$$

$$size = \sum_i a_i \tag{2.14}$$

From these parameters the direction of the primary particle can be calculated in the camera frame and the source location in the sky is obtained via coordinate transforms. Another important parameter is the *core impact position*. It describes the position on the ground where the primary particle would have hit the Earth if it would not have created the shower and can be calculated knowing the particles origin and the azimuth angle at which it is seen from every telescope. It is a measure for "how far" away a shower happened and plays an important role in energy reconstruction together with the *size*, as showers appear bright if they are of high energy or very close and a dim shower might be either far away at higher energy or close to the telescope array with lower energy.

To calculate the event properties, the Hillas parameters of several (two to five, depending on the amount of triggered cameras) telescopes have to be combined. The *HEGRA* Collaboration already had multiple IACT telescopes operational and developed the *mean scaled width* and *length* (MSW, MSL) parameters [Hofmann, 1997].
The H.E.S.S. collaboration built on that principle and refined those to the *mean reduced scaled width* and *length* (MRSW, MRSL):

$$\text{MRSW} = \frac{1}{N_{tel}} \sum_{t=0}^{N_{tel}} \frac{w_t^{data} - mean(w_t^{MC})}{stddev(w_t^{MC})} \tag{2.15}$$

The expression for MRSL is equivalent. $N_{tel}$ is the total number of telescopes and $mean(w_t^{MC})$ and $stddev(w_t^{MC})$ are the interesting parts: Those values are obtained as the mean and standard deviation of measured width $w_t$ in telescope number $t$ from massive 2D lookup tables generated from gamma-ray Monte Carlo simulations, with the lookup axis being *size* and *core impact position*. There are four smoothed tables (*mean* and *stddev* for $w$ and $l$) for every combination of 13 zenith distances and 6 different source offsets each. Values in between are linear interpolated.
MRSW and MRSL tell how far away from the expectation in width and length a particular event was, given its image amplitude (*size*), the distance from the telescopes (*core impact*

*position*) and the telescope pointing direction. Since the lookups are generated from gamma-ray simulations only, a value far off from the middle of the distribution (zero) is a strong indicator for the event being of hadronic nature and therefore background. The event classifier, based on a boosted decision tree, mainly depends on those parameters.

Now that the direction and type of the primary particle is determined, the last missing piece of information is its energy. This is, as mentioned earlier, strongly dependent on *size* and *core impact position* and can therefore be obtained by another lookup scheme based on Monte Carlo Simulations.

### Model and ImPACT Analysis

There are two advanced analysis chains within the H.E.S.S. collaboration, namely: the *Model* [de Naurois and Rolland, 2009] and the *ImPACT* [Parsons and Hinton, 2014] analysis. For in depth understanding, the reader is referred to the individual publications on those methods.

Both techniques use a likelihood approach to fit the entire camera images and not only use the image moments like the *Hillas* method. The difference lies in to what the data is fitted. In the case of *Model* it is a semi-analytical shower model, that simply spoken produces gamma-ray like camera images given an energy, direction and core impact position. The background rejection is done via a goodness-of-fit-value, since hadronic events cannot be described very good with the model and thus can be identified this way.

The *ImPACT* analysis builds on top of an image-template library which is generated from gamma-ray simulations to which the fit is performed. Background rejection based on a similar goodness-of-fit value is mentioned in the publication, but discarded by the authors due to too strong dependencies on night sky background and the required knowledge of the SinglePE and Pedestal width. The *ImPACT* chain therefore uses a boosted decision tree (BDT) trained on simulations to classify the events based on their *Hillas*- and MRSW/MRSL parameters. Figure 2.8 shows the key performance plots of the three analysis methods. Both advanced methods clearly outperform the *Hillas* approach in angular resolution (lower is better) and are very comparable, with *ImPACT* being dominant at high energies and *Model* performing slightly better on the lowest energies.

The second plot showing the effective area, describes the classification efficiency and is calculated as the relative amount of detected MC gamma-simulations out of all simulated events multiplied with the area on the ground the simulations were scattered over. This value is crucial to obtain a flux spectrum and needs to be as high as possible over the full energy range. At high energies, *Model* has clear deficiencies and *ImPACT* shows the best efficiency.

At the low end it is again *Model* with the best performance and this time *ImPACT* being even slightly behind the standard *Hillas* approach.

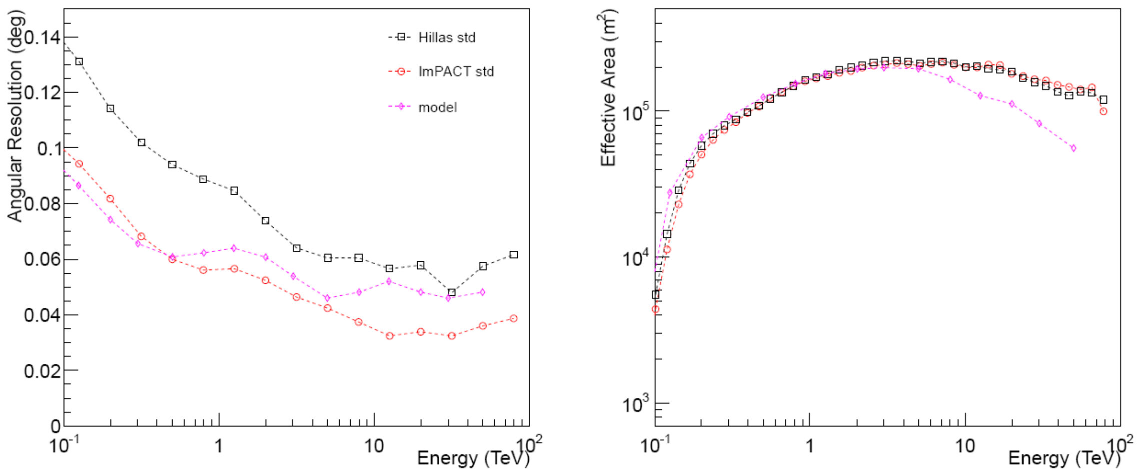


Figure 2.8.: Comparison of the three analysis chains. Left: Angular resolution, given as the 68% containment radius. Right: Effective area, calculated as the relative amount of detected MC gamma-simulations out of all simulated events times the area on the ground the simulations were scattered over. Figure from Parsons and Hinton [2014]

## 2.2.2. Limitations of the current Techniques

Apart from the slight differences the advanced methods show in the high and low energy extremes, both clearly outperform the traditional method. Nevertheless there might still be room for improvement, since all methods rely on data quality cuts. In the standard[*] configuration these preselection cuts are applied before any analysis step takes place:

- size $>$ 60 p.e.: Excludes very faint images or such with few pixels triggered.
- local distance $<$ 0.525 m: Ensures the event images are mostly contained within the camera.
- multiplicity $>=$ 2: Only stereo data will be analyzed.

These cuts sort out data that can usually not be analyzed with good precision. Different analysis approaches might be able to include them and boost the overall sensitivity of the experiment.
The size cut mainly affects low energy events and is not very hard. Even images with very few pixels will already pass the 10-5 tailcuts (see illustration in figure 1 in appendix A). Its main

[*]H.E.S.S. internally called *std_zeta*

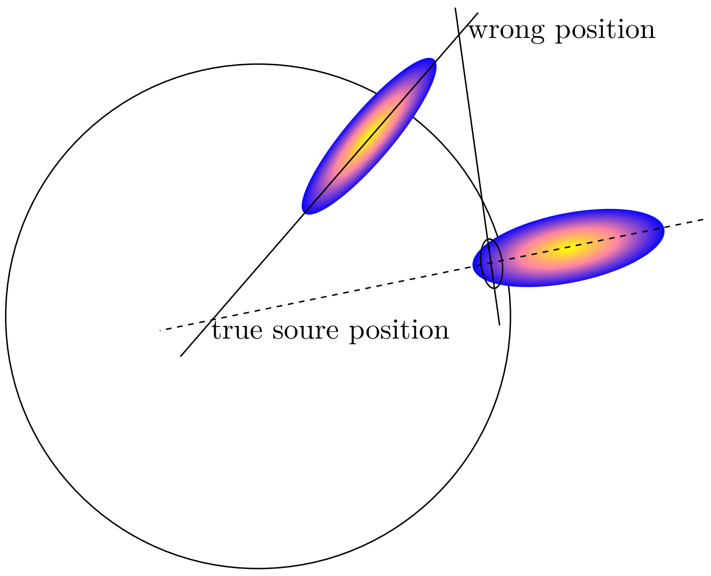


Figure 2.9.: The local distance cut only keeps events where the images center of gravity is closer to the camera center than a given radius to prevent misinterpretations of truncated events.

goal is to keep out noise images that accidentally made it passed the stereoscopic trigger and the cleaning step. There is no big potential in lowering the cut value here. Having a data set without a multiplicity cut is of course possible, but the big advantage of having an array of telescopes versus a single one then vanishes. The biggest potential for improvement lies in changing the local distance. Figure 2.9 illustrates why this cut was introduced in the first place: Shower images at the edge of the camera get truncated, leading to a misinterpretation of the main shower axis. While an experienced human would still be capable of identifying this case and trying to resolve the issue, the automated analysis of just the image moments or the fit of a template/model will definitely either fail or produce a wrong result. Since it is not a good idea to train an army of PhD students to look at event images by hand, maybe a computer could be trained to perform this task. More on that topic later in chapter 4.

To push the limits of H.E.S.S., this work focused on the two aspects presented in the following chapters:

Either enable more data taking to increase the total number of events recorded to disc, by allowing moonlight observations.

Or improve the data analysis to include more of the already recorded events, by using modern machine learning tools, which can analyze data with loosened cuts.

# Chapter 3.

# Moonlight Observations with H.E.S.S.

This chapter will present the development of the night sky background (NSB) prediction tool and show how it was used to make observations during moonlight happen with H.E.S.S. For the first 15 years of operation, the cameras were only switched on if the Sun is more than 18° below the horizon and there is no moon at all above the horizon. This strict conditions were chosen to guarantee best data quality and maximum security for the very sensitive PMTs within the cameras. But since both other big active IACT collaborations that use PMT based cameras (VERITAS and MAGIC) already showed that safe and successful operations under moderate moonlight are possible (see Archambault et al. [2017] and Ahnen et al. [2017] respectively), the time had come for H.E.S.S. to join the party. To sooth conservative voices and assure that no harm will be done to the newly upgraded cameras, predictive calculations had to be performed and presented to the collaboration and its board first.

## 3.1. The NSB Tool as Foundation

The goal is to deliver a user friendly and platform independent tool that can predict the brightness of a certain region in the sky for every location, time and date on earth. Python is chosen as cross platform programming language so that the final tool does not have to be compiled for a specific system. The final model functions can however be ported to a system of choice to be used e.g. within the data acquisition software (written in C++) on site in Namibia.

### 3.1.1. The early Model

As a starting point, the *MODEL OF THE BRIGHTNESS OF MOONLIGHT* from Krisciunas and Schaefer [1991] was used and implemented as is. The authors of that publication tried to find a compromise between pure functional model fitting and first hand derivatives and ended up with "an easy-to-use equation that will predict the moonlight brightness with an accuracy of 8% to 23%." [Krisciunas and Schaefer, 1991] Equations 3.1 - 3.7 are taken from this publication.

The two main components are the background brightness of the night sky $B_0$ and the contribution from scattered moonlight $B_{moon}$:

$$B_0 = B_{zen} 10^{-0.4k(X(zen)-1)} X(zen) \tag{3.1}$$

$$B_{moon} = f(\rho) I^*(\alpha) \times 10^{-0.4k\ X(zen_m)} [1 - 10^{-0.4k\ X(zen)}] \tag{3.2}$$

with

$$X(zen) = \frac{1}{\sqrt{1 - 0.96 \sin^2 zen}} \tag{3.3}$$

$$I^*(\alpha) = 10^{-0.4(3.84 + 0.026|\alpha| + 4 \cdot 10^{-9} \alpha^4)} \tag{3.4}$$

$$f(\rho) = 10^{5.36}[1.06 + \cos^2 \rho] + 10^{6.15 - \rho/40} \tag{3.5}$$

$B_{zen}$ is the sky brightness in zenith which scales the whole equation 3.1. Parameter $k$ describes the extinction coefficient in units of magnitudes per air mass. Equation 3.3 is their choice of atmospheric depth calculation, where $zen$ and $zen_m$ are the zenith distance of the point of interest and the Moon respectively. Equation 3.4 describes the illuminance of the Moon based on the phase angle $\alpha$, where $\alpha$ is the angular distance between earth and sun as seen from the Moon with full moon for $\alpha = 0°$ and new moon at $\alpha = 180°$ (see fig 3.13 on page 46 to easily adapt this convention). The equation for illuminance is derived from the following relation between illuminance in footcandles and the V magnitude of the Moon $m$:

$$I(m) = 10^{-0.4(m + 16.57)} \tag{3.6}$$

$$m(\alpha) = -12.73 + 0.026|\alpha| + 4 \cdot 10^{-9} \alpha^4 \tag{3.7}$$

The final piece for describing the brightness of the Moon $B_{moon}$ is the scattering function $f(\rho)$ (Eq. 3.5). It contains the overall scaling and the dependence to the separation angle $\rho$ between moon and point of interest in the sky, which will turn out to be one of the most important parameters when thinking about moonlight observations.

All angles in the mentioned equations are taken in degrees with the appropriate trigonometric functions.

In the end, the two components have to be summed

$$B = B_0 + B_{moon} \tag{3.8}$$

The resulting brightness is given in units of nanoLambert [nLb] and is therefore actually a measure of luminance*. The relation to SI units is given by:

$$1\ \mathrm{Lb} = \frac{1}{\pi} 1 \times 10^4\,\mathrm{cd\,m^{-2}} \tag{3.9}$$

To compute brightness values from the model the necessary inputs are:
The zenith distances of the Moon $zen_m$ and of the point of interest (POI) in the sky $zen$, angular separation of the Moon and POI $\rho$ and the moon phase $\alpha$.
Note that there is no explicit dependency on any azimuth angle, but practically one would take the two coordinate pairs ($az$, $alt$ with $alt = 90° - zen$) for moon and POI and calculate the separation $\rho$ on the go using the great circle distance on a sphere:

$$\rho(alt_1, az_1, alt_2, az_2) = \cos^{-1}[\sin(alt_1)\sin(alt_2) + \cos(alt_1)\cos(alt_2)\cos(az_1 - az_2)] \tag{3.10}$$

Given some values for the five parameters these relations can be used to draw an image of the nightsky by scanning the point of interest across the field of view. To do so a python program was developed which calculates the moons coordinates in the sky based on time, date and location of the observer on earth. The result can be viewed on screen or saved to a *fits*-file.
To compare the results and to improve the model, brightness data of the sky over the H.E.S.S. site is needed. No such direct measurements are available, especially not covering the whole sky and monitoring an extended period of time to cover a significant part of the parameter space. The tool *PyASB* developed and described by Nievas [2013] seemed promising. The exact details of the pipeline can be found in the author's thesis. Here is a simplified version of the general steps the tool provides:

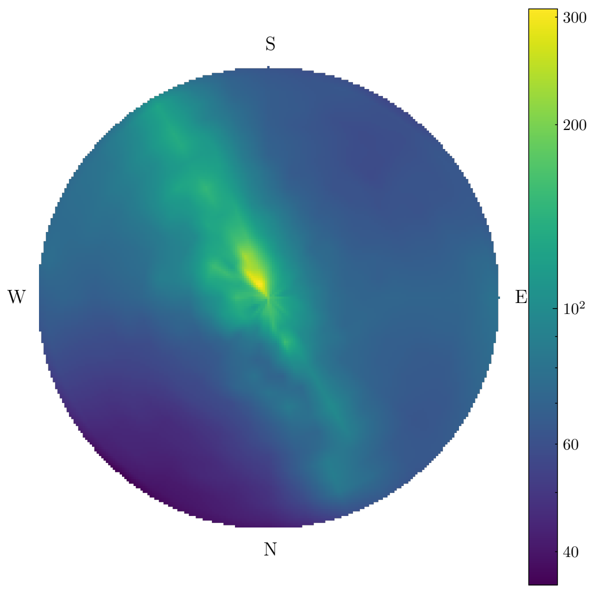
- load allsky image (CCD photograph)
- dark signal subtraction
- star detection using catalogs
- fitting the photometric data based on the tabulated star magnitudes
- resulting in a night sky background map with units [mag / arcsec$^2$].

At the H.E.S.S. site in Namibia, just 300 meters away from the Cherenkov telescopes, ATOM (the Automatic Telescope for Optical Monitoring) is located and with it its cloudmonitor: a *Starlight Xpress SXV-H5* CCD camera with fisheye lens to capture an image of the sky every

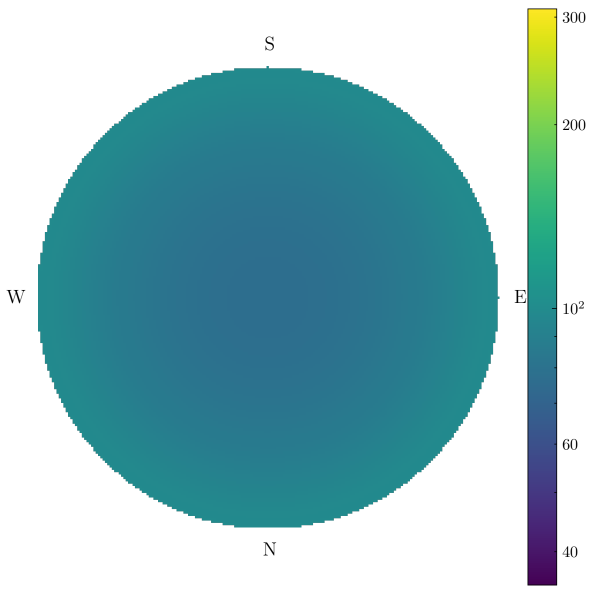
*Strictly spoken, brightness is the term for the subjective impression of the objective luminance, but for the rest of this work the two terms will be used interchangeable, since we will not be dealing with light sources far off the visual spectrum.

three minutes. This data source and the *PyASB* pipeline were used to generate verification data for the NSB model.

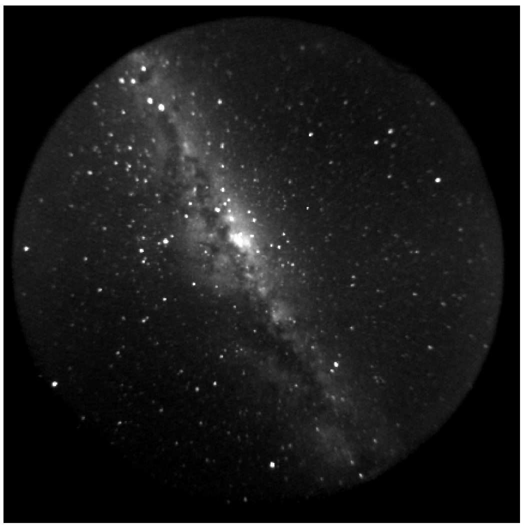
One month of data during the winter in Namibia (June 2010, 6664 images in total) was used for the analysis. Figures 3.1, 3.2 and 3.3 show the images from the all-sky camera together with the data obtained from those and the corresponding models for nights without moon, half moon and full moon respectively. The fourth plot always shows the relative deviation calculated as $\frac{\text{data}-\text{model}}{0.5(\text{data}+\text{model})}$. At the very first glance, one sees that the model lacks a description of the milky way. With the simple background description of $B_o$ (formula 3.1) which is basically just a mean brightness value with some added zenith dependency, the galactic plane is underestimated while extragalactic regions are too bright, leading to the relative deviations seen in figure 3.1d, where the edge of the milky way is the area of zero deviation. With higher moon phase, the moon induced brightness is dominant and the overall description of the model is much better. The original model by Krisciunas et al. did not claim to be valid for coordinates closer than 5° to the Moon, which is why no value is returned in this case.
Attention has to be paid to the lens flares and reflections of the direct moonlight in the CCD images. Those are wrongly interpreted as sky brightness values by the PyASB tool and lead to additional differences of data and model close to the Moon. Nevertheless, the model as a functional basis seemed promising as most of the sky regions fall within the ±0.25 contour lines.



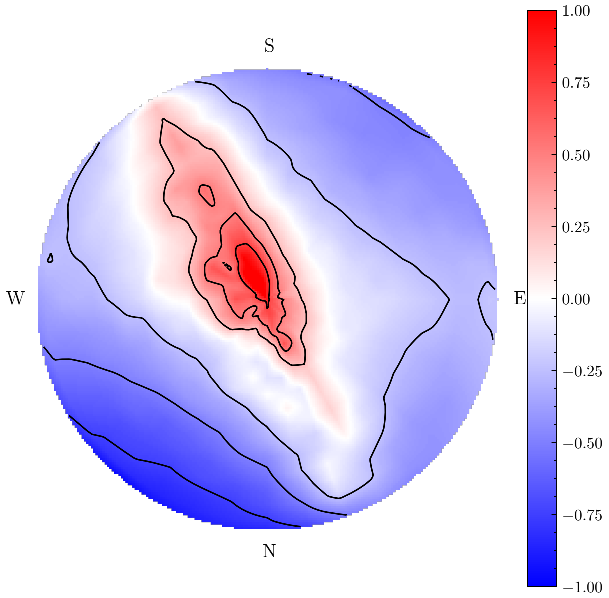
(a) PyASB Data from Image



(b) Simple model

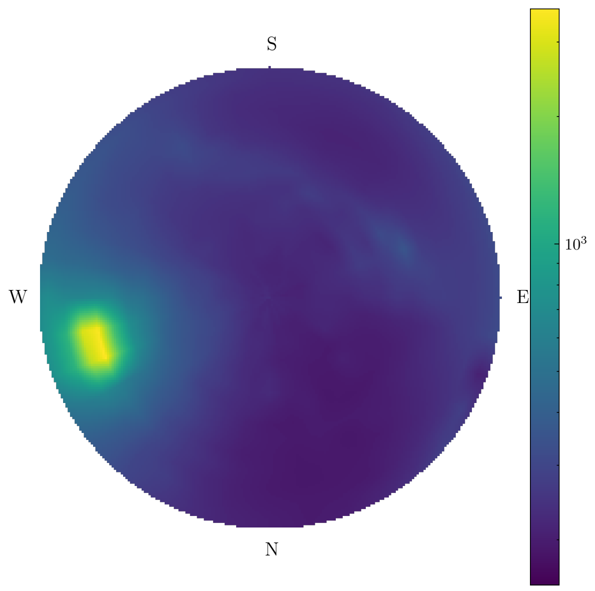


(c) CCD image

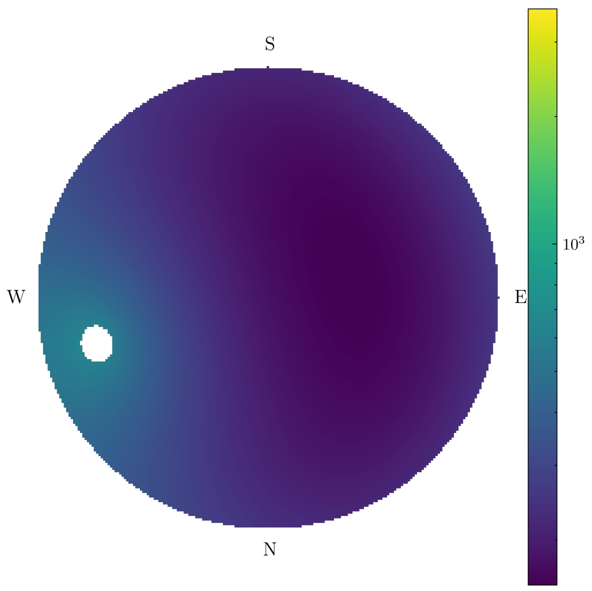


(d) Relative difference of data and model, contour lines at $\pm 0.25, \pm 0.5, \pm 0.75$

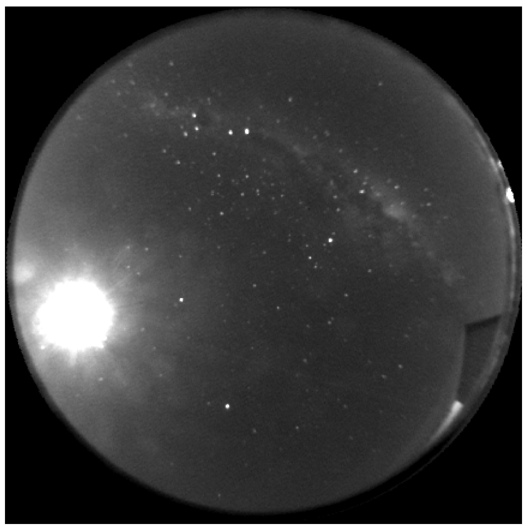
Figure 3.1.: Comparison of the data obtained from the CCD image using the PyASB software and the NSB model without moonlight (2010/06/11 00:02:21).



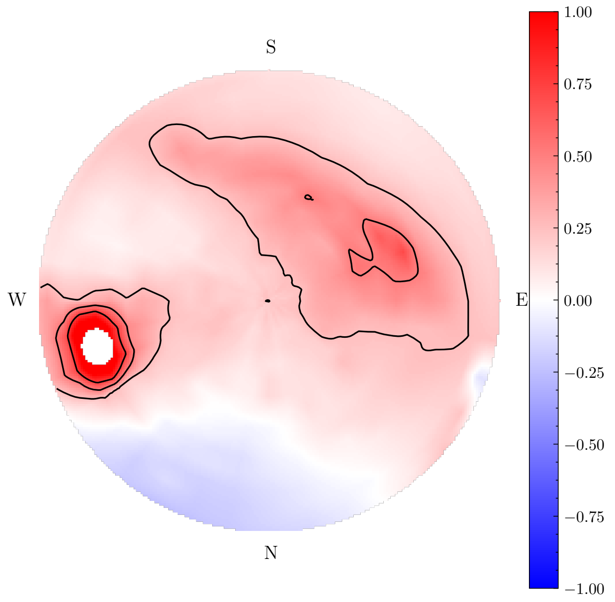
(a) PyASB Data from Image



(b) Simple model

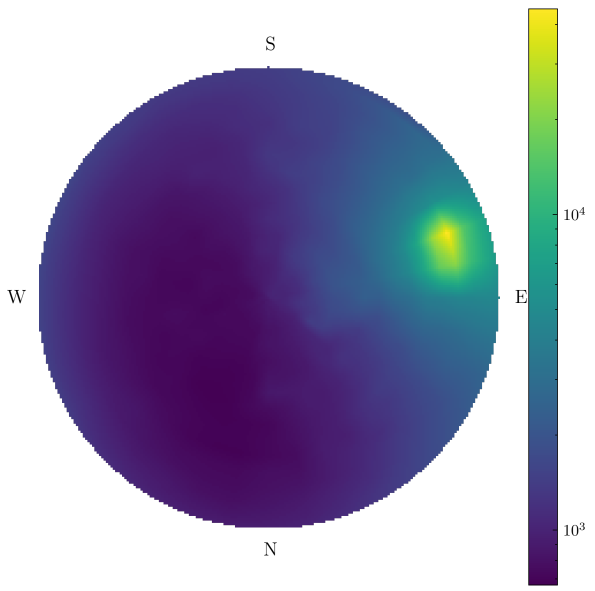


(c) CCD image

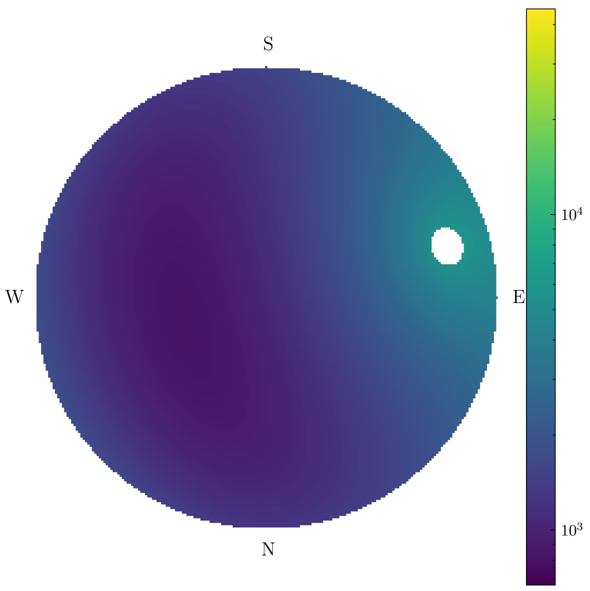


(d) Relative difference of data and model, contour lines at $\pm 0.25, \pm 0.5, \pm 0.75$

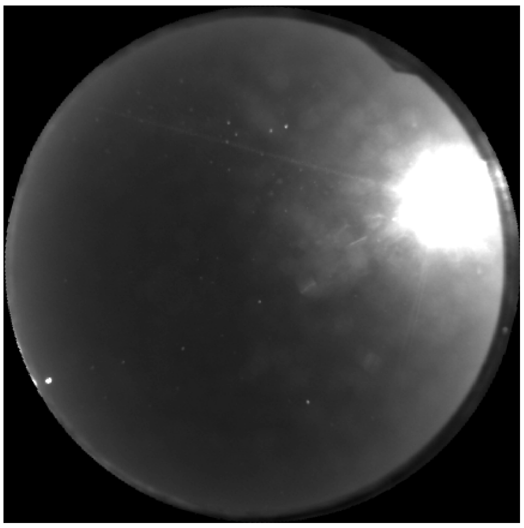
Figure 3.2.: Comparison of the data obtained from the CCD image using the PyASB software and the NSB model at half moon (2010/06/18 20:01:37).



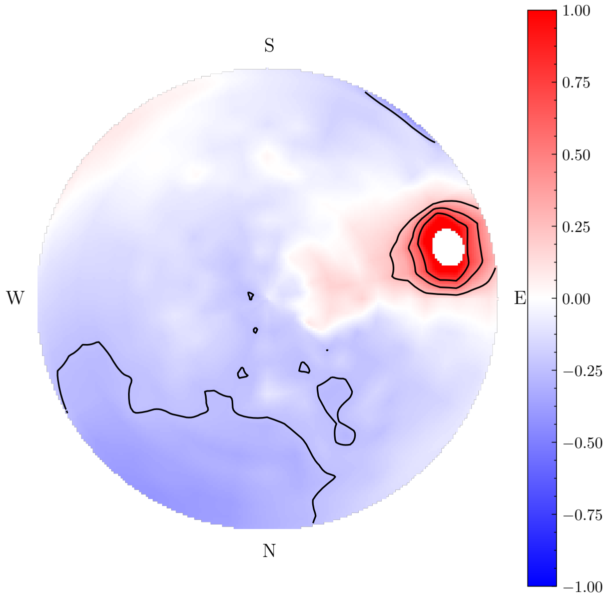
(a) PyASB Data from Image



(b) Simple model



(c) CCD image



(d) Relative difference of data and model, contour lines at $\pm 0.25, \pm 0.5, \pm 0.75$

Figure 3.3.: Comparison of the data obtained from the CCD image using the PyASB software and the NSB model at full moon (2010/06/25 18:01:09).

### 3.1.2. The advanced Model

In order to get a better description of the whole sky, starlight definitely had to be considered. At the end of 2016, which was the time the advanced model was developed, the *GAIA*-Collaboration released their first public data release [Gaia Collaboration et al., 2016]. It includes photometric data of over $10^9$ stars and is accessible online for download and via database queries. The python NSB tool was adapted to query and process that data automatically for the user and store a $4\pi$ skymap in arbitrary resolution in **H**ierarchical **E**qual **A**rea iso**L**atitude **Pix**elisation (HEALPix, Gorski et al. [1999]) with stored values of brightness in units of Magnitude per arcsecond squared. This step had only to be done once, since the background of fixed stars will very likely not change in the next millennial (only updated versions of the *GAIA* dataset will become available, which will not affect the results significantly). Once prepared, this starlight map can be used as lookup and the values per sky location can be added to the model. For this step the conversion from the logarithmic unit Magnitude per arcsecond squared to the linear nano Lambert is done via[*]:

$$B_{stars}(M[\text{Mag/arcsec}^2]) = \pi \times 10^5 \times 108400 \times 10^{-0.4M} \ [\text{nLb}] \tag{3.11}$$

The new model now consists of three parts:

$$B = B_0 + B_{stars} + B_{moon} \tag{3.12}$$

To compensate the increase in overall brightness, the scaling factor $B_{zen}$ (within expression 3.1 of $B_0$) was reduced from 79.0 to 52.0, since the star light is no longer implicitly contained in that part of the model. Further improvements to the model were made by using the extinction factor $k = 0.479$ from the photometric fit of *PyASB* and fitting the scattering function to measurements. For this, equation 3.12 was solved for $f(\rho)$, which is hidden in $B_{moon}$, and the brightness values $B(zen, az)$ were taken from data, which resulted in values for $f(\rho)$ for various separation angles, which could be collected in a binned histogram.

$$f(\rho) = \frac{B(zen, az) - B_0(zen) - B_{stars}(zen, az)}{I^*(\alpha) \times 10^{-0.4k\ X(Z_m)}[1 - 10^{-0.4k\ X(Z)}]} \tag{3.13}$$

Then the fixed Parameters of the original formula for $f(\rho)$ were replaced by variables and fitted with a least squares algorithm. Note, that the values for $\rho$ are given in units of degrees.

$$f(\rho) = 10^A[1.06 + \cos^2\rho + 10^{B-\rho/40}] \tag{3.14}$$

[*] http://hms.sternhell.at/lightwiki/index.php/Conversion_of_light_measurements

The result together with the original curve of Krisciunas et al. is shown in figure 3.4. The new scattering function shows a slightly different shape and is shifted to lower values. The minimum at roughly 90° is maintained. This represents the minimum in dipole scattering efficiency perpendicular to the incoming light. Overall the fit to the data can be described as very good, emphasizing the well chosen model functions of Krisciunas et al. No change was needed, just a refit to the differently behaving atmosphere in Namibia.

Unfortunately this means the NSB prediction tool will not be a universal model for the sky over an arbitrary location, but especially fitted to Namibian conditions. However if one would like to include other places, it would be not too much work to fit $f(\rho)$ again.

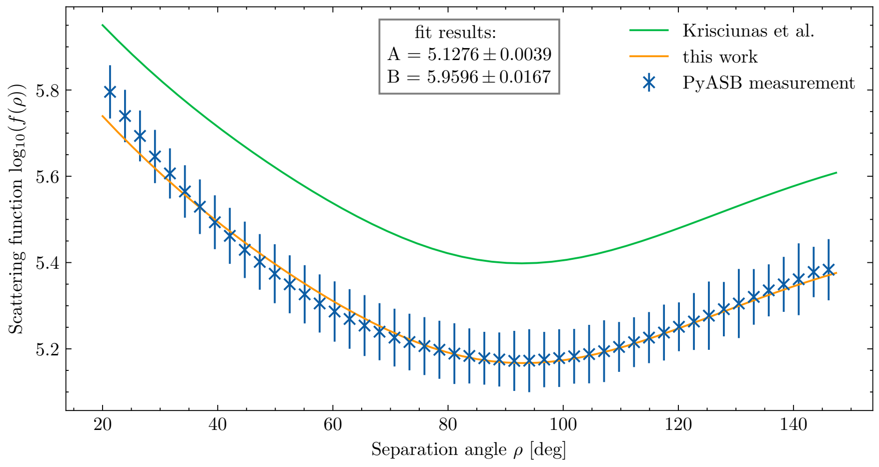


Figure 3.4.: Measurement and fit of the scattering function $f(\rho)$

The new images produced with the advanced model in figure 3.6 on page 34 show great similarity to the CCD images. To put in on the test, histograms were filled with the relative deviations $x = \frac{\text{data}-\text{model}}{0.5(\text{data}+\text{model})}$ measured on all pixels (image size 200 × 200, with roughly 75% being data points due to the circular field of view) for all 6664 images resulting in close to $2.0 \cdot 10^8$ evaluated points in the sky. This was done for the basic and advanced NSB model. The results can be seen in figure 3.5a and 3.5b. Negative values represent the model being too bright, positive values stem from the data being brighter. For better understanding the performance of the individual model parts, distributions for subsets of the data were generated. Namely for times with the Moon being above the horizon and without. A simple Gaussian function with mean $\mu$, standard deviation $\sigma$ and an amplitude $a$ was fitted to compare the distributions. All follow the normal distribution quite well. Looking at the bottom row of figure 3.5, where no moon time is considered, the background models can be compared. As expected, the basic model having only a very simple description for the background is not only too bright overall ($\mu = -0.3554$), but shows a wide spread with $\sigma = 0.3984$. The adjustment made to the advanced version of the model by the introduction

of the *GAIA* star data improved the background description clearly. This is visible in less spread ($\sigma = 0.2462$) and a shift towards the center ($\mu = -0.0359$).
Looking at the formula for the relative deviation $x$, one can calculate the relative brightness of the model $m$ with respect to the data $d$ for the case where all data is included:

$$x = \frac{d-m}{0.5(d+m)} \tag{3.15}$$

$$m(x) = \frac{2-x}{x+2}\, d \tag{3.16}$$

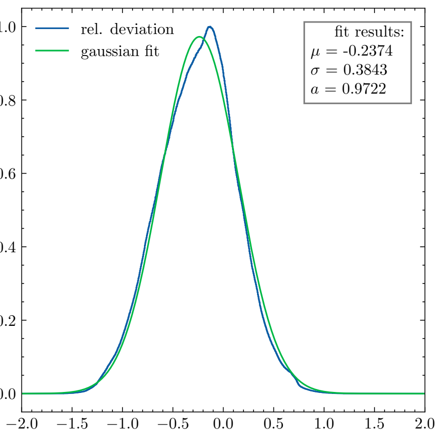
$$m(\mu) = 1.037\ d \tag{3.17}$$

and for the $\mu \pm 1\sigma$ extremes:

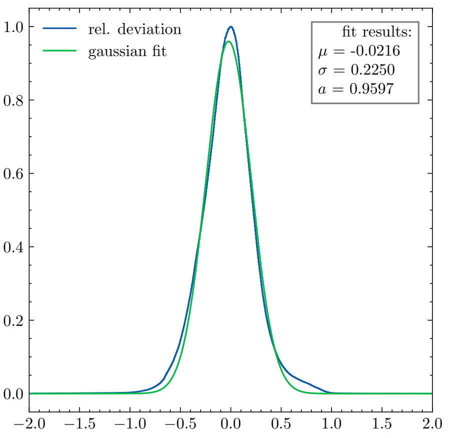
$$m(\mu+\sigma) = 0.810\ d \tag{3.18}$$

$$m(\mu-\sigma) = 1.328\ d \tag{3.19}$$

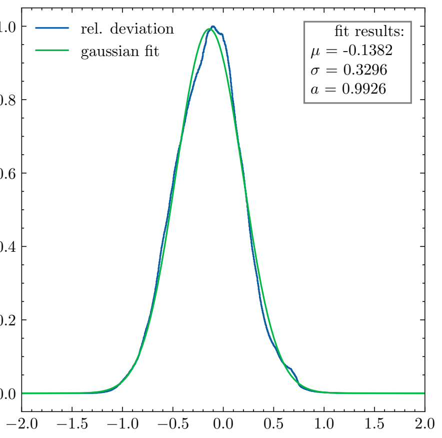
Having a small tendency of being rather too bright than too dim is not a bad thing, since the model will be used to decide whether observations with H.E.S.S. are safe to perform. This way the tendency means having more conservative estimates.
An interesting characteristic of the deviation plots is the shoulder on the positive side, which is not described by the normal distribution. Since it is basically absent in the case without moonlight, it is not related to the new background model and can most likely be explained with the lens flares and reflections that are visible in the CCD images. Those are wrongly interpreted as bright sky but not covered by the moonlight model. This explanation will be supported at the end of the following section, where the deviation plots will be shown for a model that is not based on CCD images.



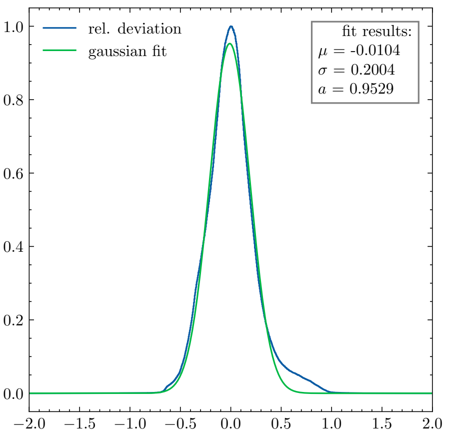
(a) Basic model all data



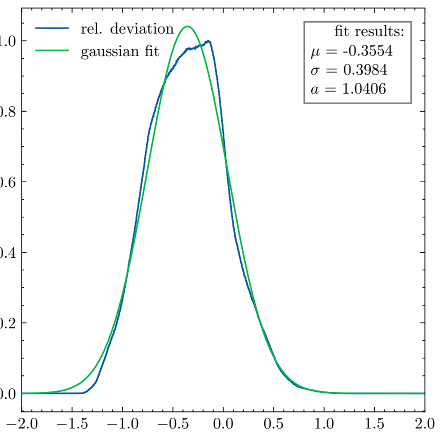
(b) Advanced model all data



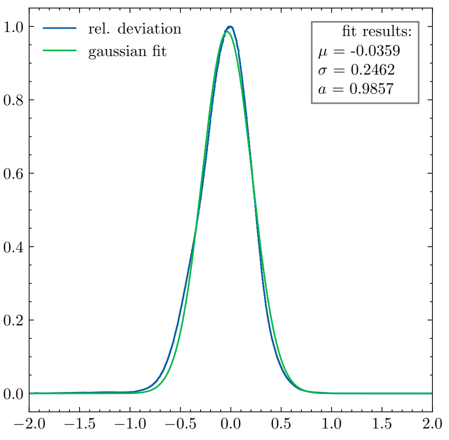
(c) Basic model only moonlight



(d) Advanced model only moonlight

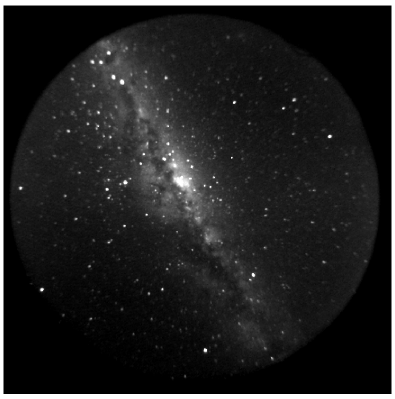


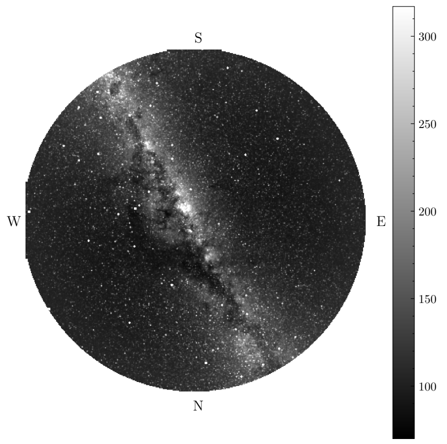
(e) Basic model no moonlight



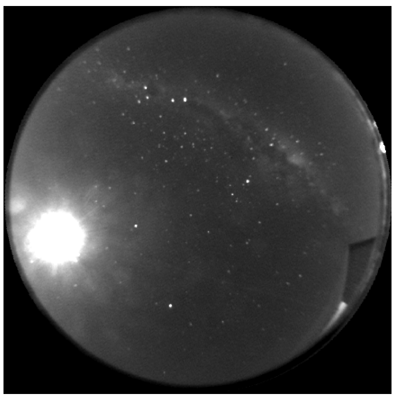
(f) Advanced model no moonlight

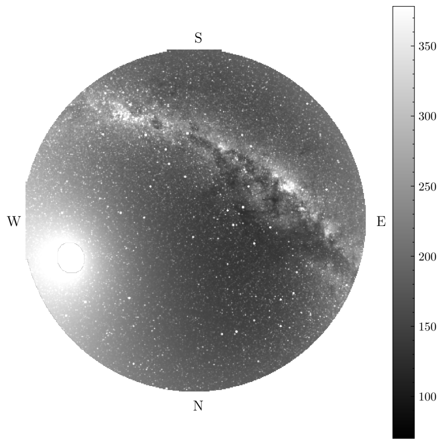
Figure 3.5.: Comparison of the basic and advanced model with the data, calculated from all 6664 images. Relative deviation is given as $\frac{\text{data}-\text{model}}{0.5(\text{data}+\text{model})}$.


(a) Image new moon

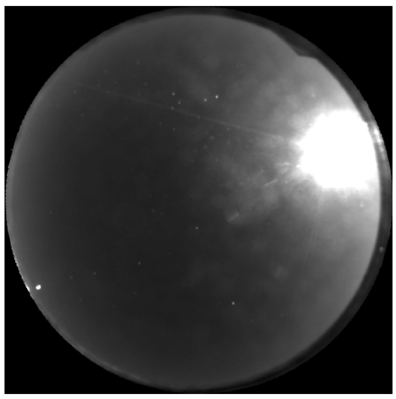


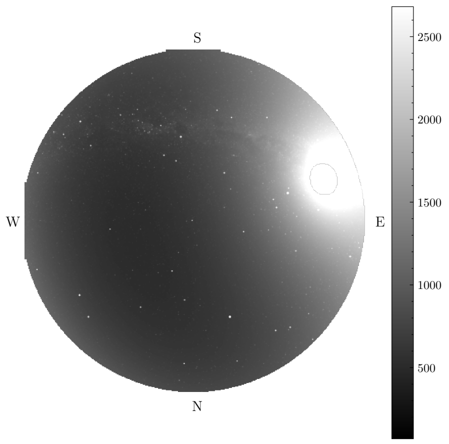
(b) Model new moon


(c) Image half moon



(d) Model half moon


(e) Image full moon



(f) Model full moon

Figure 3.6.: The advanced NSB model in comparison with the CCD camera images. For visualization the model maps were smoothed with a Gaussian kernel of radius 0.5 pixel

### 3.1.3. The currents Model

At this point in the development, the Python tool was in a state ready to be used for scheduling moonlight observations, but unfortunately the H.E.S.S. Collaboration was not (yet). Most of the features described in chapter 3.2 were already implemented. The Collaboration hesitated for multiple reasons, but the bottom line is: No moonlight observations took place for nearly 2 years after the tool and the plans for such observations were presented at the spring collaboration meeting in 2017. During that time I worked on different ways of pushing the limits of H.E.S.S. Namely on new ways of analyzing data, which can be found in chapter 4.
Only after the detection of a gamma ray burst during moonlight by the MAGIC Collaboration on January 14th 2019 [Mirzoyan et al., 2019], H.E.S.S. began to perform moonlight observations on a semi regular basis just a few days later. This gave the first true measurements of perceived brightness as seen by the telescopes and lead to the development of this new model.
The limiting factor for a H.E.S.S. camera to operate safely is the current flowing through the photomultiplier tube (PMT) anode. If this current exceeds a certain threshold, the PMT and thus the corresponding pixel can be damaged permanently. Technically this should never happen due to the presence of security mechanisms that turn off the pixel before reaching the limit, but it is always good practice to better be safe than sorry. Another important fact is the PMT aging. Photomultipliers permanently "drop gain" even under nominal circumstances, which has to be readjusted by raising the high voltage. This procedure is done on a regular basis but cannot be done indefinitely since the high voltage has a technical limited working range, which leads to a finite lifetime of such devices. In this way the illumination of the PMT directly correlates to the amount of aging and has to be kept within an acceptable range.
The new model will predict night sky background (NSB) based on measurements of the PMT currents. Since the relation between current and illumination is linear to a great extend, the new model takes the existing functional dependencies and introduces linear scaling parameters. It is then fitted to the newly acquired data.

#### The data

The H.E.S.S. cameras save monitoring data to disc with a frequency of roughly 1.0 Hz, which is stored in a separate file from the one containing the data of triggered events. From here, the values of total current drawn from the high voltage divider (HVI) for every pixel can be extracted. To get only the contribution caused by the illumination, one has to subtract the dark current on a pixel by pixel basis. This so called electronic pedestal is measured in dedicated runs several times per month by artificially triggering the camera with turned on high voltage while keeping the camera lid shut and the telescopes parked in the shelter. See figure 3.7 for details.

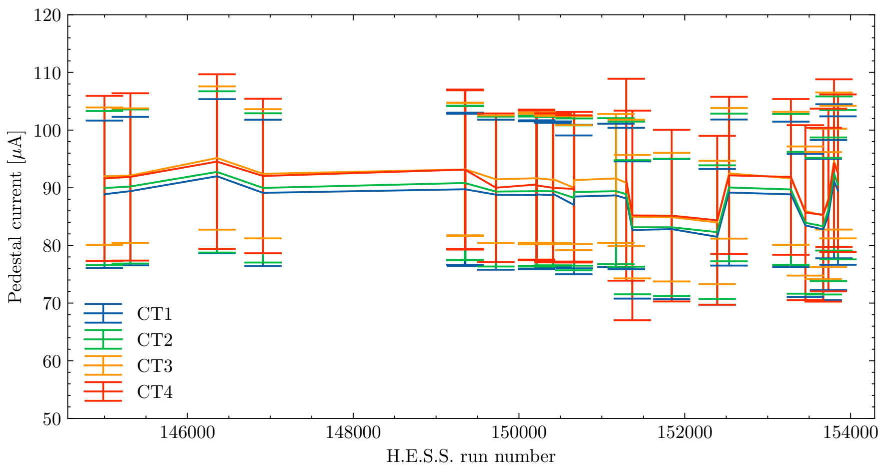


Figure 3.7.: Pedestal measurements. Calculated from raw camera monitoring data as median and standard deviation over all pixels within each camera per run.

The resulting $\Delta HVI$ (in units of µA) is directly proportional to the illumination $I$ (the photon flux measured in Hz):

$$\Delta HVI[A] = I[Hz] \cdot gain \cdot charge \tag{3.20}$$

with the nominal gain of $3.158 \cdot 10^5$ and the charge being 1 unit charge $e$

$$\Delta HVI[A] = I[\text{Hz}] \cdot 3.158 \cdot 10^5 \cdot 1.6 \cdot 10^{-19} As \tag{3.21}$$

$$I[\text{Hz}] = \Delta HVI[A] \cdot \frac{1}{5.0528 \cdot 10^{-14} As} \tag{3.22}$$

$$I[\text{MHz}] = 19.791 \frac{\text{MHz}}{\text{µA}} \cdot \Delta HVI[\text{µA}] \tag{3.23}$$

As an example, figure 3.8 shows the NSB averaged over the four telescopes in 1 minute bins for the first successful MoonlightObservationRun in the history of H.E.S.S. It was performed on the standard candle Crab Nebula with a moon phase of 61% and an angular separation of the source and the Moon in the sky of 121° in the night of 2019/01/26. The gray shade indicates the Moon being above the horizon. One can see the brightness increasing shortly before the Moon rises. This moonlight dawn will be taken into account by the new model version, since those transition times from absolute darkness to moonlight are the most crucial for extending the H.E.S.S. schedule in an easy and save approach.
As already mentioned, the requirement of extending the H.E.S.S. observation time into moonlight is to always guarantee a safe operation of the camera hardware. To get a feeling

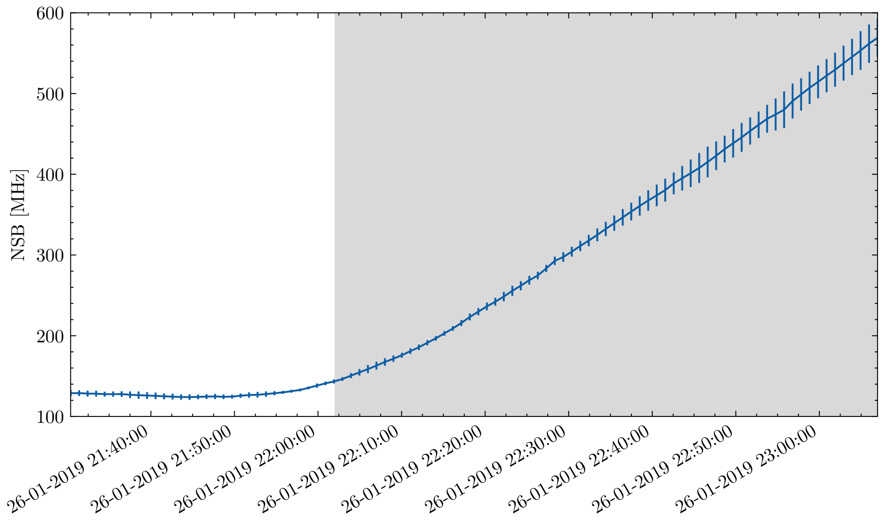


Figure 3.8.: First Moonlight Observations of Crab. NSB calculated from $\Delta HVI$ for the runs taken on Crab (145077 - 145089). Moon rises at 22:02 (gray shade). Values averaged over the four telescopes in one minute bins, error bars given as standard deviation within each bin.

for the stress put on the PMTs, the percentage of turned off pixels due to too much current drawn by them can be plotted for the same time period (figure 3.9). The observation runs were performed with nominal gain settings and only the pixel threshold for triggering events was changed on the go between the runs, to keep the rate under control. This however does not affect the data shown here. The PMTs are always operational and draw current independent of the camera read-out rate. For an extended period, the total number of affected pixels stayed at the baseline level, even after the Moon came into play. At 23:07 observations were stopped since the amount of turned off pixels reached 10% on average with single telescopes occasionally spiking over 30%.

This first successful test confirmed the collaboration's guesstimate that safe operations are indeed possible up to a NSB rate of roughly 550 MHz with nominal settings, which is in correspondence with the technical limits. The hardware limit for the total HVI per pixel is at 120 µA.

A simple back-of-the-envelope calculation shows:

With an averaged dark current of roughly 90±10 µA (eye balled from fig 3.7), the overhead will be 20-40 µA. Following Eq. 3.23 this would translate to a theoretical maximum NSB rate of 400-800 MHz, matching the observations.

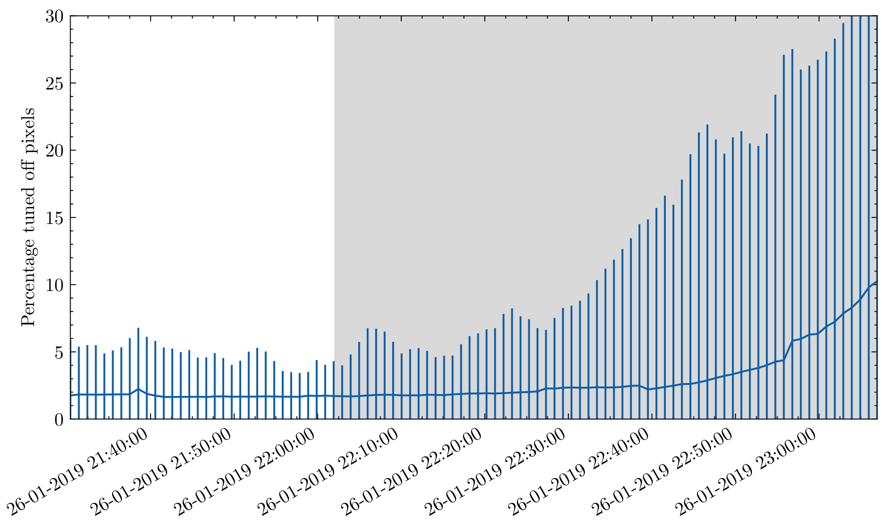


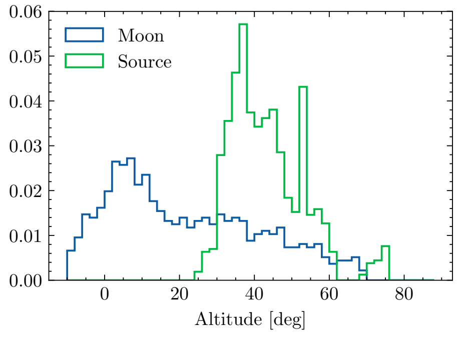
Figure 3.9.: Pixels turned off due to too much illumination for the runs taken on Crab (145077 - 145089). Moon rises at 22:02 (gray shade). Values averaged over the four telescopes in one minute bins, error bars given as standard deviation within each bin.

The following considerations determine the next steps:

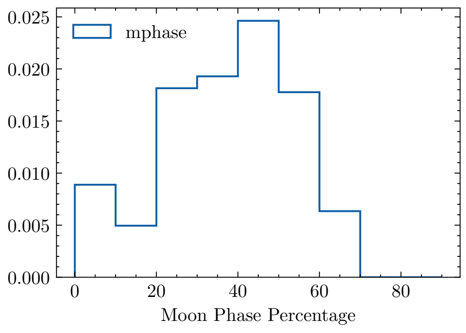
- How much observation time can be gained in safe operation conditions?
- Should the gain be lowered, to allow more NSB and thus get even more additional observation time while eventually sacrificing sensitivity? And if so, to which extent?
- What is the amount of extra time at those settings, how does it compare to regular dark time observations?

To answer those questions there was a huge demand for a reliable model to predict the currents in the cameras depending on the sky brightness. To fit the model properly, more data taken under various conditions is needed to span as much of the parameter space as possible. With the important parameters being: Moon phase, moon altitude, source altitude and moon-source angular separation. It is very hard to cover the full space with a pointing device that shall keep doing science at the same time. This means having at least one complete 28 minutes measurement run per target position in the sky and only pointing at reasonable gamma ray sources in the sky. To get the additional data which will improve the model, the collaboration understandably asked for NSB predictions of the planned measurements to not accidentally expose the camera to unnecessary harsh conditions, which in returned required a proper model and therefore turned out to be challenging.
With the allsky CCD images, on which the previous model was based on, one could get an extensive amount of combinations of source altitudes and separations out of a single image

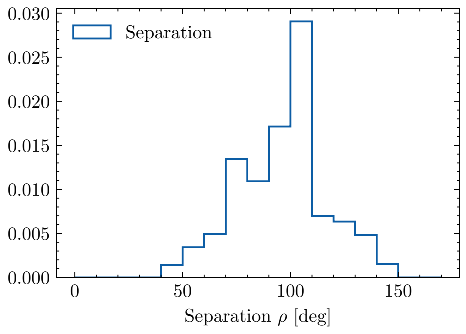
by simply picking different pixels / spots in the image. The moon altitude was naturally scanned by taking images over the whole night and the moon phase on a similar fashion by doing so for concurring nights during a complete month. And all of this without interfering with the scientific schedule of the telescopes (since they were not involved at all) and with no technical limitations on how far to go into bright nights for example around full moon.
Nevertheless a variety of moonlight runs were taken with H.E.S.S. in 2019 and the ones used in this analysis covered a fair amount of the parameter space. The full run list can be found in appendix B. It contains more than 70 runs with a total observation time of 1610 minutes (26.8 hours). The main parameter distributions are shown below (Fig. 3.10).



(a) Moon and source altitudes



(b) Moon phases



(c) Moon-source separations

Figure 3.10.: Distributions of the different parameters within the moonlight observations data set.

### Changes to the model function

For a better overview, the whole set of dependencies is summarized below. Up to now it was assumed that the spectra of the different contributions and the sensitivity curve of the PMTs are all compatible and match reasonably well, so that the total brightness can be expressed as a simple sum as in Eq. 3.12. Since the conversion from brightness units to currents in the camera is expected to be linear, a scaling factor $p_i$ is introduced for every individual part to take a potential wavelength spectra mismatch into account. The total NSB brightness will be:

$$B = p_0 \cdot B_0 + p_1 \cdot B_{moon} + p_2 \cdot B_{stars} \tag{3.24}$$

with

$$\begin{aligned}
B_0 &= 10^{-0.4k(X(zen)-1)} X(zen) && (3.25)\\
B_{moon} &= f(\rho) \cdot I^*(\alpha) \cdot [10^{-0.4k\ X(zen_m - h)} - 10^{-0.4k\ X(\pi/2)}] \cdot [1 - 10^{-0.4k\ X(zen)}] && (3.26)\\
X(zen) &= \frac{1}{\sqrt{1 - 0.96 \sin^2 zen}} && (3.27)\\
I^*(\alpha) &= 10^{-0.4(3.84 + 0.026|\alpha| + 4 \cdot 10^{-9} \alpha^4)} && (3.28)\\
f(\rho) &= 10^A [1.06 + \cos^2 \rho] + 10^{B - \rho/40} && (3.29)
\end{aligned}$$

$B_{stars}$ being the lookup created based on the *GAIA* data and $\rho$ the angular separation calculated from the sky location ($zen$, $az$) and moon position ($zen_m$, $az_m$). The extinction coefficient $k$ is free to be fitted this time as well.

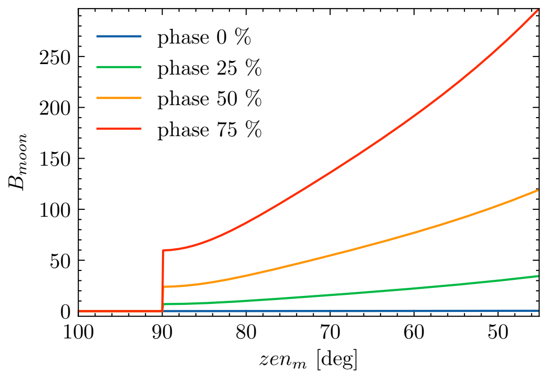
The equation for $B_0$ lost its parameter $B_{zen}$, which is now represented as $p_0$ for clarity. Furthermore, $p_1$ would be redundant if $A$ and $B$ within $f(\rho)$ are fitted at the same time.

At first this was done, but the fit did not proof to be stable. The result for $A$ showed two solutions it would fall into (see the two peaks and the exact fit results in appendix A figure 11a):

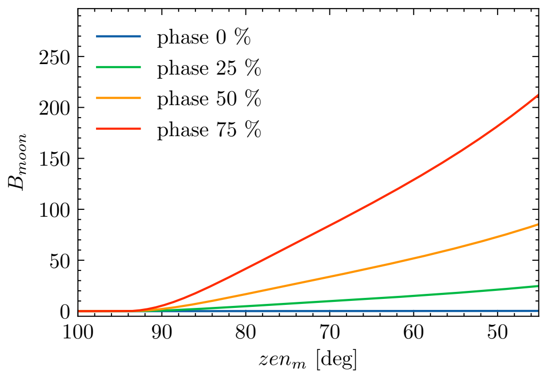
The first one had low and even negative values with huge spread and the other one had values very close to the previously determined result (first new solution: $\approx 3.3 \pm 9.6$, second new solution: $\approx 5.7 \pm 0.6$, previous result: $5.1276 \pm 0.0039$). Additionally, the newly fitted $B$ parameter was as well close to the formerly derived one (new solution: $\approx 6.1 \pm 0.6$, previous result: $5.9596 \pm 0.0167$), but with a comparably big uncertainty. This lead to the decision to keep the two parameters as fixed in favor of overall fit stability. Therefore $p_1$ will remain in the formula. Leading to the final equation with five free parameters:

$$B(p_0, p_1, p_2, k, h) = p_0 \cdot B_0(k) + p_1 \cdot B_{moon}(k, h) + p_2 \cdot B_{stars} \tag{3.30}$$

The aforementioned extension to better handle moon dawn and dusk can be seen in equation 3.26. The introduced moon horizon correction factor $h$, which artificially shifts the Moon above the horizon even though physically it is not there yet, allows to keep the trigonometric functions as they are, while being able to deal with zenith distances of more than 90°. Furthermore the base model had the flaw of not being continuous when the Moon passes the horizon. $B_{moon}$ does return values even with the Moon having a zenith distance greater than 90°, so the tool was set to return zero if $(zen_m - h) > \pi/2$, which effectively switched off the Moon below the horizon, but did not give a satisfying transition. The extra term $-10^{-0.4k\ X(\pi/2)}$ was introduced to subtract the baseline. Now $B_{moon}$ will transition smoothly during moon rise and set. The difference in the functional behavior is visualized in figure 3.11. A potential brightness bias is compensated by the refit of the remaining parameters. The source position and moon-source angular separation are all taken to be the exact values and are not affected by the moon horizon correction $h$.



(a) Without correction terms (Eq. 3.2), the transition from the Moon being below to above the horizon is unrealistic.



(b) With the additional term (Eq. 3.26), the transition is continous.

Figure 3.11.: The changed $B_{moon}$ function with additional horizon terms for different moon phases. The observation position has been fixed to $zen = 0$ for this plot.

### The fit to data

The final fit was done combined over all runs simultaneously with the free parameters $p_0, p_1, p_2, k, h$ and the real values of $zen, zen_m, \rho, \alpha$ for every time bin and source accordingly. To test the stability, 200 fits with different initial values for the parameters were performed. Those were chosen semi-randomly each time in the following fashion: For each of the 200 fits, 50 fully random guesses (within a reasonable range in all dimensions) of the start parameters were performed and the squared errors of the model evaluated with those 50 parameter sets compared to each other. The best random guess of those 50 was taken as starting point for the actual fit of the model to the data, using the Trust Region Reflective (TRF) algorithm for least-squares optimization within the *scipy* Python package based on Branch et al. [1999]. It is an algorithm designed to avoid getting stuck at parameter space boundaries. All 200 iterations succeeded to find an optimal solution with nearly identical final loss value and the final parameters all agreeing within their error ranges, proving the fit to be stable. The parameters and errors for all iterations are then combined by summing 200 normal distributions derived from each pair of fit value $\mathbf{X}_i$ and the corresponding standard deviation $\sigma_i$ via

$$N(\mathbf{X}) = \sum_{i=0}^{200} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-0.5\frac{(\mathbf{X}-\mathbf{X}_i)^2}{\sigma_i^2}} \tag{3.31}$$

and then fitted again with a Gaussian. Those distributions for all parameters can be found in appendix A (figure 12) and showed no noticeable problems.
The combined results of the currents model are shown together with the values of the previous models in the following overview:

| Currents Model: | Advanced Model: | Krisciunas et al.: |
|---|---|---|
| $p_0 = 49.204 \pm 0.646$ | $p_0 = 52$ | $p_0 = 79$ |
| $p_1 = 0.952 \pm 0.100$ | $p_1 = 1$ | $p_1 = 1$ |
| $p_2 = 1.520 \pm 0.017$ | $p_2 = 1$ | $p_2 = 1$ |
| $k = 0.293 \pm 0.030$ | $k = 0.479$ | $k = 0.172$ |
| $h = 4.346 \pm 0.311$ | $h = 0$ | $h = 0$ |
| $A = 5.1276$ (fixed) | $A = 5.1276 \pm 0.0039$ | $A = 5.36$ |
| $B = 5.9596$ (fixed) | $B = 5.9596 \pm 0.0167$ | $B = 6.15$ |

As mentioned earlier, $A$ and $B$ have not been fitted again but were used as fixed from the advanced model and are shown here for means of completeness.

In comparison to the previous model versions, the parameters look reasonable. The moon contribution scaling $p_1$ is basically unchanged, the sky background contribution $p_0$ (formerly $B_{zen}$) is lower where the starlight contribution increases. The extinction $k$ went down as well, but still is higher than Krisciunas' original value of 0.172 and the newly introduced moon horizon correction $h$ has a reasonable value of roughly 4°. That is the altitude below the horizon at which moon dawn starts and dusk ends respectively.
To understand the parameters, it is worth looking at the Pearson correlation coefficients $\mathrm{corr}(\mathbf{X})$ calculated as:

$$\mathrm{corr}(\mathbf{X}) = \left(\mathrm{diag}(\mathrm{K}_{\mathbf{XX}})\right)^{-\frac{1}{2}} \mathrm{K}_{\mathbf{XX}} \left(\mathrm{diag}(\mathrm{K}_{\mathbf{XX}})\right)^{-\frac{1}{2}}, \tag{3.32}$$

with $\mathrm{diag}(\mathrm{K}_{\mathbf{XX}})$ being the diagonal elements of the covariance matrix $\mathrm{K}_{\mathbf{XX}}$. It reveals dependencies and redundancies of parameters. For this, the covariance matrix of a randomly selected fit result is taken. The correlation coefficients can be seen in table 3.1.

| | $p_0$ | $p_1$ | $p_2$ | $k$ | $h$ |
|---|---|---|---|---|---|
| $p_0$ | 1.000 | | | | |
| $p_1$ | -0.234 | 1.000 | | | |
| $p_2$ | -0.366 | -0.104 | 1.000 | | |
| $k$ | 0.228 | -0.994 | 0.105 | 1.000 | |
| $h$ | 0.055 | 0.893 | -0.096 | -0.879 | 1.000 |

Table 3.1.: Pearson correlation coefficients of the fit parameters

The parameters $p_0$, $p_1$ and $p_2$ are all slightly anti-correlated, which is to be expected as they are the scaling factors for the individual parts. A higher contribution of one results in a lower of another. The parameter responsible for scaling the moonlight $p_1$ is strongly connected to the extinction coefficient $k$, since the moonlight is the brightest contributor and only a higher atmospheric extinction could lower it down.
The last noticeable correlation is between the moon horizon correction $h$ and $p_1$ and $k$, but this has only technical reasons, since looking at the model formula one sees that $k$ is the most powerful parameter to correct misleading values of $h$ since $p_0$, $p_2$ are not directly connected to $h$. There is no meaningful dependency to be found in this one.
Figure 3.12 shows the comparison of the currents model to the performance of the two previously described models. Clearly the statistic is the worst in the latest model, but nevertheless the distribution is the sharpest and most centered of the three. The currents model outperforms both previous versions.

Calculating again the $\mu \pm 1\sigma$ extremes of the model $m$ with respect to the data $d$ yields

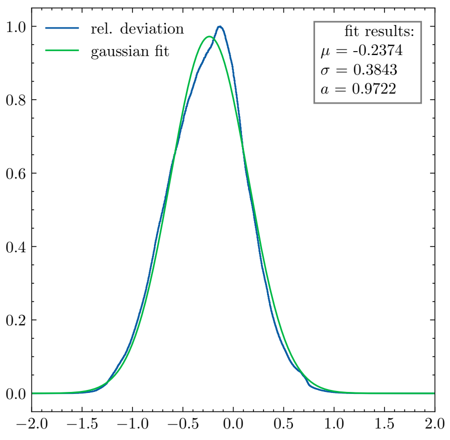
$$m(\mu + \sigma) = 0.851\ d \tag{3.33}$$

$$m(\mu) = 1.004\ d \tag{3.34}$$

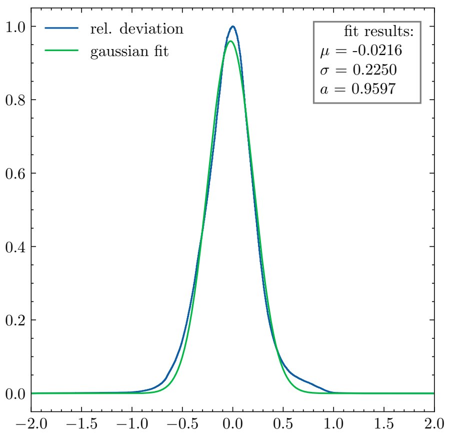
$$m(\mu - \sigma) = 1.185\ d \tag{3.35}$$

which means on average the model is only 0.4% too bright and in 68% of the cases the calculated values are within the range of not more than 15.3% too dim or 18.1 % too bright. The aforementioned prominent positive shoulder in figure 3.12b, which was assumed to stem from lens flares within the CCD images, with deviation values in the range of 0.5 to 1.0 is no longer present in the deviation plot of the currents model (Fig. 3.12c). One could argue that there is still a small feature on the positive side of the distribution (in the range 0.2 to 0.4), but since the fitted normal distribution does cover it very well and still is significantly narrower than the two previous ones, it is not considered to be a problem with the model.

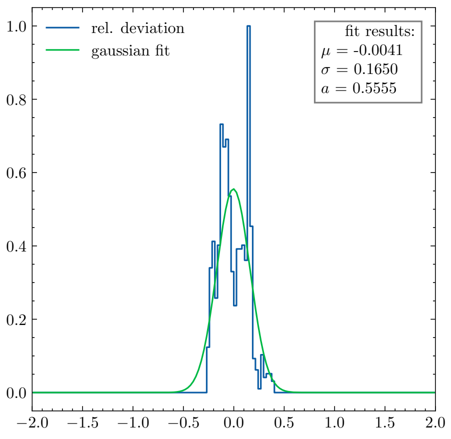
This concludes the section on model development. Starting from a simple approach, several improvements were made and the final result is a model describing the true brightness perceived by the upgraded H.E.S.S. telescopes baked into an easy to use and open source Python tool. Details on the installation and usage, including code to reproduce all results shown in this thesis can be found in section 3.4 on page 62.



(a) Basic model all data



(b) Advanced model all data



(c) Currents model all data

Figure 3.12.: Comparison of the currents model with the data and the previous models. Relative deviation is given as $\frac{\text{data}-\text{model}}{0.5(\text{data}+\text{model})}$.

## 3.2. Usage of the Model

### 3.2.1. Predicting the Sky Brightness

The model can be used to predict the night sky brightness at any given time. For example, to create a plot like figure 3.14 on page 47, showing the estimated brightness for a particular source over the course of a whole week.

The shown source, the galactic center, culminates close to zenith in that particular week and without moonlight (white background in the plot) the sky brightness only depends on the altitude angle in an inverse proportional fashion. During the week the moon time gets more relevant from night to night as the moon phase trends towards full moon and as expected, full moon is very bright and the NSB increases dramatically.

Naively one could say: *Well, if full moon is too much, let's use everything up to half moon and we're good, gaining 50% of observation time.*
But it is not that simple: Although it is true that half of a month the Moon has a phase of less than 50 %, during that two weeks it is above the horizon mostly during daytime.

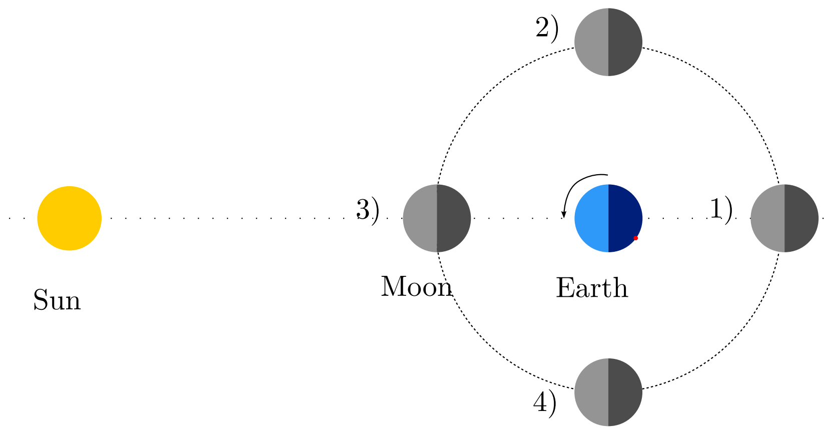


Figure 3.13.: Simplified top view of earth, moon and sun. The shading will never change, since the slow motion of the Earth around the Sun can be neglected. During a 24h day/night cycle for the observer (red dot), the moon position can also be assumed to be fixed.

See figure 3.13 and imagine the observer (red dot) looking for new moon (3). It is only visible during the day, waning moon (2) is visible in the end of the night, full moon (1) is visible the whole night. This is the reason for H.E.S.S. shifts switching personal during full moon, since no observations of dark sky are possible at all. The safest time gains at low moon phases do not make up a lot of time, where as the most could be gained in principle with very strong moon.

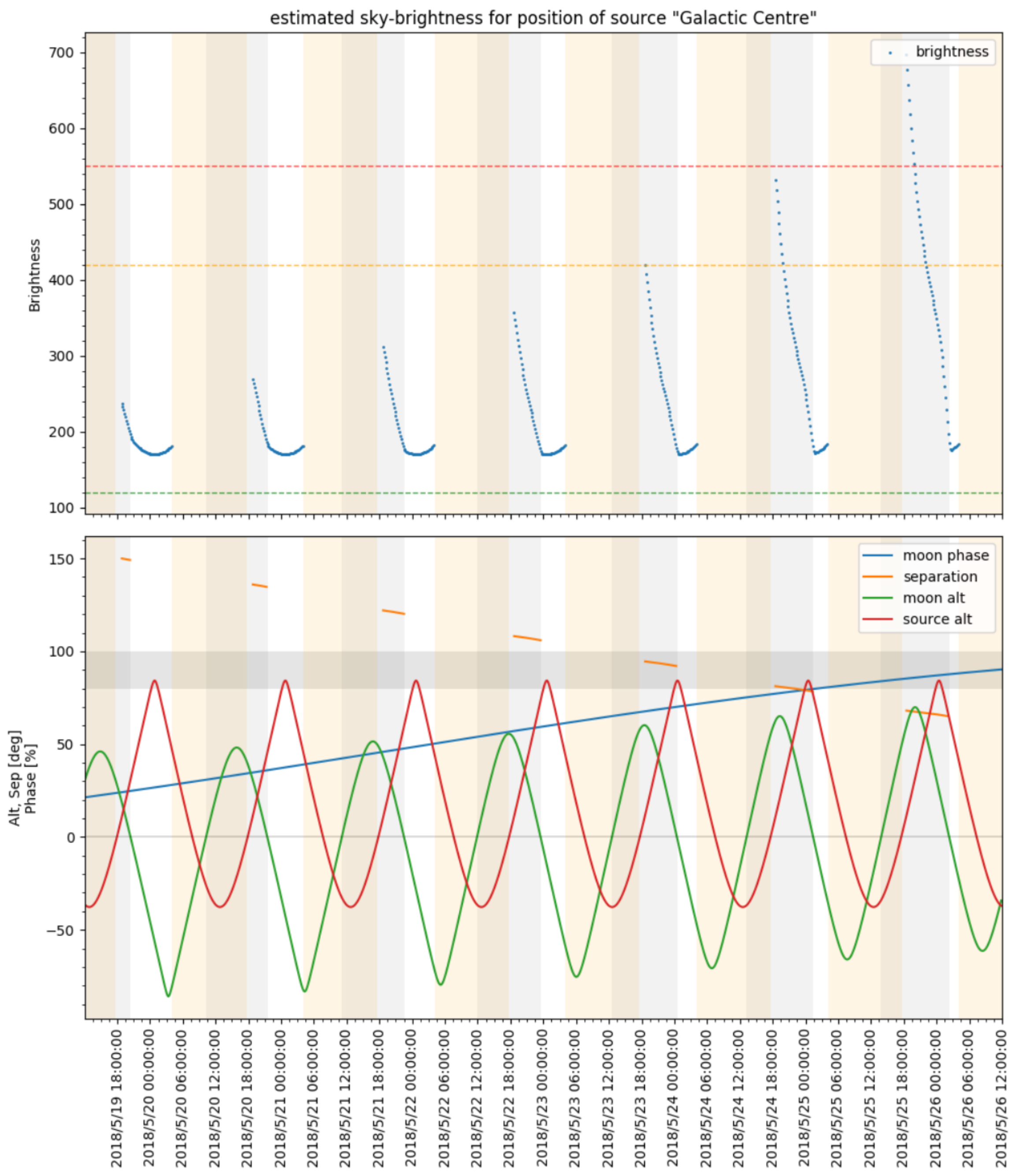


Figure 3.14.: Estimated night sky background for the Galactic Center during one week in May 2018. Upper plot: NSB together with marks for low, intermediate and very strong NSB. Lower plot: Altitude of source and moon together with the separation between both if they are above the horizon at the same time and the moon phase. Both plots: daytime is orange shaded, moontime is grey shaded. The gray band from 80 to 100 serves as a visual aid for source altitudes being close to zenith, separation being the safest (minimum of the scattering function $f(\rho)$) and moon phase being extremely high.

Even neglecting the moon phase for a moment, the amount of allowed additional background brightness for deeper exposures cannot easily be determined. Consider a dark location in the sky outside the galactic plane with very few stars in the vicinity, which happens to be visible for H.E.S.S. the whole year. Since the threshold for maximum allowed brightness the cameras can take is an absolute value, dark patches in the sky allow more NSB from moonlight until they are too bright. Technically, many additional hours of observation time would be possible on such a location, but practically if there is no scientific relevant target, this extra data will be mostly useless for regular source monitoring. For getting the reader into picture, following summary contains rough numbers on NSB strength, which are also shown as green/orange/red dashed lines in Fig. 3.14:

100 MHz very dark, extra galactic
: e.g. PKS 2155-304 without moon (see appendix A Fig. 10 on p. 117)

∼120 MHz normal background (green),
: e.g. Crab Nebula without moon (see Fig. 3.8 on p. 37)

∼420 MHz high background (orange),
: e.g. galactic plane or Eta Carinae region without moon (see appendix A Fig. 9c on p. 116)

>550 MHz too much light (red), pixels turn off,
: e.g. Crab Nebula under 60 % moonlight (see Fig. 3.8 on p. 37)

For the following calculations, the sky was divided in HEALPix with a resolution index of 6. This gives 49152 pixels over the sphere with a mean spacing of 0.916° and was chosen as a compromise between computation time (every resolution index higher quadruples the amount of pixels and with it the computation load) and resolution compared to the field of view of the H.E.S.S. telescopes (5°). A dummy source was placed at every pixel center an then tracked by the NSB Python tool for 365 days with time steps of 20 minutes to calculate the predicted brightness with the latest NSB model at that location and time, resulting in $(365 \cdot 24 \cdot 3 =)26280 \cdot 49152 \approx 1.3 \times 10^9$ data points. If a given source is not visible at a certain time (below the horizon or lower than the minimum required altitude to be observed), the entry will be NaN (Not a Number, which can be correctly identified by Python later). This was done for 6 scenarios: allowing and disallowing the Moon to be above the horizon for minimum source altitudes of 0°, 30° and 60°. Now the amount of valid entries per pixel smaller then a given limit $x$ can be counted for each scenario and interpreted as time bins this HEALPix coordinate can be observed by H.E.S.S. Since the steps were fixed at 20 minute intervals and calculated for a complete year, a simple division by 3 converts to units of hours per year in which that spot has a brightness lower than $x$. Subtracting the

corresponding data maps obtained with moonlight allowed from those without, yields a map with additional hours per year.
Figures 3.15 and 3.16 show the resulting data with $alt_{min} = 30°$. The limit of 250 MHz was chosen arbitrary in this figure. To understand the two big holes in the map, the data is shown in both galactic and equatorial coordinate systems. For many people, the first system makes it easier to locate prominent sources but harder to imagine the reason for the missing data points. In equatorial projection their origin is obvious: The big northern blind spot corresponds to coordinates in the sky sphere, where the H.E.S.S. telescopes can never point to, due to their location on the southern hemisphere on earth. The second and smaller gap stems from the fact, that indeed some points in the sky are too far south. Only lowering the minimum required source altitude would enable observations in that direction if the telescopes point close to the horizon.
The map shows time in hours which could be gained from moonlight observations, if the maximum brightness the cameras are allowed to take is set to 250 MHz. Dark sky regions like in the vicinity of PKS 2155-304 or the Large Magellanic Cloud profit the most, where i.e. the galactic center or Eta Carinae regions do only get very little extra time ore none at all since they are already very bright without moonlight. The small sprinkles without data in this map have to be ignored. They come from individual stars, which are so bright that they boost the average sky brightness in a particular HEALPix above the threshold. In reality such a star in the field of view of a H.E.S.S. camera would only illuminate a very small number of pixels, which in response would be turned off, which does not interfere with data taking.

## 3.2.2. Predicting the Time Gain through Moonlight Observations

Coming back to the question how much the collaboration will gain in observation time: The next plausible answer would be to look at a similar plot like 3.15 but done for the maximum brightness the cameras can take (550 MHz) and take the maximum in the map. This map can be found in figure 2 in appendix A and the answer would be: We gain at most 715 additional hours/year with moonlight. Pointing the telescopes somewhere in between PKS 2155-304 and the Large Magellanic Cloud all year long...
But as already mentioned this is neither an option, nor the truth. Always pointing at that single location does not make any scientific sense and there might be (and indeed are) periods during the year where that particular spot is not visible during darkness. With a specially optimized pointing schedule strategy the total amount of hours could in principle be improved even more, by looking at the next best location when the first one is not available etc. However all of this would most probably still not be relevant to the H.E.S.S. collaboration, which pursues to observe $\gamma$-ray sources and not only random spots. Although

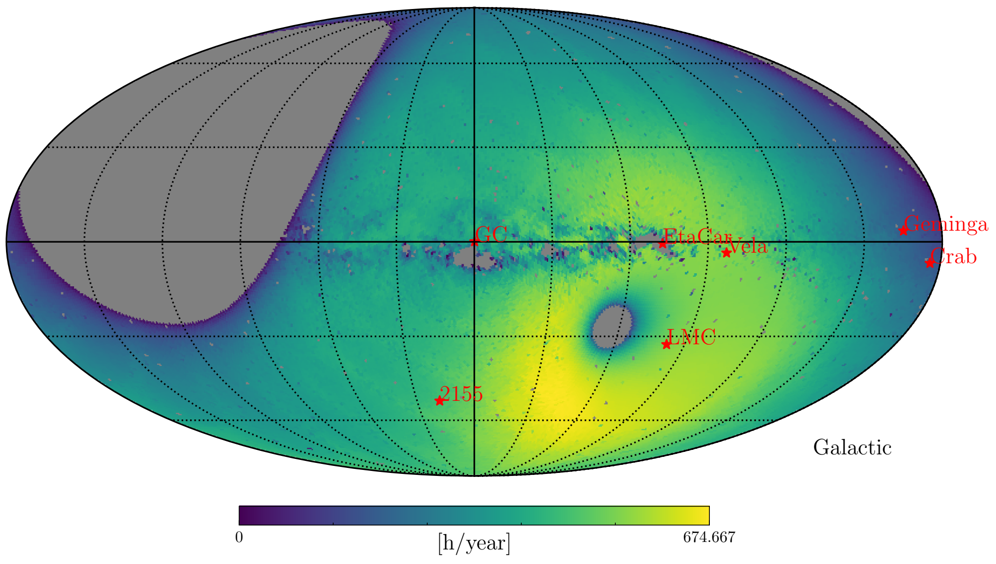


Figure 3.15.: The potential extra time to be gained per year for every HEALPix, when setting the brightness limit to 250 MHz, the minimum source altitude to 30° and allowing the Moon to be above the horizon.

Targets of Opportunity (ToO) are getting more and more important in the late phase of the H.E.S.S. experiment, but since they are not predictable when and where to happen, another method for a realistic estimate of the time gain is chosen.
For this, a typical H.E.S.S. observation schedule is considered. It is obtained by collecting all coordinates of successful data runs over several past years from the collaboration database and averaging the time spent on each. The result is an exposure map with mean hours taken per HEALPix. The next step is to calculate the available dark time. This is done using the already generated maps but without limits and excluding moon time. The corresponding map is shown in figure 3 in appendix A.
Dividing the exposure data by the available dark time, one ends up with a map which will be called *sky usage* (Fig. 3.17). It can be understood as: "Out of all the time a certain location in the sky was visible to H.E.S.S., what percentage of that time did the telescopes typically point there". Many prominent targets can be identified in this map and are labeled in the corresponding figure.

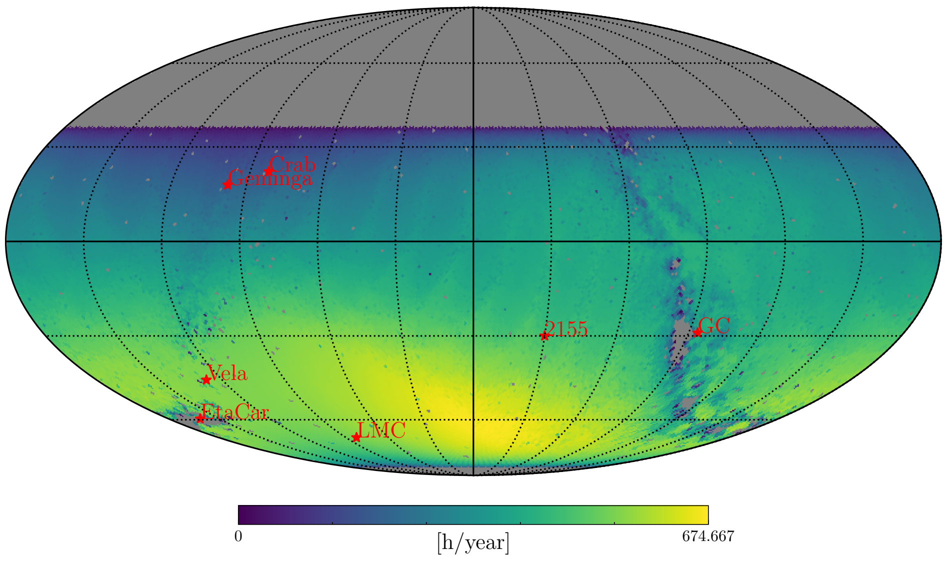


Figure 3.16.: Same as figure 3.15, but in equatorial map projection.

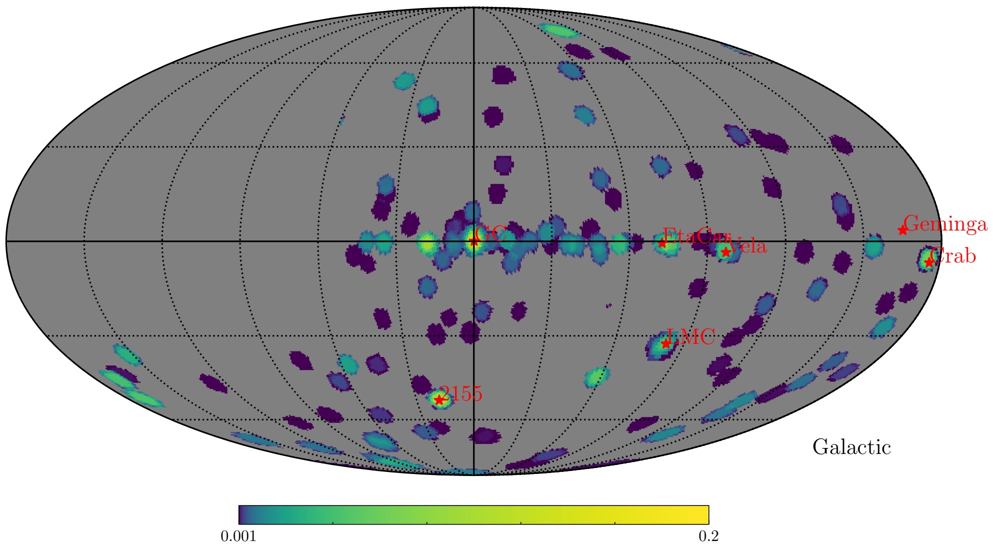


Figure 3.17.: Sky usage map giving the fraction of observation time which was on average used out of all available time per HEALPix.

This *sky usage* can now be used as a mask for the allsky time gain map at different limits. The observation times per year for moonlight and dark conditions at that specific brightness limit $x$ can be calculated as:

$$T_m(x) = \frac{1}{A_{fov}} \sum_{pixel} \left[ \mathbf{T}_m(x) \odot \left( \frac{\mathbf{E}}{\lim_{x\to\infty} \mathbf{T}_d(x)} \right) \right] \tag{3.36}$$

$$T_d(x) = \frac{1}{A_{fov}} \sum_{pixel} \left[ \mathbf{T}_d(x) \odot \left( \frac{\mathbf{E}}{\lim_{x\to\infty} \mathbf{T}_d(x)} \right) \right] \tag{3.37}$$

where $\mathbf{T}_m(x)$ is the map of moon time (excluding regular dark time), which is available per spot allowing a maximum NSB limit of $x$ MHz, $\mathbf{T}_d(x)$ is the similar map for only dark time, $\mathbf{E}$ is the exposure map and $\odot$ denotes the element wise product of two maps. The summation is performed over all pixels with normalization $A_{fov}$ which is the area of the telescope field of view expressed in HEALPix. In the case of resolution index 6, it is $A_{fov} = 33.7$.

The total observation time consisting of regular dark observations without limit (denoted with $d'$) and moonlight observations up to the limit $x$, can be written as:

$$T_{d'+m}(x) := \lim_{x\to\infty} T_d(x) + T_m(x) \tag{3.38}$$

The total dark observation time without limits is constant and will therefore be replaced with

$$\lim_{x\to\infty} T_d(x) = const := T_{d\infty} \tag{3.39}$$

Yielding

$$T_{d'+m}(x) = T_{d\infty} + T_m(x) \tag{3.40}$$

Calculating the moon and dark time maps for many limits, the time gain data and other implications can be obtained and will be discussed in the following.

Looking at the low NSB end of the time gain plot (Fig. 3.18), one can see that there is nothing to be observed at 50 MHz, as what is usually considered very dark sky still has a background brightness of roughly 100 MHz. Without moonlight (dark time $T_d$, blue line) the brightest regions in the night sky have a NSB of about 400 MHz and the total observation time reaches a plateau of 730 hours/year (blue dashed line) at this point. Allowing higher limits does not gain any more time, since everything is already covered. But there are many possibilities to observe $\gamma$-ray sources up to that limit of 400 MHz when including moon time, resulting in additional $T_m(400) = 605.1$ hours/year (green line). The direct conclusion would be to set the limit here, since it would not require any major changes to the analysis chain.

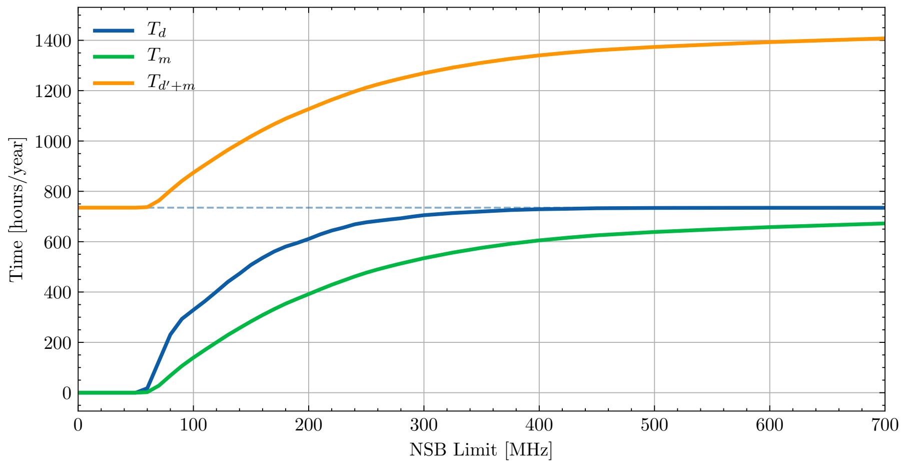


Figure 3.18.: Absolute dark and moon time per year for a typical H.E.S.S. schedule depending on the allowed NSB limit.

Although analyzing the Eta Carina Region with roughly 420 MHz NSB proved to be difficult [H.E.S.S. Collaboration et al., 2020], this was mainly due to strong lateral variability in the background and not solely due to the high values per se.

Potentially stopping the H.E.S.S. collaboration from straight going for regular moonlight observations up to that limit is the reduction in camera lifetime due to accelerated PMT aging under brighter illumination. PMTs degrade over time proportionally to the total accumulated charge, which is directly proportional to the integrated brightness over time. In parallel to calculating the time gain maps, integrated brightness maps were produced, where every HEALPix holds the information on how much total illumination the camera is exposed to, if looking at that spot for the whole year. This map was then masked and summarized similar to the time gain maps. Equation 3.23 which was used to calculate the background flux $I$ from the excess PMT current $\Delta HVI$ can now be used to convert back to the absolute charge $Q$ collected per PMT during a year:

$$\Delta HVI = I[\text{Hz}] \cdot 3.158 \cdot 10^5 \cdot 1.6 \cdot 10^{-19} As \tag{3.41}$$

$$\Delta HVI = I[\text{MHz}] \cdot 5.0528 \cdot 10^{-8} As$$

$$Q = 5.0528 \cdot 10^{-8} As \int I[\text{MHz}] dt \tag{3.42}$$

$$Q = 5.0528 \cdot 10^{-8} As \cdot \Delta t \sum I[\text{MHz}] \tag{3.43}$$

Since the brightness is calculated in discrete time steps of $\Delta t = 20$ minutes, where it can be considered constant, the integration over time can be replaced by a discrete sum multiplied with the time interval. This can again be done depending on the maximum allowed brightness limit and for dark and moon conditions, which is presented in fig 3.19.

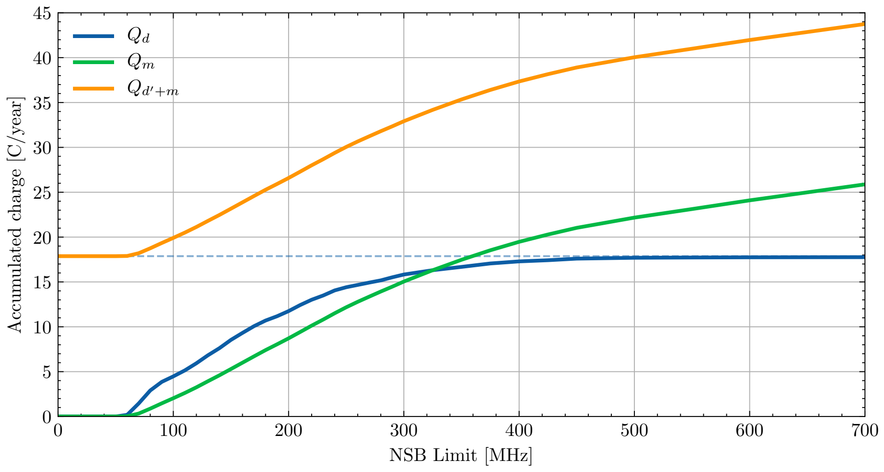


Figure 3.19.: Absolute charge accumulation per year for a typical H.E.S.S. schedule depending on the allowed NSB limit.

At roughly 350 MHz the additional charge on the PMTs from moonlight observations (green line) reaches the same value as regular dark observations without limit (blue dashed line) and therefore would cut the lifetime into half. To better visualize this, one can calculate the relative lifetime $\tau$ of the PMTs with respect to the NSB limit $x$ compared to observations without moonlight as the usually accumulated current in darkness ($Q_{d\infty}$, read as "charge dark at infinite limit") divided by the usual total amount plus the extra charge due to moonlight depending on the limit ($Q_m(x)$):

$$\tau(x) = \frac{Q_{d\infty}}{Q_{d\infty} + Q_m(x)} \tag{3.44}$$

Equation 3.44 is displayed in figure 3.20. It is no surprise that moonlight observations will reduce the PMT lifetime. The question is: Should one do it anyway? At which cost?
Using $\tau(x)$, one can convert the time gain into a *lifetime corrected* version, which accounts for the lost time in PMT life, if one would stress them regularly with a certain amount of additional moonlight until the end of their lifetime. It will be defined as the difference of potential total time including moonlight observations up to the limit ($T_{d'+m}(x)$), which is scaled by the relative lifetime ($\tau(x)$), and the regular time per year for dark observations without limit ($T_{d\infty}$):

$$T^*_m(x) = \tau(x) \cdot T_{d'+m}(x) - T_{d\infty} \tag{3.45}$$

$$T^*_m(x) = \tau(x) \cdot (T_{d\infty} + T_m(x)) - T_{d\infty} \tag{3.46}$$

The implications will be discussed in the following section.

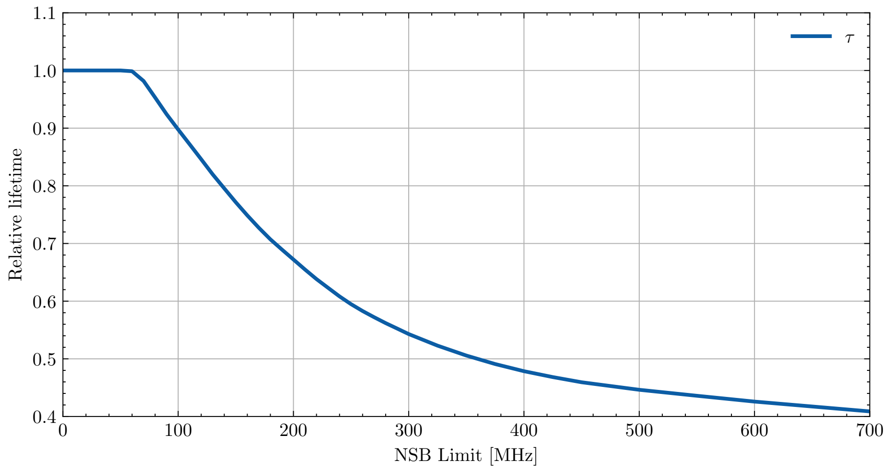


Figure 3.20.: Relative PMT lifetime with respect to moon less observations depending on the amount of allowed additional moonlight observations.

## 3.3. Conclusions on Moonlight Observations

### 3.3.1. Nominal PMT Gain

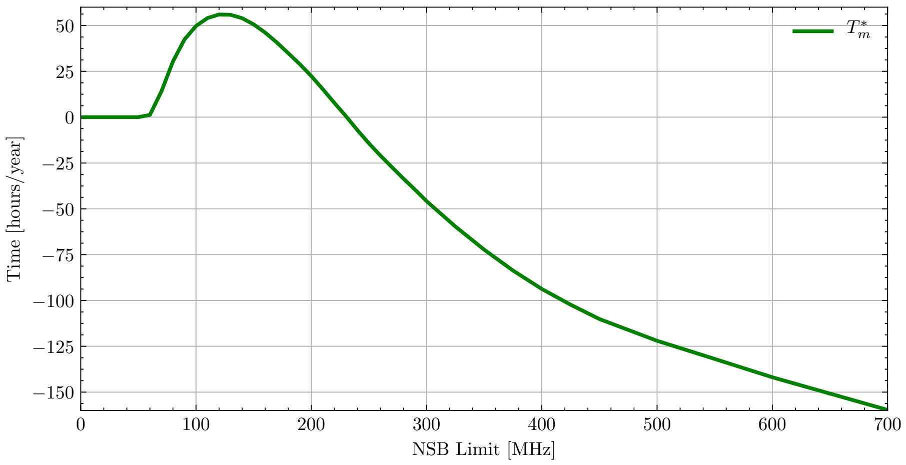


Figure 3.21.: Lifetime corrected time gain from moonlight runs as function of the maximum NSB brightness.

Up to ≈230 MHz the net time gain is positive with a maximum at 125 MHz. Within this window, one does gain more hours per year with moonlight observations than those put stress on the PMTs. If the PMT supply is limited, i.e. they will not be exchanged, and one has to get as much of time out of them as possible, moonlight observations up to 125 MHz are the way to go. This would already mean 200 additional hours (400 data runs) in that

particular year, which translates to an extra of 56 hours (112 data runs) if accounting for the lifetime reduction (see Fig. 3.18 on page 53, or table 3 in appendix B with all results)
If the PMT's end of life is far enough away and one can afford shortening it by 40 % or to buy new ones, then moonlight observations up to 230 MHz are the most efficient use of the hardware. Yes, they have to be replaced earlier than without moonlight observations, but thanks to those extra data runs, the same amount of total hours of data is already on disk after just 60% of the time it would have taken normally.
Observations beyond that brightness should not be considered for regular scheduling, as they degrade the cameras too heavily, Targets of Opportunity however should definitely always be observed as long as they fall bellow the technical limit of 550 MHz (see the runs taken on the Crab Nebula in January 2019, figure 3.8).

### 3.3.2. Reduced PMT Gain

This previous discussion was based on the assumption that the PMT gain will be held unchanged for both regular and moonlight observations at the nominal value of 80 ADC/p.e. The picture looks different if the gain is allowed to be altered:

- The current in each PMT will change linear with the gain.
- The total remaining lifetime will change non-linear, since the relation between gain (current) and the high voltage setting that produces that gain is non-linear, but the lifetime depends on the gap between the momentary HV setting and the maximum allowed HV for each pixel.
- NSB maps and the model will stay unchanged. Although the night sky background rate data in MHz, which is the foundation of the model, is calculated from the pixel currents, the results will remain consistent (The sky brightness does not care what setting the cameras are in).
- The calculations of observable time with respect to certain limits will not be changed, *but*
- The possible/allowed limits will change
- The relative PMT lifetime $\tau$ will change, since it depends on the charge collected during moontime runs ($Q_m(x)$).
- Thus, the lifetime corrected time gain $T^*_m(x)$ will change.

To which extent a gain reduction makes sense will depend on the ability of the calibration chain to still be able to resolve single photo electron signals and the tolerated shift in energy threshold. A camera with lower gain will not trigger on faint images which directly translates into a worse effective area at the low energy end of the spectrum.

The in-depth calculations of those effects based on Monte Carlo simulations were out of the scope of this work, but will certainly be pursued by H.E.S.S. research colleagues.

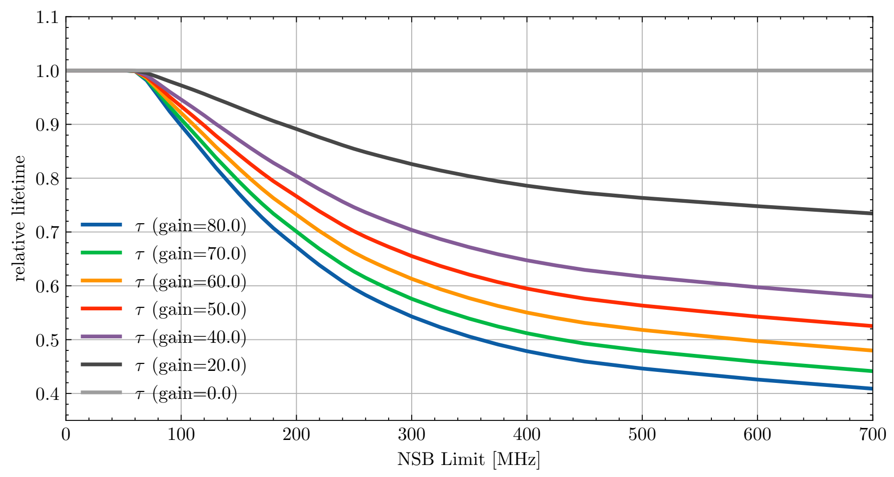


Figure 3.22.: Relative PMT lifetime for different PMT gain settings.

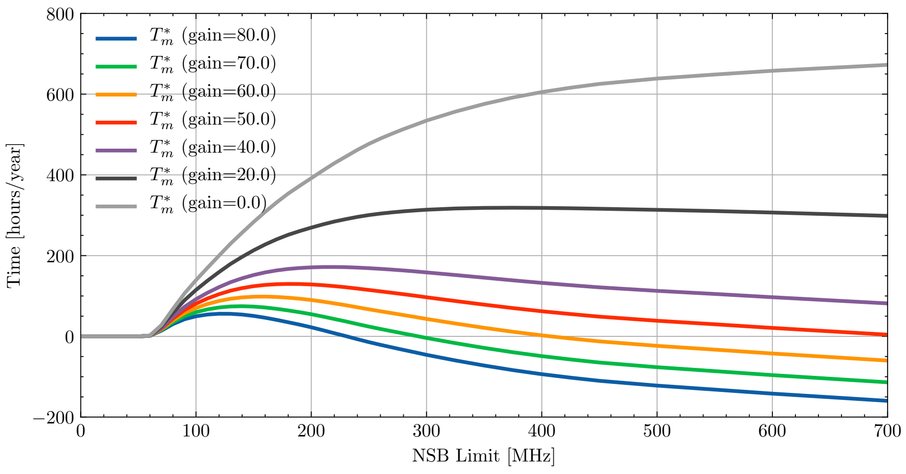


Figure 3.23.: Lifetime corrected time gain for different PMT gain settings

Figures 3.22 and 3.23 show plots of relative lifetime and lifetime-corrected time gain for different HV gain settings. It is obvious that the relative PMT lifetime will only get better and reaches a constant behavior of $\tau = 1.0$ in the extreme case of zero gain for moonlight runs. In the end, the cameras are not stressed at all if they are basically turned off and one could point them anywhere and as long as desired. In this theoretical corner case, the lifetime corrected time gain $T_m^*(x)|_{gain=0}$ (gray line) is equal to the non corrected time gain $T_m(x)$ (green line in Fig. 3.18).

Below a HV gain value of about 40 ADC/p.e. there is no negative lifetime-corrected time gain any more. Moonlight observations would always bring more hours than they cost the PMTs lifetime.

The two NSB limit recommendations for most effective time gain (maximum of $T^*_m(x)$) and best compromise (second zero passing of $T^*_m(x)$) can now be calculated per PMT gain to develop limit strategies and are presented in figure 3.24. Due to the numeric approach of calculating the time gain values for binned values of limits, the maxima were not always properly covered and are therefore derived from parabolic approximations of a few data points around the numeric extreme points of $T^*_m(x)$. In a similar fashion, the second zero passing was calculated from a linear fit in the corresponding region.

The technical observable NSB limit in MHz depends on the difference in current between hardware limit ($HVI_{max} = 120$µA) and dark Pedestal current $HVI_{ped}$:

$$\Delta HVI = HVI_{max} - HVI_{ped} \tag{3.47}$$

And is calculated via equation (3.23):

$$I[\text{MHz}] = 19.791\,\tfrac{\text{MHz}}{\text{µA}} \cdot \Delta HVI[\text{µA}]$$

yielding the following relation

$$I[\text{MHz}] = (HVI_{max}[\text{µA}] - HVI_{ped}[\text{µA}]) \cdot 19.791\,\tfrac{\text{MHz}}{\text{µA}} \tag{3.48}$$

The changed gain can be expressed with a factor $g \in [0,1]$ and will affect the equation in the following way:

$$I[\text{MHz}] = (HVI_{max}[\text{µA}] - g \cdot HVI_{ped}[\text{µA}]) \cdot \frac{1}{g} 19.791\,\tfrac{\text{MHz}}{\text{µA}} \tag{3.49}$$

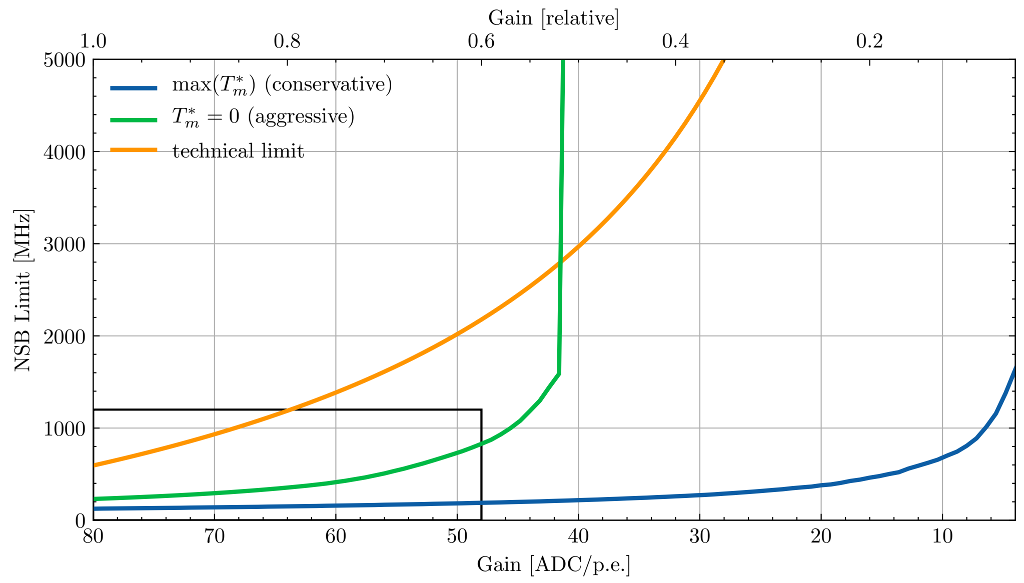
The pedestal current will decrease with lowering the gain, whereas the scaling factor will increase, e.g. with half the gain: 39.582 MHz NSB would induce 1 µA of current instead of the previous 19.791 MHz. The hardware limit will stay the same. This can be rewritten to be

$$I[\text{MHz}] = (\frac{1}{g} HVI_{max}[\text{µA}] - HVI_{ped}[\text{µA}]) \cdot 19.791\,\tfrac{\text{MHz}}{\text{µA}} \tag{3.50}$$

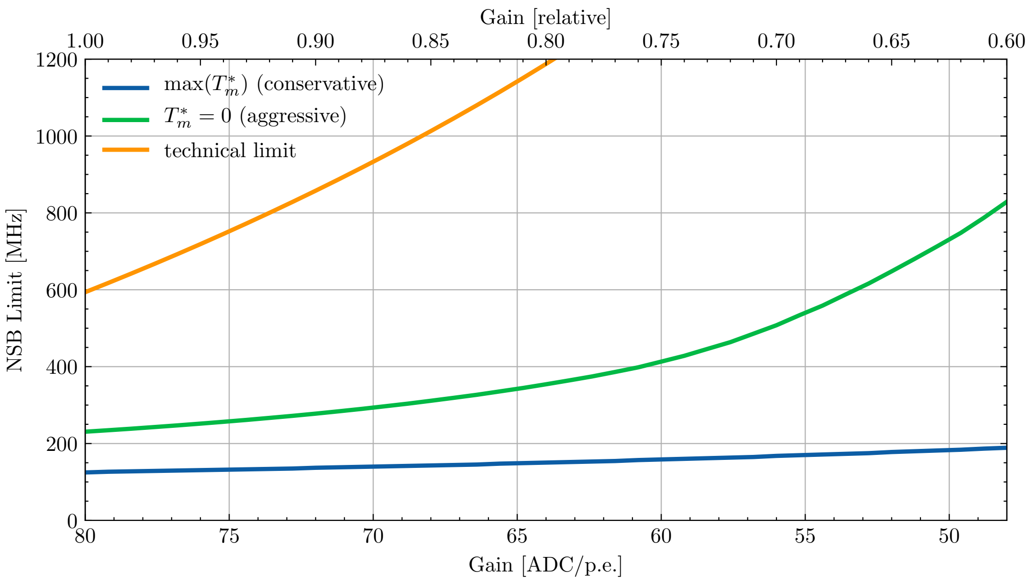
All the three limit strategies in terms of gain are shown in Fig. 3.24.

The blue line of $\max(T^*_m)$ again describing the conservative save optimum of the lifetime reduced time gain stays relatively low over a long range of gain values. A strong increase is only observed for very low gain factors, where the PMTs are practically turned off.
The picture is different for the aggressive optimum ($T^*_m = 0$, green) which describes the situation, where the same amount of data will be taken as without moonlight observations but in a much shorter time. It is the balance between extra hours and lifetime reduction. This limit increases rapidly and approaches a singularity at the point where the extra hours are always greater than the PMT stress, which happens at a gain of 42 ADC/p.e. The technical limit as well very rapidly shifts to high NSB values, since the dark pedestal current no longer occupies the vast amount of the PMT's working range.
Those three strategies of setting a NSB limit can now be used to look up the corresponding time gains per year (absolut and corrected). They are presented in figure 3.25a.
The conservative approach has the least absolute gain, but by design maximizes the lifetime corrected time gain. The aggressive one (balance between gain and aging, lifetime corrected time gain per definition zero) seems the most applicable, as it yields enormous absolute time gains per year in the range of 450 to over 650 hours per year for the shown gain reductions. Going to the technical limit results as expected in the highest absolute time gains per year, but puts maximum stress on the PMTs reducing their lifetime significantly. This can be seen in the negative values of figure 3.25b.

Independent of the PMT gain reduction the H.E.S.S. collaboration decides on, my recommendation would be to go for the balanced aggressive approach, which effectively speeds up the absolute data taking every month and year without sacrificing the available PMTs.

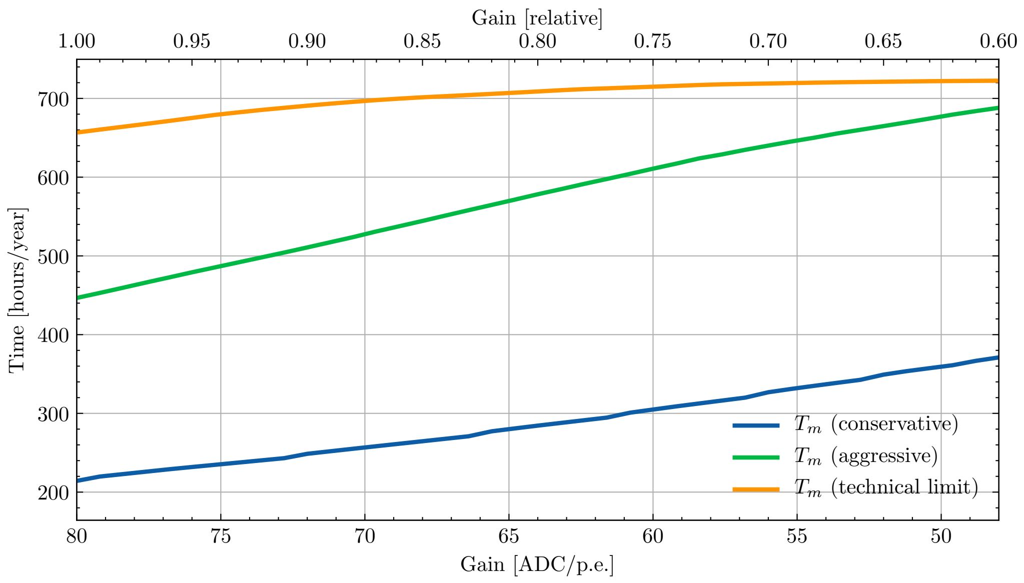


(a) Overview

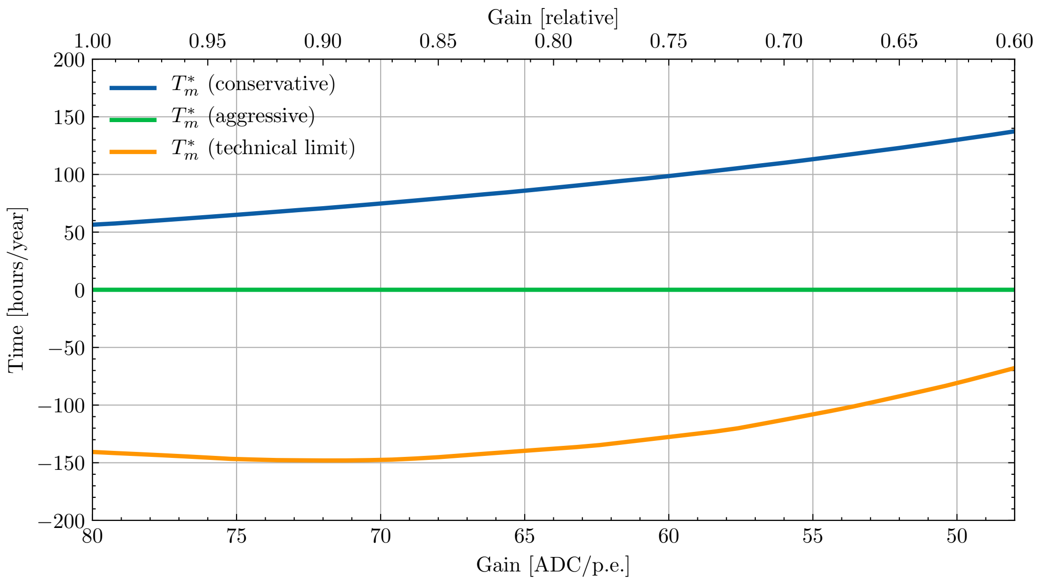


(b) Zoomed in

Figure 3.24.: NSB limit strategies based on selected PMT gain.



(a) Absolute time gain for the three strategies



(b) Lifetime corrected time gain for the three strategies

Figure 3.25.: Time gain per year for the three limit strategies (absolut and litetime corrected)

## 3.4. Implementation Details

This Section will not provide any new results, but show the details on how the previous ones were obtained. The less technical versed reader may skip over to the Deep Learning Chapter 4 on page 75.

### 3.4.1. Basic Usage of the NSB Python Tool

A major part of the work for this thesis went into the development of code. The source for the NSB Python tool is openly available and its working principles have been discussed extensively in the first sections of this chapter. Therefore no further source code explanations will be given. However, the usage of the tool will now be explained in greater detail with the hope to set the foundation for new work that can build on top.

The reader is encouraged to download, try and use the tool for them self. Macintosh and a vast majority of Unix operating systems are supported. Unfortunately Microsoft Windows is not, due to a non compatible dependency (pyEphem).
To get started, simply open a terminal shell and install the tool with:

```
(sudo) pip install nsb
```

The most recent version during the writing of this thesis was 0.5.1. Basic examples can be found directly at the *Python Package Index* website https://pypi.org/project/nsb/, where the module is published. In the following, the necessary additional code based on the tool to reproduce the results from the previous sections will be presented and discussed.

The easiest way to interact with the tool is directly via the command line.

To generate the allsky images from the model based on the CCD images (figure 3.6 on page 34) the following command has been used (this example shows the case for half moon, the other dates work accordingly):

```
python -m nsb --skymap --version hess_basic --fov 150 --savefits --size 431 --time 2010/06/18
    20:01:36 --hp 8
```

It saves the result automatically as a *fits* file, which was then enhanced in contrast for this thesis and exported as *pdf*.

The trend plot in figure 3.14 at page 47 of the night sky brightness of the galactic center region can be reproduced with the following command. The observer's location will default to the H.E.S.S. site in Namibia and does not require an additional argument.

```
python -m nsb --trend --version hess_currents --t1 2018/05/19 12:00:00 --t2 2018/05/26 12:00:00 --
    source "Galactic Centre"
```

### 3.4.2. Obtaining the Time Gain Plots from Section 3.2.2

Next up are the more advanced plots, which require the Python tool being used as an imported module. To obtain the time gain results, one has to calculate the brightness of the sky for each possible location in the sky for as many time steps as possible. As mentioned in the corresponding chapter, 49152 pixels (healpix resolution 6) and 26280 time steps (365 days, 20 minute interval) were chosen.

#### The raw data

Figure 3.26 shows the Python script. After the import statements of the NSB functions, the configuration object is created by reading a text file from disc (see the example in figure 3.27). The important part for the current task are the three last settings of the horizon values. These give the angles in degrees at which an object (moon/sun/source) is considered setting or rising.

```python
 1  from nsb import config
 2  from nsb.model import nsbModel
 3  from nsb.mypycat import gammasource
 4  from nsb.gaia import Gaia
 5
 6  import numpy as np
 7  from multiprocessing import Pool
 8
 9  con = config.TheConfiguration()
10  con.readConfig("timegain_config.cfg")
11
12  time_and_date_1 = ephem.Date("2019/01/01 00:00:00")
13  time_and_date_2 = ephem.Date("2020/01/01 00:00:00")
14
15  level=6
16  gaiamap = Gaia(level=level)
17  pixel_list = np.arange(12*(2**level)**2)
18
19  def calculate(healpix):
20      ra, dec = gaiamap.getRa(healpix), gaiamap.getDec(healpix)
21      dummy = gammasource(["dummy_%s"%healpix, ra, dec, "dummy"])
22      model = nsbModel(con, gaiamap, time_and_date_1, time_and_date_2, version="hess_currents",
                         timeresolution=20, verbose=False)
23      model.setSource(source=dummy)
24      model.calculateTimespan_fast(moonallowed=True)
25      return np.array(model.bright)
26
27  # start the parallel calculations
28  with Pool(processes=18) as pool:
29      results = pool.map(calculate, pixel_list)
30
31  np.save("skymap_timegain_data_moon_alt30.npy", results)
```

Figure 3.26.: Script to generate the raw data for the time gain calculations

Back in the script at lines 12-16, the parameters for start and end date are set and the gaia map is initialized. This is done once before all the calculations take place, as it takes up to a few seconds to load the star lookup file from disc. Because the following calculation is quite extensive, a multithreading approach was used: 18 CPU cores work asynchronously in parallel, each on a single healpix, to calculate the brightness for the whole timespan. Each processor will therefore call the function `calculate`. The first lines of that function set up

```
# allskymaps Config File
# Date and Time
time = today now
#time = 2017/01/18 23:30:00
# Observer Location (HESS is at 16.5028 -23.27280 @ 1800.0)
Lon = 16.5028
Lat = -23.27280
elevation = 1800.
# model function version [krisciunas, hess_basic, hess_advanced]
version = hess_advanced
# output Imagesize in Pixels (its gonna be square)
image_size = 200
# observation position in the sky [deg]
alt = 90.0
az  = 0.0
# Level of HealPixMap used for plotting gaia catalog
healpixlevel = 7
# Gaussian smoothing kernel [pixel]
gauss = 0.0
moon_above_horizon = 0.0
sun_below_horizon = -18.0
source_above_horizon = 30.0
```

Figure 3.27.: NSB tool standard configuration file

a dummy source for the model to be tracked. It is located at the given healpix center and provides a convenient way of encapsulating a coordinate pair that can be read by the model tool.

After that, the model can be initialized with the proper parameter options and finally `model.calculateTimespan_fast(moonallowed=True)` is called in line 24. There exists a corresponding function without the "_fast" in the name, which is used to generate seamless plots. The fast version however will skip in time over periods where either the Sun is above or the source is below the horizon in order to save calculation time. The passed option steers the behavior in order to in- or exclude moonlight observations. For the time gain calculations, both variants have to be calculated one after the other.

In the end, the results object is stored as a plain numpy file to be read and processed later. In the case with 18 parallel threads, this script took 2 hours and 27 minutes on a modern workstation PC to finish once, but of course both cases for moonlight and regular observations have to be calculated. Luckily this only has to be done once, at least as long as the model does not change.

### The actual Plots

The next script to generate all the data-plots and results is presented here in five parts, but of course originally is only one file. There is intentionally no pseudo code shown here, but the actual Python code used during the work on this thesis, to maximize the transparency and enable full reproducibility of the work for future research.

In part 1 of the script (figure 3.28), the previously calculated raw data maps are loaded (line 9 and 10) together with the exposure maps of H.E.S.S. for four consecutive years, which are then averaged. The `make_map(limit)` function will be called in parallel for all the given

limits. This happens in lines 74 and 75 with a total of 8 CPU cores and takes 1 hour and 2 minutes of computation time.
Every element in the `moon` and `no_moon` array is a list of brightness values. Each list corresponds to a HEALPix of the sky and every list entry is a single measurement at time $t$ as shown in table 3.2.

$$
\begin{array}{llllll}
\text{moon} = [ & [b_{(hp=0,t=0)}, & b_{(hp=0,t=1)}, & b_{(hp=0,t=2)}, & \cdots & b_{(hp=0,t=26280)} \quad ], \\
 & [b_{(hp=1,t=0)}, & b_{(hp=1,t=1)}, & b_{(hp=1,t=2)}, & \cdots & b_{(hp=1,t=26280)} \quad ], \\
 & & \vdots & & \vdots & \\
 & [b_{(hp=49152,t=0)}, & b_{(hp=49152,t=1)}, & b_{(hp=49152,t=2)}, & \cdots & b_{(hp=49152,t=26280)} \quad ]\,]
\end{array}
$$

Table 3.2.: Raw data structure for the time gain calculations. The structure is identical in the `no_moon` case.

The total amount of time a given spot is darker than the limit is the number of (in this example visualization horizontal) list entries that is smaller than the given limit. It is calculated for the moon- and moonless-case in lines 34 and 47 respectively. The expression `element<limit` yields a boolean mask with entries either `True` or `False` with the same dimension as the original `element`-array. In many programming languages, including Python, "true" evaluates to 1 and "false" to 0 if reinterpreted as numerical values. This way, the `sum()` expression can be used to count the number of true occurrences.
The accumulated charge is calculated from the integrated (summed) brightness values in lines 41 and 54. This Python expression is very similar to the one above, with the slight difference, that the boolean mask is directly applied to the array in order to get a sub set. Therefore the `sum()` expression actually sums up all the brightness values and not just zeros and ones. It is then divided and multiplied with the necessary conversion factors. The expression directly follows from equations 3.23 and respectively equation 3.43, where $5.0528 \cdot 10^{-8} = \frac{1}{19.791 \cdot 10^{-6}}$. The other two factors are just 20 minutes and 60 seconds per minute representing the time interval of each simulation step.
Finally, the time values are divided by 3 to get units of hours. In the end, everything is returned in a storage class `hpmap` for convenience.

```python
import healpy as hp
import matplotlib.pyplot as plt
import matplotlib.patches as patches
plt.style.use('science')
import numpy as np
import time
from multiprocessing import Pool

moon    = np.load("skymap_timegain_data_moon_alt30.npy")
no_moon = np.load("skymap_timegain_data_nomoon_alt30.npy")

exposure1 = np.load("skymap_timegain_data/hess_exposure_level6_2013.npy")
exposure2 = np.load("skymap_timegain_data/hess_exposure_level6_2014.npy")
exposure3 = np.load("skymap_timegain_data/hess_exposure_level6_2015.npy")
exposure4 = np.load("skymap_timegain_data/hess_exposure_level6_2016.npy")
exposure = (exposure1 + exposure2 + exposure3 + exposure4) / 4
exposure[exposure==0] = np.nan


class hpmap(object):
    def __init__(self, time_moon, time_nomoon, charge_moon, charge_nomoon, limit):
        self.time_moon = time_moon
        self.time_nomoon = time_nomoon
        self.time_diff = time_moon - time_nomoon
        self.charge_moon = charge_moon
        self.charge_nomoon = charge_nomoon
        self.charge_diff = charge_moon - charge_nomoon
        self.limit = limit


def make_map(limit):
    time_moon, time_nomoon, charge_moon, charge_nomoon = [], [], [], []
    for element in moon:
        s = sum(element<limit)
        if s > 0:
            time_moon.append(s)
        else:
            time_moon.append(np.nan)

        # charge = sum MHz / 19.791 [MHz/uA]   *   time
        c = (sum(element[element<limit])/ (19.791)) * (20 * 60)
        # get rid of the u and the result is i units of [As=C]
        c *= 10**(-6)
        charge_moon.append(c)

    for element in no_moon:
        s = sum(element<limit)
        if s > 0:
            time_nomoon.append(s)
        else:
            time_nomoon.append(np.nan)

        # charge = sum MHz / 19.791 [MHz/uA]   *   time
        c = (sum(element[element<limit])/ (19.791)) * (20 * 60)
        # get rid of the u and the result is i units of [As=C]
        c *= 10**(-6)
        charge_nomoon.append(c)

    time_moon = np.array(time_moon) / 3
    time_nomoon = np.array(time_nomoon) / 3
    charge_moon = np.array(charge_moon)
    charge_nomoon = np.array(charge_nomoon)

    return hpmap(time_moon, time_nomoon, charge_moon, charge_nomoon, limit)

limits=[0, 50, 60, 70, 80, 90, 100, 110,
        120, 130, 140, 150, 160, 170, 180, 190,
        200, 210, 220, 230, 240, 250, 260, 270,
        280, 290, 300, 325, 350, 375, 400, 425,
        450, 500, 550, 600, 750, 800, 850, 900,
        950,1000,1250,1500,1750,2000, 10000, 100000]

from multiprocessing import Pool
with Pool(processes=8) as pool:
    results = pool.map(make_map, limits)
```

Figure 3.28.: Script for the time gain calculations (part 1)

Part 2 in figure 3.29 produces the first plots, namely the allsky time map of figure 3.15 from page 50 and the all-sky-usage plot of figure 3.17 from page 51.
For the sky-usage data, the total visibility map is needed. This is obtained in line 90 as the last entry (index of -1) in the results object, which calculated the time every spot in the sky is observable below an "infinite" limit of 100 000 MHz, representing the case of *no limit*. To verify that this number was indeed chosen large enough, see the bottom two rows of table 3 on page 125, which lists the results calculated here (the LaTeX source code for this table itself is generated from data in part 5 of this script). The values do not change if going from 10 000 MHz to 100 000 MHz. Then the sky-usage can simply be calculated as the mean exposure divided by the total visibility (line 16).
After that, the time gain data will be calculated for every limit in line 103. The loop over all limit values first performs an element wise multiplication with the sky usage map and then sums up all spacial values before dividing by 33.76, the number of HEALPix in the field of view of the camera for the given resolution. The expression `np.nansum()` is a special variant, which ignores values that are not numbers (*nan*: not a number). It is used, because the model will return such *nan*-objects in the cases it is not valid (e.g. source below the horizon, sun above the horizon) as a safety measure, since any other numeric value as 0 or -1 might be wrongly interpreted as brightness and enter the calculations unnoticed. After this summation over all HEALPix, one is left with a single number containing the hours of observation time for each limit. Those are stored in a list, e.g.:

$$\texttt{total\_time\_moon} = [\quad t_{(limit=0)},\quad t_{(limit=50)},\quad t_{(limit=60)},\quad \ldots,\quad t_{(limit=100000)}]$$

Since moon time in this context was derived from the option `moonallowed=True` it contains all moonless data as well. The true additional time points can be calculated per limit by subtracting the two cases. Plain Python lists can not be subtracted, but converting them to `np.array()` first will turn them into objects similar to vectors (or matrices in the case of nested lists), which allows easy element wise operations.
The next loop iterates over PMT gain values and calculates the relative lifetime of the PMTs as well as the lifetime corrected time gain, which are presented in figure 3.22 (page 57) and figure 3.23 (page 57). The variable names connect to the formula symbols as follows:

`delta_charge` $\rightarrow Q_m(x)$ (additional charge from moon observations for a given limit)

`total_charge_nomoon[-1]` $\rightarrow Q_{d\infty}$ (total charge without limits from equation 3.44)

`total_time_nomoon[-1]` $\rightarrow T_{d\infty}$ (total dark observation time defined at equation 3.39)

`tau` $\rightarrow \tau(x)$ (relative PMT lifetime defined at equation 3.44)

`cTG` $\rightarrow T^*_m(x)$ (lifetime corrected time gain defined at equation 3.46).

```
ras = [ 17.75, 22.0,  5.5,   8.5,  6.5,    5.5,  10.75]
decs= [-29.0, -30.0, 22.0, -45.0, 17.75, -67.5, -59.6]
labels = ["GC", "2155", "Crab", "Vela", "Geminga", "LMC", "EtaCar"]

timegain_250 = results[21].time_diff
hp.mollview(timegain_250, nest=True, coord='CG', title="␣")
hp.graticule()
for ra, dec, label in zip(ras, decs, labels):
    hp.projscatter(ra* 360/ 24, dec, marker='*', c='red', lonlat=True, coord="C")
    hp.projtext(ra* 360/ 24, dec, label, lonlat=True, coord="C", color='red')
plt.tight_layout()

plt.savefig("hpmap_timegain_250.pdf", dpi=300)

visibility = results[-1].time_nomoon
sky_usage = exposure / visibility
hp.mollview(sky_usage, norm='log', min=0.001, max=0.20, nest=True, coord='CG', title="␣")
hp.graticule()
for ra, dec, label in zip(ras, decs, labels):
    hp.projscatter(ra* 360/ 24, dec, marker='*', c='red', lonlat=True, coord="C")
    hp.projtext(ra* 360/ 24, dec, label, lonlat=True, coord="C", color='red')
plt.tight_layout()
plt.savefig("hpmap_skyusage_2013-16.pdf", dpi=300)

# preparing the time gain data
total_time_moon,total_time_nomoon, total_charge_moon, total_charge_nomoon = [], [], [], []

for i, limit in enumerate(limits):
    time_moon = (results[i].time_moon) * sky_usage
    time_nomoon = (results[i].time_nomoon) * sky_usage
    charge_moon = (results[i].charge_moon) * sky_usage
    charge_nomoon = (results[i].charge_nomoon) * sky_usage

    total_time_moon.append(np.nansum(time_moon) / 33.762)
    total_time_nomoon.append(np.nansum(time_nomoon) / 33.762)
    total_charge_moon.append(np.nansum(charge_moon) / 33.762)
    total_charge_nomoon.append(np.nansum(charge_nomoon) / 33.762)

delta_time = np.array(total_time_moon) - np.array(total_time_nomoon)
delta_charge, tau, cTG = [], [], []
gains_ADC = np.array([80, 70, 60, 50, 40, 20, 0])
gains = gains_ADC/80

for gain in gains:
    delta_charge.append((np.array(total_charge_moon) - np.array(total_charge_nomoon)) * gain)
    tau.append(total_charge_nomoon[-1] / (delta_charge[-1] + total_charge_nomoon[-1]))
    cTG.append(tau[-1] * (delta_time + total_time_nomoon[-1]) - total_time_nomoon[-1])

plt.figure()
plt.ylabel("relative␣lifetime")
plt.xlabel("NSB␣Limit␣[MHz]")
plt.gca().set_xlim(0,700)
plt.gca().set_ylim(0.35,1.1)
plt.minorticks_on()
plt.grid()
for i, t in enumerate(tau):
    plt.plot(limits, t, label="$\\tau$␣(gain=%.1f)" % gains_ADC[i], linewidth=2)
plt.legend()
plt.savefig("lifetime_gain.pdf", dpi=300)
plt.show()

plt.figure()
plt.ylabel("Time␣[hours/year]")
plt.xlabel("NSB␣Limit␣[MHz]")
ax1 = plt.gca()
ax1.set_xlim(0,700)
ax1.set_ylim(-200,800)
plt.minorticks_on()
ax1.grid()
for i, tg in enumerate(cTG):
    ax1.plot(limits, tg, label="$T_m^*$␣(gain=%.1f)" % gains_ADC[i], linewidth=2)
ax1.legend(loc="upper␣left")
plt.savefig("lifetime2_gain.pdf", dpi=300)
plt.show()
```

Figure 3.29.: Script for the time gain calculations (part 2)

Part 3 in figure 3.30 prepares the data for the PMT gain dependent plots. For this task, a helper function is introduced in line 150, that can return a linear interpolated value for a list of given NSB limits. This is required to transform the NSB limit recommendations of the different approaches (conservative, aggressive, technical limit) to a corresponding time gain, since the limit axis for the original calculations was discrete (see limits list in line 66 of part 1), which does not allow a continuous inverse of the functions depending on the limits.
Functions for parabolic and linear curves are introduced in lines 172 and 175, to be used in a fit below. These fits are necessary to obtain the maximum and the second zero passing values of the discrete calculated function $T_m^*(limit)$, which represent the conservative and aggressive limit recommendations.
Those recommendations are implemented as `opt1` and `opt2`. The third one, `opt3`, the technical possible NSB limit, is calculated with the according Python function (line 178), which implements equation 3.50 from page 58.
To find the maximum of $T_m^*(limit)$, the index of the maximum value of the discrete function is taken as a starting point (line 195). From here, the parabolic fit is performed including the data points with distance `margin=3` around that index `n`. The `curve_fit` function imported from `scipy.optimize` returns two objects: a list with the optimal parameters and the covariance matrix. Line 202 therefore represents the use of the first derivative of the parabola

$$f(x) = a \cdot x^2 + b \cdot x + c$$

$$f'(x) = 2 \cdot a \cdot x + b$$

to calculate the location of the maximum based on its parameters. With `popt[0]`$==$ a, `popt[1]`$==$ b and `popt[2]`$==$ c, it can be read as $x_0 = \frac{-b}{2 \cdot a}$. Line 203 will calculate the maximum value at that location using the fitted function and stores it in `opt1_y`.
To find the second zero passing, a similar approach is used. The starting point `n` for the (this time linear) fit is found by incrementing it as long as the value of `cTG` at that index is still greater than zero and the index is still within the valid range of limits (line 208). Since the initial `n` is chosen at the rough maximum, this will always find the second zero passing as long as the slope of the linear fit is negative, otherwise the fit result is discarded on line 219. There is no `opt2_y` calculated, since it is always zero by definition.
The technical limit `opt3` does not need to be fitted, but can simply be calculated using the formula. Its corresponding time gain however again relies on the interpolation between the discrete limit values. This is done in line 222.

```python
def get_interpolated(limits_list, timegain):
    t = []
    for limit in limits_list:
        index = np.digitize(limit, limits)
        if len(limits) > index > 1 :
            #print("limit %.1f is in between [%i:%i]" % (limit, limits[index-1], limits[index]))
            upper = timegain[index]
            lower = timegain[index-1]
            delta_t = upper - lower
            delta_l = limits[index] - limits[index-1]
            temp = lower + ((limit - limits[index-1]) / (delta_l)) * delta_t
            #print("Value is %.1f out of [%.1f:%.1f]" % (temp, lower, upper))
            t.append(temp)
        else:
            #print("out of range!" , limit)
            t.append(np.nan)
    return t

opt1, opt2, opt1_y, opt3_y = [], [], [], []

from scipy.optimize import curve_fit

def parabola(x, a, b, c):
    return a*x**2 + b*x + c

def linear(x, m, t):
    return m*x + t

def nsb_limit(gain):
    return ((120/gain) - 90) * 19.791

colors = plt.rcParams['axes.prop_cycle'].by_key()['color']

delta_time = np.array(total_time_moon) - np.array(total_time_nomoon)
delta_charge = []
tau = []
cTG = []

gains = np.arange(0, 1.1, 0.01)
gains_ADC = 80*gains

for gain in gains:
    delta_charge.append((np.array(total_charge_moon) - np.array(total_charge_nomoon)) * gain)
    tau.append(total_charge_nomoon[-1] / (delta_charge[-1] + total_charge_nomoon[-1]))
    cTG.append(tau[-1] * (delta_time + total_time_nomoon[-1]) - total_time_nomoon[-1])
    n = np.argmax(cTG[-1])
    margin = 3
    if n > margin and n < len(limits) - margin:
        y = cTG[-1][n-margin:n+margin]
        x = limits[n-margin:n+margin]
        popt, pcov = curve_fit(parabola, x, y)

        opt1.append(-popt[1] / (2*popt[0]))
        opt1_y.append(parabola(-popt[1] / (2*popt[0]), *popt))
    else:
        opt1.append(np.nan)
        opt1_y.append(np.nan)

    while n < len(limits) and cTG[-1][n] > 0:
        n += 1
    if n + margin < len(limits):
        y = cTG[-1][n-margin:n+margin]
        x = limits[n-margin:n+margin]
    else:
        y = cTG[-1][n-margin:n]
        x = limits[n-margin:n]
    popt, pcov = curve_fit(linear, x, y)
    if popt[0] < 0:
        opt = -popt[1] / (popt[0])
    else: # slope positive -> no second zero passing
        opt = np.nan
    opt2.append(opt)
    opt3_y.append(get_interpolated([nsb_limit(gain)], cTG[-1]))
```

Figure 3.30.: Script for the time gain calculations (part 3)

Parts 4 and 5 (figures 3.31 and 3.32 do not calculate any new values, but create the remaining plots using the values calculated before. The results are:

The absolute time gain depending on the NSN limit:
Figure 3.18 ("timegain.pdf") from page 53

The absolute accumulated charges depending on the NSB limit:
Figure 3.19 ("charge.pdf") on page 54

The relative lifetime of a PMT $\tau$ depending on the NSB limit:
Figure 3.20 ("lifetime.pdf") on page 57

The lifetime corrected time gain depending on the NSB limit:
Figure 3.21 ("lifetime2.pdf") on page 55.

The optimum NSB limit recommendations based on the PMT gain:
Figure 3.24a ("opt_gain.pdf") and the zoomed in version figure 3.24b ("opt_gain_zoom.pdf"), both on page 60.

The resulting absolute time gains if using those limit recommendations: Figure 3.25a ("opt_gain_times.pdf") and the corresponding corrected time gains in figure 3.25b ("opt_gain_cor_times.pdf"). Both to be found on page 61.

Additionally, a table in LaTeX format with all calculated values is generated in line 315, which can be seen in the appendix at page 125. Using this table, all plots could in principle be reproduced without any further calculations (except for the interpolation needed for the inverse functions).

### Final remarks on the tool usage

Modern data science often relies on writing code in any form. But since scientists are no software engineers per se, this task often consumes a huge amount of their working time. Explaining and sharing the code in such an explicit manner, I hope to set a good example and provide a solid foundation for further work based on my findings for others so that they do not have to redo the majority of the workload. I was myself pleased to find the full (working) code of the *pyASB* tool developed by Nievas [2013] alongside the corresponding thesis publicly available. It enabled me to very quickly get started developing the early model versions using the available data at that time.

```
plt.figure()
plt.minorticks_on()
plt.grid()
ax1 = plt.gca()
ax2 = ax1.twiny()
ax2.set_ylabel("NSB␣Limit␣[MHz]")
ax1.set_ylabel("NSB␣Limit␣[MHz]")
ax2.set_xlabel("Gain␣[relative]")
ax1.set_xlabel("Gain␣[ADC/p.e.]")
ax2.set_xlim(1,0.05)
ax2.set_ylim(0,5000)
ax1.set_xlim(80, 4)
ax1.set_ylim(0,5000)
ax1.add_patch(
patches.Rectangle(
    (80, 0),-0.4*80,1200,fill=False,zorder=2))
lns1 = ax2.plot(gains, opt1, label="max($T_m^*$)␣␣(conservative)", linewidth=2, color=colors[0])
lns2 = ax2.plot(gains, opt2, label="$T_m^*␣=␣0$␣␣(aggressive)", linewidth=2, color=colors[1])
lns3 = ax1.plot(gains_ADC, nsb_limit(gains), label="technical␣limit", linewidth=2, color=colors[2])
lns = lns1+lns2+lns3
labs = [l.get_label() for l in lns]
plt.legend(lns, labs, loc="upper␣left")
plt.savefig("opt_gain.pdf", dpi=300)
plt.show()

plt.figure()
plt.minorticks_on()
plt.grid()
ax1 = plt.gca()
ax2 = ax1.twiny()
ax2.set_ylabel("NSB␣Limit␣[MHz]")
ax1.set_ylabel("NSB␣Limit␣[MHz]")
ax2.set_xlabel("Gain␣[relative]")
ax1.set_xlabel("Gain␣[ADC/p.e.]")
ax2.set_xlim(1, 0.6)
ax2.set_ylim(0, 1200)
ax1.set_xlim(80, 0.6*80)
ax1.set_ylim(0, 1200)
lns1 = ax2.plot(gains, opt1, label="max($T_m^*$)␣(conservative)", linewidth=2, color=colors[0])
lns2 = ax2.plot(gains, opt2, label="$T_m^*␣=␣0$␣(aggressive)", linewidth=2, color=colors[1])
lns3 = ax1.plot(gains_ADC, nsb_limit(gains), label="technical␣limit", linewidth=2, color=colors[2])
lns = lns1+lns2+lns3
labs = [l.get_label() for l in lns]
plt.legend(lns, labs, loc="upper␣left")
plt.savefig("opt_gain_zoom.pdf", dpi=300)
plt.show()

plt.figure()
plt.minorticks_on()
plt.grid()
ax1 = plt.gca()
ax2 = ax1.twiny()
ax2.set_ylabel("Time␣[hours/year]")
ax1.set_ylabel("Time␣[hours/year]")
ax2.set_xlabel("Gain␣[relative]")
ax1.set_xlabel("Gain␣[ADC/p.e.]")
ax2.set_xlim(1, 0.6)
ax1.set_xlim(80, 0.6*80)
lns1 = ax2.plot(gains, get_interpolated(opt1, delta_time), label="$T_{m}$␣(conservative)", linewidth=2, color=colors[0])
lns2 = ax2.plot(gains, get_interpolated(opt2, delta_time), label="$T_{m}$␣(aggressive)", linewidth=2, color=colors[1])
lns3 = ax2.plot(gains, get_interpolated(nsb_limit(gains), delta_time), label="$T_{m}$␣(technical␣limit)", linewidth=2, color=colors[2])
lns = lns1+lns2+lns3
labs = [l.get_label() for l in lns]
plt.legend(lns, labs)#, loc="upper left")
plt.savefig("opt_gain_times.pdf", dpi=300)
plt.show()

plt.figure()
plt.minorticks_on()
plt.grid()
ax1 = plt.gca()
ax2 = ax1.twiny()
ax2.set_ylabel("Time␣[hours/year]")
ax1.set_ylabel("Time␣[hours/year]")
ax2.set_xlabel("Gain␣[relative]")
ax1.set_xlabel("Gain␣[ADC/p.e.]")
ax2.set_xlim(1, 0.6)
ax2.set_ylim(-200, 200)
ax1.set_xlim(80, 0.6*80)
lns1 = ax2.plot(gains, opt1_y, label="$T^*_{m}$␣(conservative)", linewidth=2, color=colors[0])
lns2 = ax2.plot(gains, np.zeros_like(gains), label="$T^*_{m}$␣(aggressive)", linewidth=2, color=colors[1])
lns3 = ax2.plot(gains,opt3_y, label="$T^*_{m}$␣(technical␣limit)", linewidth=2, color=colors[2])
lns = lns1+lns2+lns3
labs = [l.get_label() for l in lns]
plt.legend(lns, labs)#, loc="upper left")
plt.savefig("opt_gain_cor_times.pdf", dpi=300)
plt.show()
```

Figure 3.31.: Script for the time gain calculations (part 4)

```python
delta_time = np.array(total_time_moon) - np.array(total_time_nomoon)
delta_charge = (np.array(total_charge_moon) - np.array(total_charge_nomoon))
tau = total_charge_nomoon[-1] / (delta_charge + total_charge_nomoon[-1])
cTG = tau * (delta_time + total_time_nomoon[-1]) - total_time_nomoon[-1]

for i in range(len(limits)):
    print("%i & %.1f & %.1f & %.1f & %.1f & %.1f & %.1f & %.2f & %.1f \\\\"
          % (limits[i],
             delta_time[i], total_time_nomoon[i],  delta_time[i] + total_time_nomoon[-1],
             delta_charge[i], total_charge_nomoon[i], delta_charge[i] + total_charge_nomoon[-1],
             tau[i], cTG[i]))
    if limits[i] == 550:
        print("\\hline")

plt.figure()
plt.ylabel("Time [hours/year]")
plt.xlabel("NSB Limit [MHz]")
plt.gca().set_xlim(0,700)
plt.minorticks_on()
plt.grid()
plt.plot(limits, total_time_nomoon, label="$T_{d}$", linewidth=2)
plt.axhline(total_time_nomoon[-1], 0, 700, alpha=0.5, ls="--")
plt.plot(limits, delta_time, label="$T_{m}$", linewidth=2)
plt.plot(limits, delta_time + total_time_nomoon[-1], label="$T_{d'+m}$", linewidth=2)
plt.legend()
plt.savefig("timegain.pdf", dpi=300)
plt.show()

plt.figure()
plt.ylabel("Accumulated charge [C/year]")
plt.xlabel("NSB Limit [MHz]")
plt.gca().set_xlim(0,700)
plt.gca().set_ylim(0,45)
plt.minorticks_on()
plt.grid()
plt.plot(limits, total_charge_nomoon, label="$Q_{d}$", linewidth=2)
plt.axhline(total_charge_nomoon[-1], 0, 700, alpha=0.5, ls="--")
plt.plot(limits, delta_charge, label="$Q_{m}$", linewidth=2)
plt.plot(limits, delta_charge + total_charge_nomoon[-1], label="$Q_{d'+m}$", linewidth=2)
plt.legend()
plt.savefig("charge.pdf", dpi=300)
plt.show()

plt.figure()
plt.ylabel("Relative lifetime")
plt.xlabel("NSB Limit [MHz]")
plt.gca().set_xlim(0,700)
plt.gca().set_ylim(0.4,1.1)
plt.minorticks_on()
plt.grid()
plt.plot(limits, tau, label="$\\tau$", linewidth=2)
plt.legend()
plt.savefig("lifetime.pdf", dpi=300)
plt.show()

plt.figure()
plt.ylabel("Time [hours/year]")
plt.xlabel("NSB Limit [MHz]")
ax1 = plt.gca()
ax1.set_xlim(0,700)
ax1.set_ylim(-160,60)
plt.minorticks_on()
ax1.grid()
ax1.plot(limits, cTG, label="$T_m^*$",color="green", linewidth=2)
ax1.legend()
plt.savefig("lifetime2.pdf", dpi=300)
plt.show()
```

Figure 3.32.: Script for the time gain calculations (part 5)

# Chapter 4.

# Machine Learning Data Analysis

## 4.1. Introduction

The history of machine learning reaches back to the 1950s. Therefore, the next section will give an overview. The term machine learning was very well defined by Tom M. Mitchel in his Book *Machine Learning* [Mitchel, 1997]:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

In general it is important to realize that all machine learning (ML) tasks can be understood as functions that map an input vector $\mathbf{x}$ to an output value $f(\mathbf{x})$. For a given input, there is always a deterministic output. The often heard and read term of a "magical black box" is inappropriate. For all ML algorithms, no matter how sophisticated or advanced, it holds that: data encoded as numbers (measurement values, image pixel intensities, encoded letters, etc.) coming in will undergo mathematical operations (vector, matrix, higher dimension tensor calculations) and result in an output. Surely those operations can seem overwhelming to an uneducated observer, but there is nothing unpredictable going on when performing those calculations. We will come back to that topic later when talking about deep learning.

### 4.1.1. From Single- to Multi-Layer Perceptrons

In 1949, the Canadian psychologist Donald Hebb postulated the model of learning brain cells in his book *The Organization of Behaviour* [Hebb, 1949] on page 62:

> Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability. [...] When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

This idea of how one cell triggers another one and that learning means increasing their bond and influencing each other, lead to the development of the first perceptron by Rosenblatt [1958]. Initially the perceptron was meant to be a machine rather than an algorithm and the IBM Mark 1 was a device with $200 \times 200$ photocells to classify images, where the weights were built of potentiometers, and weight updates during learning were performed by electric motors turning those knobs.

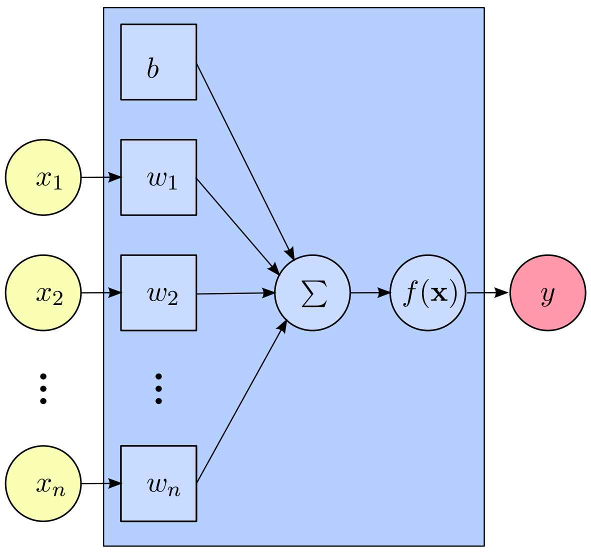


Figure 4.1.: Schematics of the perceptron (blue). The input vector $\mathbf{x}$, where the additional entry $x_0$ is set to 1, is multiplied with the weights vector (dot product, shown here as element wise multiplication and sum) and then fed through the activation function $f(\mathbf{x})$.

The perceptron algorithm can be summarized as follows:
The underlying activation function is known as the Heaviside step function

$$y = f(\mathbf{x}) = \begin{cases} 1 & \text{if } \; b + \mathbf{w} \cdot \mathbf{x} > 0, \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

where $\mathbf{w} \cdot \mathbf{x}$ is the dot product of the weights vector and the input vector and $b$ is the bias. Given a set $D$ of $s$ training samples

$$D = \{(\mathbf{x}^{(1)}, d^{(1)}), \ldots, (\mathbf{x}^{(s)}, d^{(s)})\} \tag{4.2}$$

with $d^{(i)}$ being the desired output for a given input $\mathbf{x}^{(i)}$, the algorithm will be trained. Note that in practice the input vector could have one additional entry more than actual inputs, which is set to be $\mathbf{x}_0^{(i)} = 1$ for all input samples. In this way $w_0 = b$ of the weight vector

takes the role of the bias term when multiplied with the input.
For every sample $i$ of the training data set calculate the output

$$\begin{aligned} y^{(i)} &= f(b + \mathbf{w} \cdot \mathbf{x}^{(i)}) && (4.3) \\ &= f(b + w_1 x_1^{(i)} + \cdots + w_s x_s^{(i)}) && (4.4) \end{aligned}$$

and then update the weights for the next step using the learning rate r

$$\mathbf{w}^* = \mathbf{w} + r \cdot (d^{(i)} - y^{(i)}) \mathbf{x}^{(i)} \tag{4.5}$$

These two steps are altered until the model converges, which is only guaranteed for linearly separable problems, since the perceptron is a linear classifier.

Here, Hebb's idea can be clearly seen: Having an important feature within the input (a specific entry in the vector $\mathbf{x}$ being a large number) will let the corresponding weight entry "grow" with every time it is "excited", resulting in a bigger influence of that neuron to the general outcome of the algorithm. In that sense, the memory of that important feature is stored in the weights.

Nowadays this concept is called single-layer-perceptron and was rather unsuccessful despite the hopes the scientists put on them. It took until the 1960s until the first multi-layer-perceptron emerged. The idea was simple: instead of weighting features and mapping them to an output, one introduced one or more layers of weights to combine features of features (of features of ...) to an output. The advantage was that this construct is now able to solve non-linear problems. The drawback on the other side: The weights could not be trained as easily and efficiently as before. An algorithm known as backpropagation changed the game. The origin goes back to Bryson and Ho [1969], in a following section, the modern version as proposed by LeCun [1985] will be presented.

A multi-layer-perceptron as shown in figure 4.2 can be considered as one of the simplest fully-connected feed-forward neural networks. Information flow is strictly in one direction and every node is connected to all nodes in the previous layer. Every node is hereby a single perceptron, with multiple inputs and a single output. The last layer is called the output layer, and all others (in this case two) are so-called hidden layers. Only the number of nodes in the output is dictated by the problem to be solved. The amount of hidden layers and nodes per such are free to be chosen while building a neural network design.

### 4.1.2. Model Training

The task of any network is to calculate values which are as close as possible to a desired output for every input. This truth value is often called the label and for classification problems most

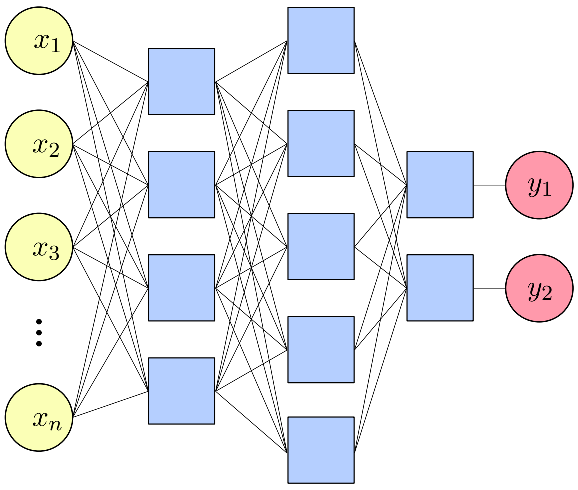


Figure 4.2.: The multi-layer-perceptron is built of perceptron blocks (blue) as described above. Every perceptron takes multiple inputs from the layer before and outputs a single value, which is then passed on to the next layer.

of the time encoded in a one-hot-vector. For a scenario composed of three classes this would be:

$$\begin{aligned}\text{class A} &\longrightarrow \text{label:}\ \ d_0 = (1\ 0\ 0)^T\\ \text{class B} &\longrightarrow \text{label:}\ \ d_1 = (0\ 1\ 0)^T\\ \text{class C} &\longrightarrow \text{label:}\ \ d_2 = (0\ 0\ 1)^T\end{aligned}$$

For regression tasks, the value in question itself can be the label, although scaling might be applied sometimes.

One has then to measure how well the output resembles the label by applying a cost function per sample and then averaging over all samples $n$. A very simple cost function for regression is the mean square error

$$C = \frac{1}{n}\sum_{i=0}^{n}(y^{(i)} - d^{(i)})^2 \tag{4.6}$$

and a very popular one for classification is cross entropy

$$C = \frac{1}{n}\sum_{i=0}^{n} - \sum_{c=0}^{classes} d_c^{(i)} \log(y_c^{(i)}) \tag{4.7}$$

to name just two examples. Initially, the model weights are randomized, resulting in some big cost value and the goal of the training is to minimize the cost by adjusting the weights.

One strategy is to use gradient descent and slowly walk down the hyperplane in weight-space to find a local minimum. For this the derivatives $\frac{\partial C}{\partial w^l_{kj}}$ with $k \times j$ weights and $\frac{\partial C}{\partial b^l_k}$ for all $k$ bias terms are needed for every layer $l$. Once they are computed, it is a matter of choosing the step size (learning rate) and proper parameter initialization to calculate new weights every step. Here I would like to come back to the "black box" mythology: Every output of a network is calculated and the steps to be taken during learning are calculated using the true derivatives. This was already possible with the computing power available to scientists in the 1960s. The only "magical" part about this is the degree of efficiency modern (consumer grade) computer hardware reached to perform such calculations with incredible speed for a comparable low cost.

### 4.1.3. Forward- and Backpropagation

To understand the math of backpropagation, it is helpful to first understand the concept of weight matrices. In figure 4.2, every black line corresponds to a weight value which is multiplied with the value it originates from and is then fed into the summation and activation of a blue perceptron block. In this example, there are two hidden layers with four and five nodes respectively. With $n$ inputs, the first layer has $4n$ weights. By writing them as a $4 \times n$ matrix, the weighted sum of each perceptron node can be performed as a single matrix-vector multiplication $\mathbf{W}^1 \cdot \mathbf{x}$. After adding the bias and applying the activation function to each output of the first layer, the next step would be a multiplication with $\mathbf{W}^2$ which is of size $5 \times 4$, and so on. Here and in the following, superscripts denote the layer number not a power. Subscripts refer to indexes. Matrices are represented with bold upper case letters, vectors with bold lower case letters.

In the general case, the output of all nodes in layer $l$ after activation can be written as a vector

$$\mathbf{a}^l = \sigma(\mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l) \tag{4.8}$$

$$\mathbf{a}^l = \sigma(\mathbf{z}^l) \tag{4.9}$$

with $\sigma$ being an arbitrary but derivable activation function and $\mathbf{z}^l$ the non activated sum within a node.
Now consider three layers ($l-1$, $l$ and $l+1$) in the middle of a network of depth $L$ with $j$, $k$ and $m$ nodes in each of those three. The two connecting matrices are denoted $\mathbf{W}^l_{(k\times j)}$ and $\mathbf{W}^{l+1}_{(m\times k)}$.

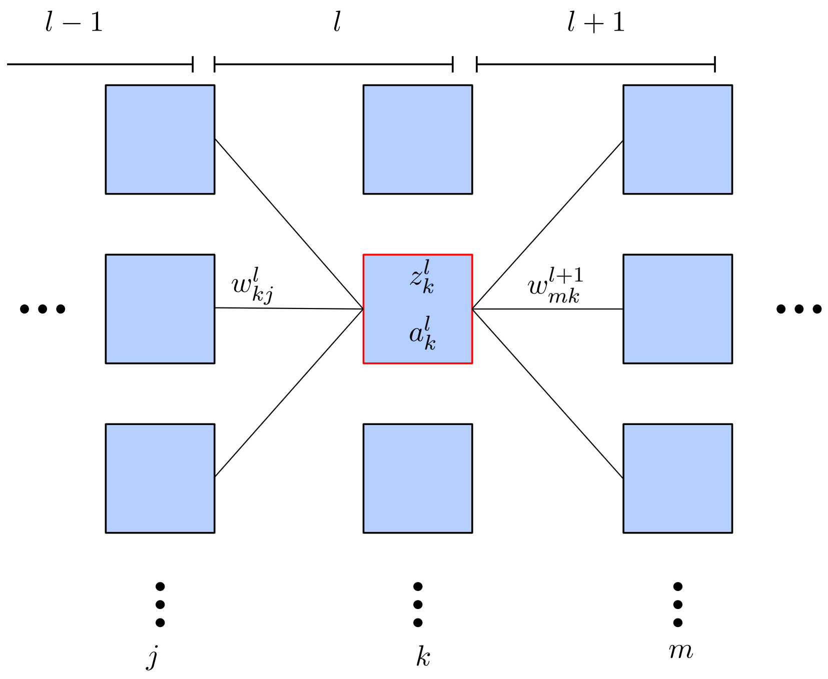


Figure 4.3.: Backpropagation single node

For the highlighted neuron $k$ in layer $l$ (figure 4.3) the pre-activation sum reads:

$$z^l_k = \sum_j w^l_{kj} a^{l-1}_j + b^l_k \tag{4.10}$$

with applied transfer function we can write:

$$a^l_k = \sigma(z^l_k) \tag{4.11}$$

Going one step ahead in the neural network ($l \to l+1$) the corresponding sum reads:

$$z^{l+1}_m = \sum_k w^{l+1}_{mk} a^l_k + b^{l+1}_m \tag{4.12}$$

$$\tag{4.13}$$

Calculating the influence of the highlighted node onto the cost function using the chain rule yields:

$$\frac{\partial C}{\partial w^l_{kj}} = \frac{\partial C}{\partial z^l_k}\frac{\partial z^l_k}{\partial w^l_{kj}} = \frac{\partial C}{\partial a^l_k}\frac{\partial a^l_k}{\partial z^l_k}\frac{\partial z^l_k}{\partial w^l_{kj}} = \tag{4.14}$$

$$= \left(\sum_m \frac{\partial C}{\partial z^{l+1}_m}\frac{\partial z^{l+1}_m}{\partial a^l_k}\right)\frac{\partial a^l_k}{\partial z^l_k}\frac{\partial z^l_k}{\partial w^l_{kj}} = \tag{4.15}$$

with applied derivatives:

$$= \left( \sum_m \frac{\partial C}{\partial z_m^{l+1}} w_{mk}^{l+1} \right) \sigma'(z_k^l) a_j^{l-1} \tag{4.16}$$

Here, $\sigma'(z)$ is short for the derivative of the transfer function $\sigma(z)$.

Defining the "error signal" $\delta$ as the amount the total cost changes under the influence of a single node's pre-activation sum:

$$\delta_k^l \equiv \frac{\partial C}{\partial z_k^l} \tag{4.17}$$

Which can be expanded in a similar fashion:

$$\delta_k^l = \left( \sum_m \frac{\partial C}{\partial z_m^{l+1}} w_{mk}^{l+1} \right) \sigma'(z_k^l) \tag{4.18}$$

using the definition 4.17, we get the recursive formula:

$$\delta_k^l = \left( \sum_m \delta_m^{l+1} w_{mk}^{l+1} \right) \sigma'(z_k^l) \tag{4.19}$$

The input variables for obtaining the error signal are available from prior steps: $w_{mk}^{l+1}$ are the weight values in the matrix, $z_k^l$ is calculated during every forward pass through the network and can be stored for the backwards pass and $\sigma'$ is the derivative of the activation function which should already be known during model architecture design.

One must not forget the bias terms and their influence on the change in cost:

$$\frac{\partial C}{\partial b_k^l} = \frac{\partial C}{\partial z_k^l} \frac{\partial z_k^l}{\partial b_k^l} = \frac{\partial C}{\partial z_k^l} \cdot 1 = \delta_k^l \tag{4.20}$$

The gradient of the cost function with respect to the bias for every neuron is just the error signal for that neuron.
Now we can start at the last layer $L$ and from there on propagate backwards through the network to calculate all error signals.

$$\delta_m^L = \frac{\partial C}{\partial z_m^L} = \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial z_m^L} = \frac{\partial C}{\partial a_m^L} \sigma'(z_m^L) \tag{4.21}$$

Note that the only thing to calculate here is the derivative of the cost function with respect to the last layer's activation, which is the neural networks final output $\mathbf{a}^L = \mathbf{y}$. It only depends on the choice of cost function, e.g. using Eq. 4.6 for the mean square error and deriving it

for y is easy to do. After that, it is a walk backwards through the model using the recursive formula 4.19 to obtain all error signals and therefore all derivatives needed for a gradient descent step.
All this can be written rather short with vector and matrix notations.

Error signal of the last layer:

$$\boldsymbol{\delta}^L = \boldsymbol{\nabla}_{a^L} C \odot \sigma'(\mathbf{z}^L) \tag{4.22}$$

Recursion for all other layers:

$$\boldsymbol{\delta}^l = (\mathbf{W}^{l+1})^T \boldsymbol{\delta}^{l+1} \odot \sigma'(\mathbf{z}^l) \tag{4.23}$$

The derivatives of the cost function with respect to the biases and weights:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{4.24}$$

$$\frac{\partial C}{\partial w_{kj}^l} = \delta_k^l a_j^{l-1} \tag{4.25}$$

Luckily, these algorithms are available in a vast selection of software frameworks, ready to be used. For this work, the Python library *Tensorflow* [Abadi et al., 2015] was used, which was "developed by the *Google Brain* team for internal *Google* use. It was released under the Apache License 2.0 on November 9, 2015." [*]

[*] https://en.wikipedia.org/wiki/TensorFlow, as of April 2020

### 4.1.4. Convolutional Neural Networks

Having introduced the basic principles behind model training, forward- and backpropagation, it is time to talk about architectures. Because the list of possibilities is endless, this section will focus on introducing Convolutional Neural Networks (CNNs) since they represent modern deep learning and were used heavily during the work for this thesis.
Up to now, the input to a model was considered to be a vector $\mathbf{x}$ with all the entries being features of the problem. A simple example classification on which a model can be trained would be: *Decide on an action for stock trading, based on the features*:

- price change since yesterday
- price change since last year
- USD to EUR exchange rate
- time of the day
- and many more...

Using those inputs, a network can find dependencies in the data and then output a prediction: Buy, hold or sell. Similar examples are possible for regression tasks, where a continuous value has to be predicted. In this example it could be the estimated stock value for the next day.
When focusing on two-dimensional data like grayscale images, one could still imagine to just flatten the pixel array and take every brightness value as an input. Indeed, that is what IBM's Mark 1 Perceptron was doing back in the days. This only works to a certain extend. Since the connections between nodes are weighted linear combinations of the inputs, the model could change it's weights in the following manner:
*If certain pixels in the image are bright, while specific others are dim or have some special values, the decision is: output xy.*
Inputting the exact same image data but shifted by one pixel will look extremely familiar for a human observer, but requires completely different weights to be activated in the network. This kind of architecture, by design, is able to recognize and label data on a pixel level only. There is no way of learning concepts e.g. of shapes. For this, pre-processing is definitely required. Applying kernels to the data by convolution is one such idea. These kernels could be optimized to peak at edges, corners, gradients, etc. and then the network will be based on those features instead of the raw pixel input, to make correlations between those and the desired output.

Feature engineering has been and still is today a big research area, but will ultimately be limited to a general understanding of the problem at hand. Convolutional Neural Networks are different, as in this case the values that make up the kernels are trainable parameters

themselves. For example, a specific network is not designed to extract circular features out of an image which will then hint to the presence of two eyes and finally the detection of a face. The total cost of the output is simply lower if during a training step the kernel is altered in a subtle way according to its gradient with respect to the cost and in the case of detecting facial features this might as well mean developing circular kernels. Or any other feature that works and lowers the cost. Nobody *told* the algorithm to look for circles, it *learned it by itself*! [*]
The first time such an architecture was used in a neural network together with efficient use of gradient descent was by no chance LeCun et al. [1989], who introduced the modern backpropagation algorithm just a few years earlier.

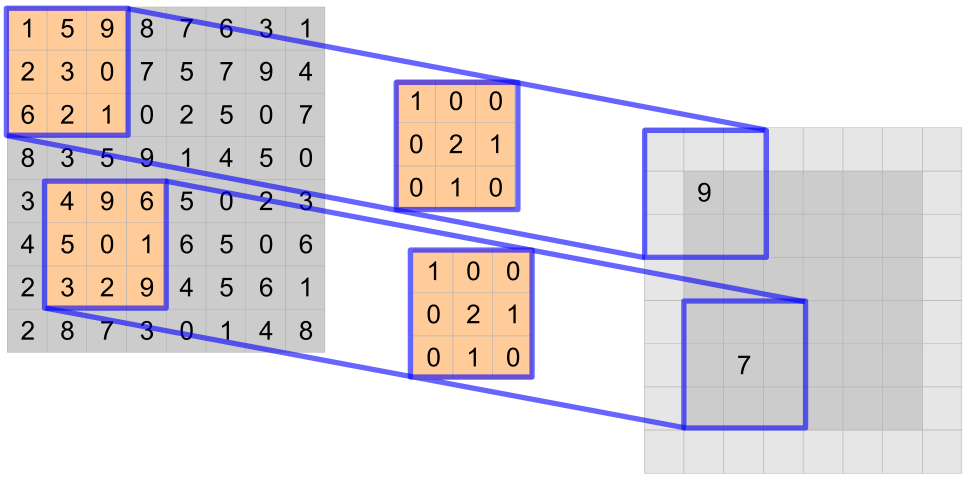


Figure 4.4.: Convolution with a $3 \times 3$ kernel at every location in the data. The output dimensions will either be reduced or the original data has to be padded with e.g. zeros at all four sides, otherwise the kernel cannot reach the edges.

Figure 4.4 shows a $3 \times 3$ kernel being applied to $8 \times 8 \times 1$ dimensional data by taking the sum of the element-wise product:

$$1 \cdot 1 + 5 \cdot 0 + 9 \cdot 0 + 2 \cdot 0 + 3 \cdot 2 + 0 \cdot 1 + 6 \cdot 0 + 2 \cdot 1 + 1 \cdot 0 = 9$$

$$4 \cdot 1 + 9 \cdot 0 + 6 \cdot 0 + 5 \cdot 0 + 0 \cdot 2 + 1 \cdot 1 + 3 \cdot 0 + 2 \cdot 1 + 9 \cdot 0 = 7$$

The result is stored at the center of the applied location. With this scheme the same kernel is applied at every possible location resulting in the output being of dimension $6 \times 6 \times 1$.

[*]As already mentioned I do not like this terminology too much because it always triggers people that are afraid of artificial intelligent algorithms and robots taking over the world. These algorithms are not clever in the sense of having a high IQ, but rather are extremely clever designed mechanisms that are able to produce results far beyond the creator intentionally thought of.

Sometimes this reduction is not desired, e.g. when many layers of convolutions have to be performed one after another. In this case, the original data can be augmented with appropriate values like zeros. This is called (zero-) padding. After the convolution step, the result is called the *feature map*, since every value represents how prominently the feature expressed by the kernel is represented at this location. The convolution step per layer is usually done with multiple kernels for them to each train on different features. Here, initialization is key. All the different kernels always have to be initialized with different (pseudo-) random numbers. Otherwise the optimization step would take the exact same action on every kernel's weights and they all developed redundant to extract the same feature. In most modern CNN architectures the next step would be *pooling*, which introduces translation invariance.

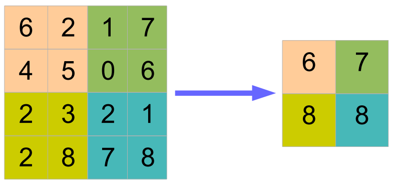


Figure 4.5.: The max pooling algorithm applied with a $2 \times 2$ kernel and stride 2.

There are many pooling techniques possible, but the two most prominent are *average pooling* and *max pooling*. To the proceeding layers it often only matters *that* there was a certain feature present in a specific area and not always *where* it was exactly. A pooling step has to be defined with the hyper-parameters of kernel size and stride (the offset where the next kernel should be applied with respect to the previous one), but does not contain any trainable weights. *Max pooling* is visualized in figure 4.5, where out of every $2 \times 2$ field in the feature map only the maximum value is kept. Average pooling and other variants of course work in accordingly similar fashion.

These two steps are now altered several times, where deeper convolutional layers create feature maps of feature maps (of feature maps ...) until the final set of features is found. How deep, which kernel sizes, how many kernels per layer, which pooling method and stride to use, etc., are all hyper-parameters the neural network designer has to decide on. There is no general rule of thumb and every problem and every data type requires thorough testing. The final feature map is then flattened into a vector and usually fed through a multi-layer perceptron network like discussed above.
A successful implementation of such an architecture can be found in our publication [Shilon et al., 2019], which will be discussed in the following.

## 4.2. Analyzing IACT Data with CNNs

As already mentioned in chapter 2, gamma-ray data analysis faces three main tasks:

- Background rejection: Only gamma-ray induced shower images are of interest, but the majority of triggered data is coming from cosmic rays.
  $\rightarrow$ classification
- Direction reconstruction: To study morphology of extended sources and to estimate background contamination.
  $\rightarrow$ regression in two coordinate variables
- Energy reconstruction: Reconstructing a spectral energy distribution (SED) for further scientific research of the source, like modeling the particle acceleration mechanisms.
  $\rightarrow$ regression

Before those problems can be tackled, some preparation is needed: Data preprocessing and data simulation.

### 4.2.1. Preprocessing

Since the H.E.S.S. camera pixels consist of photomultiplier tubes which have a circular footprint, they are stacked in a hexagonal grid. Unfortunately this means that the data cannot easily be stored as a two-dimensional pixel grid as nearly all other image data. Usually, image data has an underlying regular grid with square pixels and the only variation besides the width and height of an image is the number of color channels. Typically this is one (grayscale), three (red, green, blue; RGB) or four (red, green, blue and alpha; RGBa) values per pixel.

Not only do available software frameworks expect the data to have this common format, the concept of applying e.g. a $3 \times 3$ kernel to a honeycomb grid simply does not work out of the box. One can either resample the data to a square grid, or adjust the framework and convolutional concept to deal with hexagonal data. Within the H.E.S.S. collaboration both paths were pursued and published. See Holch et al. [2018] and Shilon et al. [2019] for details on resampling and a proposal of hexagonal convolutions, and Steppa and Holch [2019] for details on the hexagonal convolution implementation. For this work, the resampling approach was chosen using rebinning and linear interpolation.

The calibration and twofold tailcut cleaning as described in section 2.1.3 on p. 13 are applied to all data using H.E.S.S. standard software tools. After that, the data is exported using a custom exporter and saved to the open file format *HDF5* to be then read with the Python deep learning analysis chain. These files are basically one-to-one copies of the original H.E.S.S. files, still with image data stored per hexagonal pixel (PMT). Figure 4.6

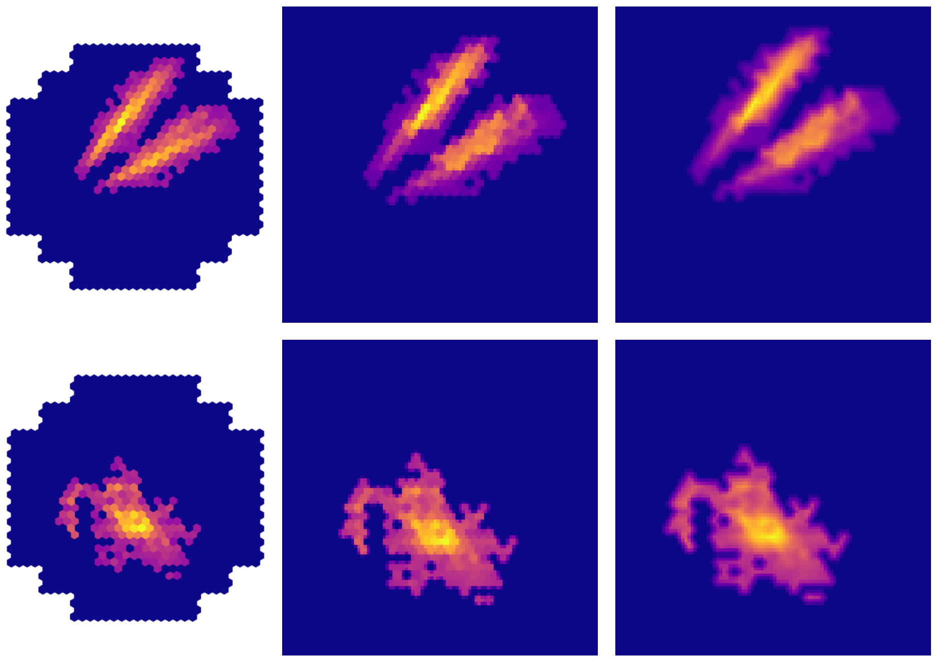


Figure 4.6.: Telescope images of a gamma-ray (top row) and a hadronic (bottom row) event. The images consist of numerous event images laid in a common camera plane, for visualization. Left column : Original camera pixel intensities. The square images show the same event obtained from rebinning the hexagonal histogram (middle) and linear interpolation (right) with a resolution of 100 × 100.

shows examples of the resampling methods applied to a hadronic and a gamma like induced event image. Either method is used during training dataset creation. Depending on the task, the images are normalized to have pixel values stored as floating point numbers in the range 0.0 to 1.0. For direction reconstruction and event type classification, the shape of the image features and their location within the the camera plane are very important. The convolution kernels that are trained to extract them do better with scaled images since these features are relative to each other. An edge that doubles in intensity should be detected as the same kind of edge, no matter if it transitions from 4 to 8 or 42 to 84 pixel units. For energy reconstruction, however, the image amplitude is a key feature and must not be normalized. Global scaling over the whole set of training images is anyway applied to bring down the numbers while preserving the ratio within the dataset. This is necessary because the weights are usually initialized with values $\mathcal{O}(1)$.
The decision whether to use rebinning or interpolation also depends on the task. The impact on the performance of both methods is comparable [Holch et al., 2018], but rebinning is much faster in terms of processing time, since it can be described as a vector matrix multiplication. When overlaying a square grid of resolution $n \times n$ onto the hexagonal arrangement of $k$ camera pixels, each rebinned pixel will contain the amount of every underlying pixel weighted by the

overlapping area of the pixels. The image data is expressed as a flattened column vector $\mathbf{v}_{hex}$, where the order of the pixels within the vector does not matter as long as it is consistent. Its dimension is therefore $(k \times 1)$. The new image is expressed similarly and will therefore have dimension $(n \cdot n \times 1)$. A $(n \cdot n \times k)$-rebinning matrix $\mathbf{M}$ will handle the transformation and contains the fractions of contribution for each combination of old and new pixels. Since naturally not all pixels will contribute to each other, the rebinning matrix will be sparse, which further speeds up computation with the use of special software libraries for sparse-matrix calculations.

$$\mathbf{v}_{square}^{(n \cdot n \times 1)} = \mathbf{M}^{(n \cdot n \times k)} \cdot \mathbf{v}_{hex}^{(k \times 1)} \tag{4.26}$$

$\mathbf{M}^{(n \cdot n \times k)}$ has to be calculated only once per combination of new and old image dimensions and is then stored on disc to be used over and over again. During the development and debugging phase, this method was preferred for its speed.

When working with actual H.E.S.S. data and not only simulations, the rebinning method can only be used for classification tasks. The reason is the way H.E.S.S. handles pointing corrections. These corrections contain deviations from an ideal telescope such as, for example, mechanical bending of the structure depending on the zenith angle, slight twists, sheers or rotations of the camera and the whole structure etc. To deal with all of that, special *Pointing Runs* are taken regularly every month and a pointing model is calculated, which adjusts the pixel coordinates within the camera frame to represent the true configuration. This unfortunately means there is no longer a pixel map where every hexagon is on its expected location, as it is the case in the simulations. A rebinning matrix would have to be calculated on an event-per-event basis, which then clearly introduces a serious overhead and linear interpolation between the pixel values comes to hand as the new best choice for resampling the hexagonal images to square pixeled ones.

If the very precise location and orientation of a shower image in the camera does not matter, as it is the case for the classification task, the fast and easy rebinning can still be used. [*]

[*]With a small catch: the index numbering of the pixels in the real cameras is not the same as in the simulations, which one has to be aware of. Once remapped, the rebinning algorithm works as expected.

### 4.2.2. Monte Carlo Simulations

The deep learning method used in this work is supervised learning, meaning there has to be an underlying ground truth from which the network can learn. In the field of IACT astronomy, this ground truth will always come from simulations, since there is no calibrated reference probe emitting gamma-rays in the range of $1 - 100$TeV one could quickly place above the atmosphere.

The software "**CO**smic **R**ay **SI**mulations for **KA**scade" (CORSIKA [Heck et al., 1998]) was originally developed for the particle detector experiment *Kascade* in Karlsruhe, Germany and has become a standard in the field. With this, one is able to simulate over 50 different particles, like leptons ($e^{\pm}$, $\mu^{\pm}$, $\nu_e$, $\nu_\mu$), mesons ($\pi^0$, $\pi^{\pm}$, $K^0$, $K^{\pm}$, ...), baryons ( $p$, $n$, $\Lambda$, $\Sigma^0$, $\Sigma^{\pm}$, ...), resonance states ($\rho^0$, $\rho^{\pm}$, $\Delta^0$, $\Delta^-$, $\Delta^+$, $\Delta^{++}$) and all corresponding anti-particles propagating through and interacting with the atmosphere, each other, and the magnetic field of the Earth. All known particle interaction channels are taken into account based on extrapolated models from accelerator experiments.

But how reliable is this extrapolated data of a 100 TeV proton, if the most powerful hadron collider, the LHC at CERN, is "only" able to reach peak energies of 14 TeV at the time of this writing? Furthermore this data is gathered from head-on collisions and not in pure forward direction as it is the case with air showers. This further complicates the adaption of the results.

The answer is: We don't know for sure. We believe those calculations extrapolate *sufficiently* well, depending on the used interaction model. See for example Knapp et al. [1996] or Parsons et al. [2011] for an investigation of different commonly used models and their validity for ultra high energy physics.

The CORSIKA output is directly piped to the software *sim_telarray* [Bernlohr, 2008] which calculates the H.E.S.S. telescopes' response to every air shower and finally stores the Monte Carlo (MC) data to disc.

Simulations were done for gamma-ray and proton induced air showers in the energy range of 0.05 to 150 TeV. For both particle types these were carried out with diffuse emission origin at 20° zenith and 180° azimuth with a cone of 5°. For gamma-rays there are additionally simulated point sources with various offsets from the pointing direction to be used for direction reconstruction.

### 4.2.3. Working with H.E.S.S. Data

The following section summarizes the findings reported in Shilon et al. [2019], which I am a co-author of. In this section, all tables and figures showing results are taken directly from that article.

#### Network architectures

The neural network for the classification task was called a *CRNN*, as it is a combination of the traditional Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) architectures. As described before, the convolution part of the network generates the feature maps. Because those features are the same in each of the four cameras, the images are fed through it separately, while the kernels share the same weights. After that, the feature maps of all telescopes have to be combined. A *Long Short Term Memory* (LSTM) layer is used in this place. These are the key components in recurrent networks since they excel in analyzing sequences.

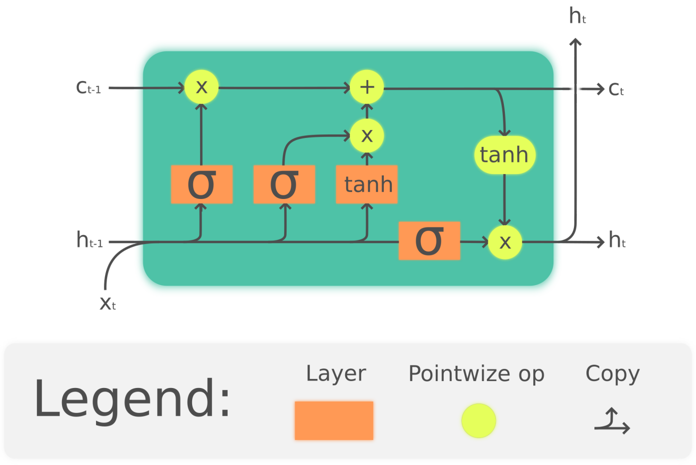


Figure 4.7.: Illustration of the LSTM cell. Image credits: Guillaume Chevalier under the CC-BY 4.0 license. *

The LSTM cell contains three so-called gates: Forget-, input- and output-gate. The index $t$ can be understood as timestamp in a timed sequence or just as sample index in an arbitrary list of inputs $x_t$. The internal state (the memory) is stored in $c_t$ and the output is labeled $h_t$. Even before diving into the understanding of the internals, one can see that each cell's output is related to the previous output and the previously stored internal cell state. The cell state is altered and then passed on every time the cell is called.

*Source: https://commons.wikimedia.org/wiki/File:The_LSTM_cell.png, Creative Commons License: https://creativecommons.org/licenses/by/4.0/legalcode

Taking the previous output $h_{t-1}$ and the current input $x_t$, the perceptron layer followed by a sigmoid activation in the forget-gate outputs a vector with values in the rage 0 to 1 of the same size as the internal state $c$, which is then multiplied element-wise with the passed cell state $c_{t-1}$. The forget-gate is therefore responsible in learning what to ignore of the previous memory, based on the current input.
After cleaning up the internal memory state, it is the input-gate that updates the internal state. This requires two separate tasks, namely identifying which information should be stored and how important it is. Therefore, it consists of two trainable layers, one with again the sigmoid as activation (to encode the importance per feature between 0 and 1) and the other with the hyperbolic tangent function (to encode the feature in the range -1 to 1). They get combined in order to apply the importance scaling and then the result is added to the internal state, which herewith got updated.
At the output-gate, the last trainable layer will calculate a result based on the input. It is then multiplied with the newly updated internal cell state, which is scaled beforehand by an element-wise tangens hyperbolicus ($tanh$). This procedure is repeated for as many elements $t$ as there are in the sequence.
In the case of H.E.S.S. data classification, the sequence consists of the four feature maps sorted by the total image amplitude of the original images. Ideally one would like to sort by the arrival time of the shower front at each telescope, but unfortunately this data is only available for the newly upgraded H.E.S.S. cameras. The simple assumption of a brighter image coming from a shower closer to the telescope is therefore used. The full architecture as used in the publication can be seen in figure 4.8a.
For the regression task of predicting the direction of gamma-ray showers, a much deeper model was necessary. This can be understood by considering the fact that predicting a continuous label is equivalent to doing classification with an infinite number of classes. In this case, it turned out to be advantageous in terms of output accuracy to stack the four camera images and treat them similar to color channels. Usually convolution and pooling alternate each other, but since this reduces the dimension of the feature map each time (roughly by a factor of two, depending on padding and kernel size), pooling is applied only after every other convolution layer. Otherwise the network could not be of the desired depth, e.g. if starting with inputs of dimension $64 \times 64$ pixel as in Fig. 4.8b.
Both networks use dropout. This is a layer without trainable weights that randomly sets a certain amount of inputs to zero. This technique is used as regularization and helps against overfitting by forcing the network to learn broader, more complex feature relations and not relying on individual (sometimes misleading) single features.

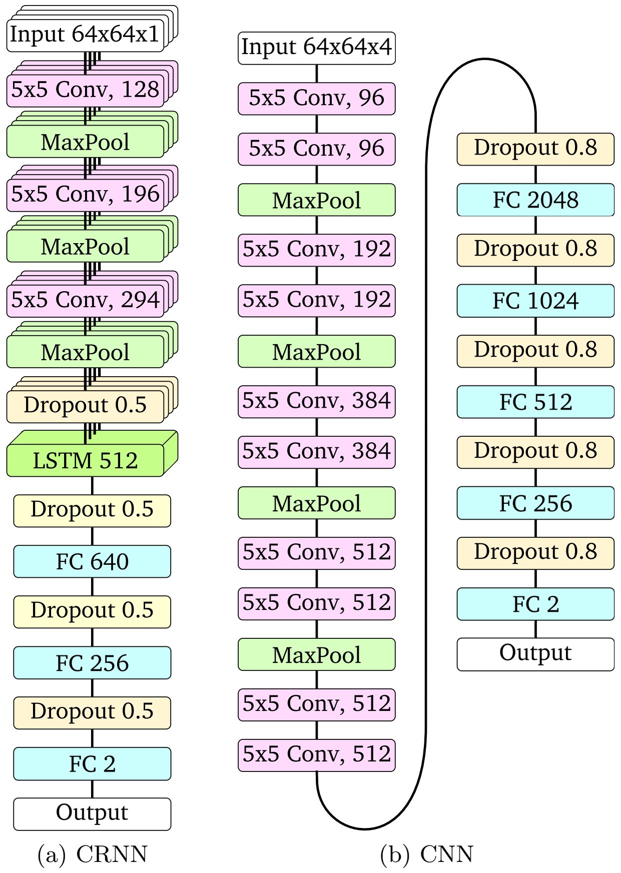


Figure 4.8.: Neural Network architectures used for classification (CRNN) and regression (CNN). FC $x$ means fully connected with $x$ neurons. $n \times n$ Conv, $k$ stands for a convolutional layer with $k$ kernels, each of dimension $n \times n$. (Figure as published in Shilon et al. [2019])

### Network training and data sets

Training was performed on dedicated Linux machines, equipped with powerful but still consumer grade hardware (full system specifications in the appendix, table 4 on page 126). No high performance computing cluster was used for training tasks. Only the simulations, if not already provided by the H.E.S.S. collaboration, were carried out on a cluster, since for this task CPU parallelization is the most profitable. Neural Network training on the other hand benefits from Graphical Processing Units (GPUs) and their specialized hardware to perform vector, matrix and higher order tensor operations.

The description of the used data sets is given in table 4.1.

For classification, there was one training and two different validation sets, while for regression there have been two training and one validation set. The data sets with applied selection

| task | preselection cuts | # events | energy range | type |
|---|---|---|---|---|
| classification training | none | $\gamma$: $1 \times 10^6$ <br> p: $1 \times 10^6$ | $\gamma$: 0.02 - 100 TeV <br> p: 0.2 - 200 TeV | diffuse <br> view cone 2.5° |
| classification validation | none | $\gamma$: 320,000 <br> p: 320,000 | see above | see above |
| classification validation | H.E.S.S. standard | $\gamma$: 196,000 <br> p: 196,000 | see above | see above |
| regression training | none | $1 \times 10^6$ | 0.05 - 100 TeV | diffuse <br> view cone 3.0° |
| regression training | H.E.S.S. standard | 620,000 | see above | see above |
| regression validation | H.E.S.S. standard | 500,000 | see above | point source <br> 0.5° offset |

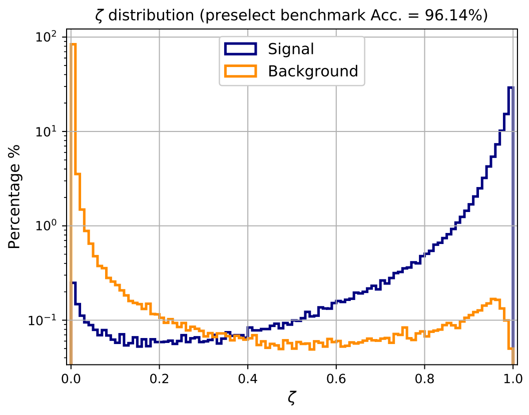
Table 4.1.: Data sets for deep learning training and evaluation. The standard preselection cuts are as explained in section 2.2.2: *size* $> 60$ p.e., *local distance* $< 0.525$ m and *multiplicity* $\geq 2$.

cuts are subsets of the original sets in both cases. Only the point source regression data set was a completely different one, since all other Monte Carlo simulations were performed with diffuse primary particle origin. It was chosen as a benchmark in comparison to gamma-ray point source analysis of real data.

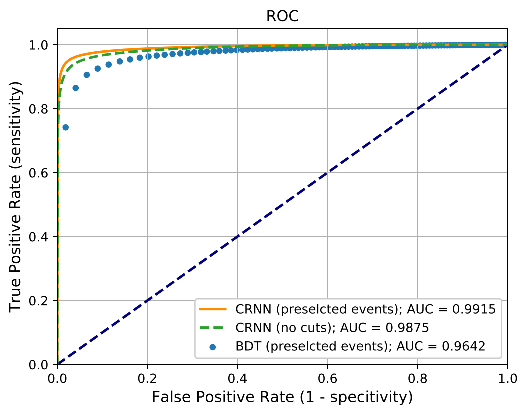
### Results on Monte Carlo Data

The results for both network tasks can be seen in figure 4.9. Figure 4.9a shows the distribution of the classifier output variable $\zeta$, which is designed to be in the range 0 to 1, where higher values mean the classified event is more likely to be a signal, e.g. a gamma-ray. In an ideal scenario both distributions for signal and background would be completely disjoint and a $\zeta$ threshold could be set in between to distinguish the two classes. With this distribution however, one has to carefully decide where to put the threshold. A higher $\zeta$-value will suppress more false positives (the orange bump on the right) while losing true positives, whereas a smaller value increases both the number of true positives and false positives (see the contingency table 4.2 for the term definitions).

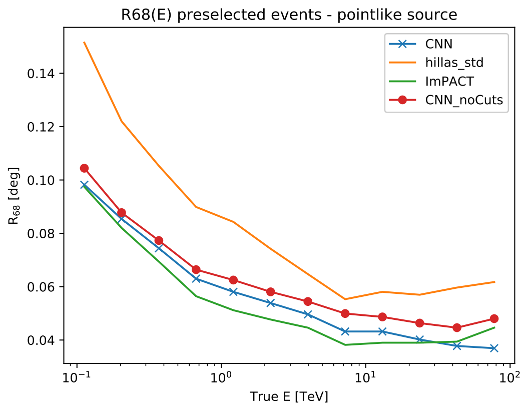
Since our research field is highly background dominated, it is in the best interest to suppress as much background as possible, even at the cost of losing some of the precious signal



(a) Classifier output distribution



(b) Classifier receiver operating characteristic curve



(c) Regressor point spread function

Figure 4.9.: The main results as published in Shilon et al. [2019]. Top: Performance of the CRNN measured on the diffuse simulation data set. Bottom: Performance of the CNN measured on the point source simulation data set

events. Having some signal events over a negligible background is better than having some more which in return get lost in an overwhelming majority of background.

One way to visualize the performance of a binary classifier is the receiver operating characteristic* (ROC) curve. It is obtained by plotting the true positive rate against the false positive rate while scanning the $\zeta$ threshold from 0 to 1. Every point in a ROC diagram is equivalent to a binary classifier with fixed $\zeta$ threshold.

The further away from the diagonal a point is, the better. The diagonal line represents random guessing. Being for example at point (0.7, 0.7) means the model predicted a signal 70% of the time independent of the input, hence it will automatically get a true positive rate

*This term goes back to World War II, where the false and true positive rates of radio antenna receivers trying to identify enemy aircrafts was investigated and tuned.

| | label signal | label background |
|---|---|---|
| prediction signal | **T**rue **P**ositive | **F**alse **P**ositive |
| prediction background | **F**alse **N**egative | **T**rue **N**egative |

| | |
|---|---|
| sensitivity or true positive rate: | $TPR = \dfrac{TP}{TP+FN}$ |
| fall-out or false positive rate: | $FPR = \dfrac{FP}{FP+TN}$ |
| specificity or true negative rate: | $TNR = \dfrac{TN}{TN+FP}$ |
| miss rate or false negative rate: | $FNR = \dfrac{FN}{FN+TP}$ |
| precision or positive predictive value: | $PPV = \dfrac{TP}{TP+FP}$ |
| accuracy: | $ACC = \dfrac{TP+TN}{TP+TN+FP+FN}$ |

Table 4.2.: Contingency table and metric definitions

of 0.7 but also a false negative rate of 0.7. A classifier in the top left corner (0.0, 1.0) is to 100% accurate without any errors.

This curve shows the possible trade-offs a classifier can take but yet alone is not sufficient to rank one over the other. A metric that tries to do so is the Area Under the Curve (AUC). It will be a number between 0 and 1, but no classifier should have a value below 0.5, which would be below the guessing diagonal. Indeed, an algorithm producing a ROC curve on the lower right of the plot would be *worse than guessing.* At first this sounds counter intuitive. It can be understood as a classifier that is seeing the relations of the features, but using them the wrong way. By negating its outcome, one would mirror the ROC at the diagonal and turn the *worse than guessing* classifier into an actually working one.

The AUC can be interpreted as the probability that a randomly selected signal event will have a higher $\zeta$ score than a randomly selected background event [Hanley and McNeil, 1982]. In contrast to other metrics like accuracy or precision, this metric does not depend on the choice of a selected $\zeta$ threshold and is therefore suitable to compare the "raw" predictive power of a given model [Hand and Till, 2001].

Now looking at Fig. 4.9b we can see that the CRNN classifier outperforms the standard H.E.S.S. BDT even on the data set without preselection cuts. Notice that it was only trained on data without cuts. The BDT performance on data without cuts is not shown, but would be unfair anyway, since the BDT was trained on preselected events. The CRNN performance gives hope for this technique to be able to loosen the cuts in general and increase the total sensitivity of the H.E.S.S. experiment.

The regression task on the other hand was not as clearly successful as its classification counter part. Figure 4.9c shows the *R68* plotted over energy. This value is calculated as the 68% containment radius of the reconstructed events around the true source position and usually serves as a measure of the point spread function (although this is normally calculated by running point source simulations through the ray tracing simulation of the telescope in sim_telarray). The performance is clearly better than the simple *Hillas* reconstruction, but failed to beat the state of the art *ImPACT* reconstruction, even with training only on preselected events.

### Results on H.E.S.S. data

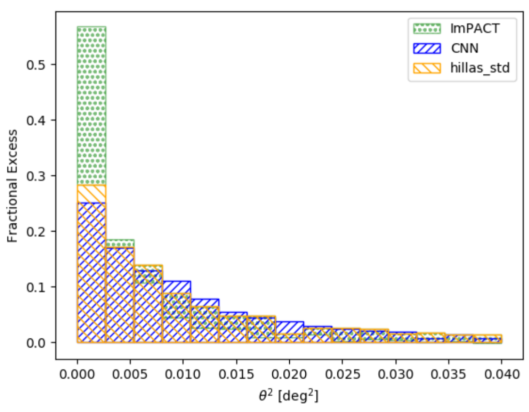
The interesting part of the evaluation of a new analysis chain is the performance on actual data. However, also here the absence of calibration targets for IACT experiments constitutes a challenge. Running a certain number of data events through a classifier and claiming one tool found more signal than another is worthless if one does not know the precision (how many of the claimed signal events really are true signal events) of the tool. Simply relying on the Monte Carlo performance was not an option, since there were already hints that the CRNN might react differently to real data than to simulated data.

The way this issue was addressed is by using the strong reconstruction power of *ImPACT*: An analysis with *ImPACT* was performed twice on the same data set of 14 runs taken on the strong gamma-ray point source PKS 2155-304, once with the class label prediction coming from the standard BDT, and once with the predictions taken from the CRNN. The runs have been carefully selected to pass the H.E.S.S. quality cuts and having a mean zenith angle during the observation of $20° \pm 0.5°$ to match the simulated training data set. After reconstruction, the standard reflected background estimation [Berge et al., 2007] was performed on both results. Since false positives (hadron induced showers classified as gamma-ray showers) are uniformly distributed over the sky, their average contamination can be estimated by looking on regions around the source position where no gamma-rays are expected (On- and Off-regions). The results are summarized in table 4.3. The CRNN classifier manages to increase the number of signal events detected over the amount of background (Signal to Noise ratio) and improves the significance of the detection over the traditional method similarly to the improvement seen on Monte Carlo simulations.

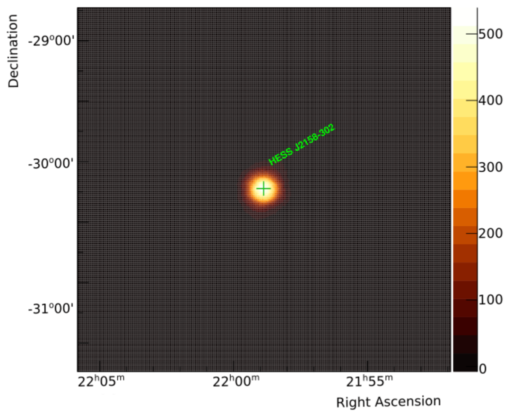
The direction reconstruction was tested on a single run of PKS 2155-304 during the exceptionally bright flare in 2. The $\theta^2$ distributions for the different reconstruction algorithms are presented 006 [Aharonian et al., 2007b]. This data set is very rich in gamma-rays, so the effect of the background rejection method is minimal in figure 4.10. $\theta$ gives the angular distance in degrees of a reconstructed event to the known source position from literature. In this case the CNN regression was no longer better than the *Hillas* method but on the very same level (both have $R68 = 0.102°$) and far behind *ImPACT* (with $R68 = 0.067°$).

| Configuration | $N_{on}$ | $\alpha N_{off}$ | $\sigma$ | Signal / Background |
|---|---|---|---|---|
| *ImPACT* with BDT | 704 | 55.8 | 46.1 | 11.6 |
| *ImPACT* with CRNN | 832 | 62.3 | 50.8 | 12.4 |

Table 4.3.: Results of the CRNN versus BDT comparison tested on PKS 2155-304. Significance $\sigma$ calculated after Li and Ma [1983]. (Table as published in Shilon et al. [2019])



(a) $\theta^2$-histograms



(b) Skymap

Figure 4.10.: Results of reconstructed events from the PKS 2155-304 flare as published in Shilon et al. [2019]. a) Comparing the $\theta^2$-histograms of *Hillas*, *ImPACT* and the CNN regressor. b) The generated skymap using the CRNN classifier and CNN regressor.

This manifests the hypothesis that the features the neural network learns from Monte Carlo simulations cannot be transferred one to one onto real data applications. There has to be an underlying fundamental difference in the event images that is picked up by the networks, but do not disturb the log likelihood fit or geometrical reconstruction methods.

To my knowledge, our article was the first peer reviewed publication on successful application of deep learning techniques to the analysis of IACT data.

### 4.2.4. Conclusions on Machine Learning

Using Machine Learning and explicitly Deep Learning as the state of the art representative for IACT analysis tasks showed mixed results. In principle, given enough data and training time, a model should always be as good as analytic counterparts with the additional potential of outperforming it due to the absence of the restrictions from human feature engineering. The *Hillas* parameters assume elliptical shapes and fail in anything deviating from that, *ImPACT* is designed to fit templates, assuming there will always be a good match available, etc. By training an algorithm on the actual data and letting it develop the necessary features, this bias is gone. The classifier showed this behavior, as it was even able to separate event images without the local distance cut and this way excelled the performance of the traditional methods. However, since this was not actual data which it was trained on, but simulations, the models were not able to generalize good enough in order to transfer that performance to actual measurements.

Unfortunately, there is neither a gamma-ray nor proton calibration source or sample available at ultra-high energies. A Gedankenexperiment would place or hover such a source (e.g. a portable and more powerful version of the LHC) above the telescopes while shielding all other radiation from reaching the cameras and collect its data. This would be a veritable goldmine of information and all analysis techniques (deep learning as well as conventional) could profit. As long as this stays science fiction, simulations based on extrapolated particle models are everything that is available and in our case it turned out they serve *sufficiently well* for the traditional methods but can only be used as a basis for deep learning to a certain extend. It shall be noted, that when speaking of discrepancies in simulations, the simulation of the telescope response is included.[*]

As an analogy I like to think of a very faint watermark-like feature in the corner of the data, a little dot or cross that is barely perceptible. Such a feature would not significantly influence the image moments and therefore the *Hillas* parameters, but could easily be picked up by one of the CNN kernels. If it is only present in one type of the simulation data, the neural network will focus on that feature and yield a perfect prediction of the particle type without even considering the actual Cherenkov light induced part of the image. When used on real data without that watermark, it then surely will fail to do a prediction, whereas the inferior *Hillas* parameters might be off by a bit, but still work as expected.

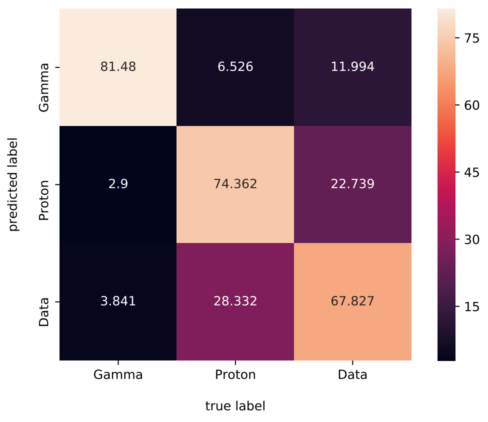
Of course it is not as simple and obvious like that, but some things in the simulations are different to the data and that difference is picked up by the networks but does not majorly disturb the traditional methods. In principle, this is one of the amazing characteristics of

[*]From private communications with D. Jankowsky during the work on his PhD thesis on the cosmic ray proton spectrum I learned, that this influence of the telescope response modeling might even be more severe than the choice of the particle interaction model.

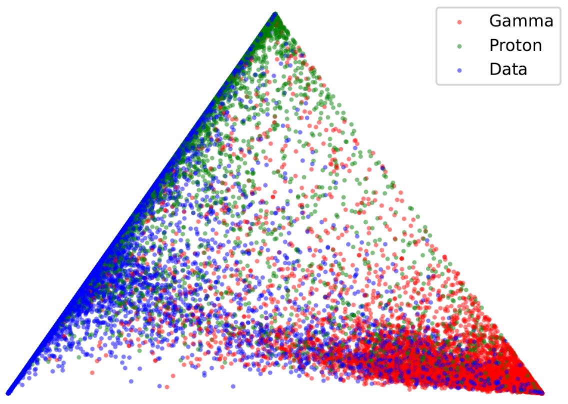
those algorithms: To be able to recognize features a human would never have thought of. But in our case it limits the applicability of those methods.

To showcase the problem, a CNN classifier was trained to separate gamma simulation, proton simulation and real data events into the three classes. In the following it is assumed that the data events are basically only of hadronic origin. To emphasize this, see table 4.3 on page 97. It lists the results of the previously presented classifiers and - depending on which one to trust - states that there were roughly 700-800 signal events in the total of 14 runs used to generate this result. With every data run containing in the order of 200.000 recorded events, the fraction of signal "contamination" can be treated as negligible, even if one would account for false negatives.

The final data set for this purpose ended up with 829537 images (267716 from gamma simulation, 281821 from proton simulation and 280000 from data). 10% of each class was randomly selected and set aside as validation data, which led to a final training sample size of 746583.



(a) Confusion Matrix



(b) Ternary Plot

Figure 4.11.: Results from the three class model trained on gamma simulation, proton simulation and real data events.

After training, the model was evaluated on the validation set. The results are presented in figure 4.11. For this case of three classes, the concept of ROC and AUC are not applicable. Instead, figure 4.11a shows the confusion matrix, which has been horizontally normalized and shows the percentage of true class for every predicted class. Reading it from left to right tells about the true nature of an event the classifier predicted to be of a certain class. Reading it from the bottom to the top gives a feeling on what label an input got on average.

A confusion matrix from a perfect classifier does not contain any values off the diagonal. The presented result of course is far from being perfect, but nevertheless astonishingly good. The model could clearly distinguish gamma simulations from the other two classes, which was to

be expected. But surprisingly it does perform far better than guessing when asked to decide between proton simulations and data events. If the proton simulations were indistinguishable from the data, the middle and bottom row would each read [0 50.0 50.0]. "*This is not a gamma event, but no idea whether it is simulated or recorded.*"
Another way to visualize the relation between the predictions is a ternary plot like in figure 4.11b. The data labels are one-hot-encoded vectors and the models prediction for every event is a tupel of three values in the range zero to one. Those can be used to generate such a plot, where every true label represents a corner of a triangle and the predictions are scattered within according to the values they got assigned. The gamma events are clearly concentrated in their corresponding corner and even the proton and real data points show a clear indication of separation. Ideally the green and blue points would be completely mixed up on a line from the bottom right corner to the middle of the opposite edge (equally likely to be proton or data). Example images of the three classes together with their prediction scores for correct and wrong cases can be found in the appendix (figures 13, 14 and 15 starting at page 120).

This concludes the studies on machine learning with a grain of salt. Those algorithms can perform incredibly well, but are not the magic answer to all data science problems as often stated in public media.
The direction reconstruction via image moments is of pure geometric nature and could in principle be done with pen and paper by intersecting the main axis of two or more elliptical shower images. As discussed before, neural networks excel in finding feature correlations that sometimes even surprise the developers. But if there already exists a good analytic description of the problem, many times it is not advisable to use machine learning algorithms to solve the problem. A nice toy example to demonstrate this is to train a model on a simple analytic task: $y = (x_0 + x_1) \cdot x_2$. Three inputs and one output with a clear relation between them. One Million random numbers were generated for each $x_0$, $x_1$ and $x_2$ in the range [0.0, 1.0] and a small (4 fully connected layers with a total of 6945 trainable parameters) regression model was trained to predict the output. The results on a test set, not included in the training data, are shown below and a more complete visualization in form of the migration matrix can be found in figure 4.12:

$$\begin{aligned}
&(0.823142 + 0.337086) \cdot 0.843242 &&= 0.978353 &&\rightarrow \text{ prediction: } 0.973204\\
&(0.021558 + 0.170060) \cdot 0.615767 &&= 0.117992 &&\rightarrow \text{ prediction: } 0.124893\\
&(0.001810 + 0.252883) \cdot 0.074860 &&= 0.019066 &&\rightarrow \text{ prediction: } 0.025878\\
&(0.152738 + 0.384317) \cdot 0.619247 &&= 0.332570 &&\rightarrow \text{ prediction: } 0.334710\\
&(0.204677 + 0.483588) \cdot 0.842557 &&= 0.579902 &&\rightarrow \text{ prediction: } 0.582187
\end{aligned}$$

Already in this case the predictions are far from perfect, even though it was such a simple task to perform. But one could argue that the model was successfully trained.
To show that the model did not learn how to add and multiply the inputs, but rather found an approximation which is only valid in the given training range, it was evaluated on a set of values in the ranges [0, 2]. In this case, the predictions could at most be called an "educated guess":

$$(0.384909 + 0.738070) \cdot 0.364378 = 0.409188 \rightarrow \text{prediction: } 0.412907$$
$$(2.270194 + 1.418471) \cdot 1.978805 = 7.299150 \rightarrow \text{prediction: } 4.891516$$
$$(2.915166 + 1.519334) \cdot 2.822722 = 12.517361 \rightarrow \text{prediction: } 6.739827$$
$$(2.428426 + 1.751576) \cdot 1.304653 = 5.453454 \rightarrow \text{prediction: } 3.998297$$
$$(0.249571 + 0.293748) \cdot 1.670190 = 0.907445 \rightarrow \text{prediction: } 1.006051$$

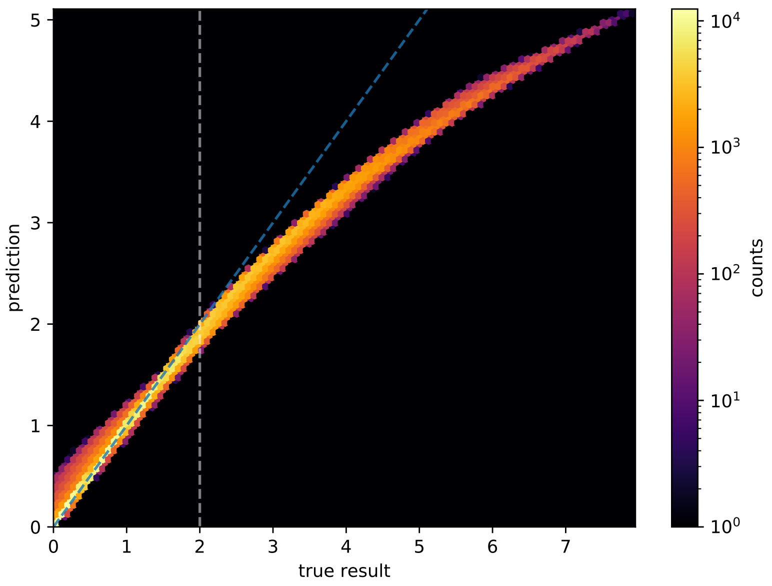
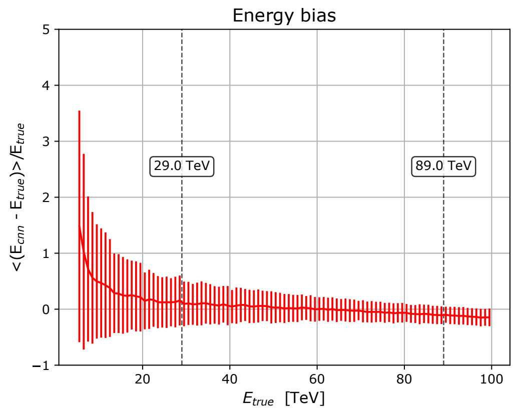


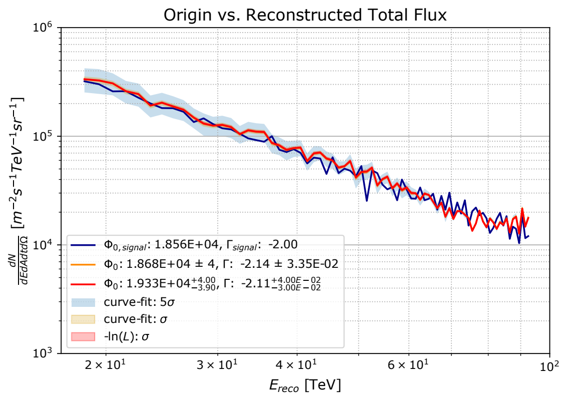
Figure 4.12.: Migration matrix of the toy example. The white dashed line shows the values used during training $2 = (1 + 1) \cdot 1$, the blue line marks the diagonal where the prediction equals the true value.

In the case of IACT data, classification of event data can indeed be improved, but it was not possible to prove superior performance in regression tasks. This is also the reason why no results on energy reconstruction were included in the publication and this thesis. In the classical analysis, the energy is obtained via lookups from the reconstructed shower geometry and the measured amount of light intensity within the camera. As long as the networks are not able to outperform the *ImPACT* and *Model* analysis in shower reconstruction, they will always lag behind in energy estimation performance, since those tasks are so closely related. And as demonstrated, not all problems are best solved with neural networks.

But this does not mean the end of neural network based astroparticle research. As an example, two results of the master thesis from Hillig [2019], which I supervised are presented. Her work examined the possibility to analyze the composition and spectra of heavier nuclei within the H.E.S.S. data, which are usually taken together as "hadronic background". Although the results on real data suffer from similar problems like presented in this work, it shows that the best goal of neural network based research might not be to beat the established techniques, but rather to grant access to topics that seem impossible with the standard methods.



(a) Energy bias from the reconstruczion of iron simulations



(b) Calculated iron flux of simulated events

Figure 4.13.: Regression results on iron simulations using neural networks.
a) The energy bias and resolution (shown as the error bars of the bias) of neural network based reconstruction. The dashed lines mark the region where the bias is within ±10% (29 TeV - 99 TeV).
b) Reconstructed total flux calculated using two different minimization routines within the *forward folding* approach.
Figures from Hillig [2019], where additional information can be found.

# Chapter 5.

# Summary

## 5.1. Moonlight Observations

In the first main chapter of this work, the night sky brightness model was developed and its application presented. Starting with the simple model of Krisciunas and Schaefer [1991], a pipeline was set up to extract photometric data from the available cloud monitor. These CCD camera images of the complete night-sky, which are taken every few minutes regardless of the observation conditions and the Cherenkov telescope schedule, proved to be a viable source of input in the early stage of the model development. It was shown that the general functional description of moonlight induced brightness over the sky and the angular dependencies were carefully chosen by the original model's authors. However, the lack of a proper background description quickly became apparent. At low moon phases or in times where it is not visible at all, the milky way clearly stands out as the brightest region in the sky. Although single stars like e.g. Sirius or Vega do represent the absolute brightest spots, they were not too worrying in first place, as the telescopes do a good job of masking their positions j by switching off the few corresponding pixels those stars are visible in. This technique to protect the camera and improve the data quality was already established within the experiment and the influence of those small numbers of turned-off pixels was proven to have little to no impact on the analysis by the collaboration. But instead of modeling the diffuse emission of the galactic plane, a different, more complete approach was taken: The Python tool which calculates the NSB model and produces the outputs was extended to query the publicly available *GAIA* database API in order to retrieve photometric data of up to one billion stars. The main properties of interest in each data line were the coordinates and visual magnitude of each object. To combine them and generate a map with unit brightness per area, the sphere of the sky was segmented with **H**ierarchical **E**qual **A**rea iso**L**atitude **Pix**elisation (HEALPix). This map is saved as a lookup for various resolutions and provides the new background for the model.

Together with this new background description, a refit of the moonlight scattering function $f(\rho)$, with $\rho$ being the angular separation between the point of interest in the sky and the

Moon, was performed. This was done in a similar fashion as by Krisciunas et al. for their original model. Instead of using the function to calculate the brightness of a certain spot in the sky, the equation was rearranged to provide values for $f(\rho)$ depending on measured brightness values at various locations, separations and moon phases. With the photometric data from the CCD images, an almost complete coverage of the parameter space could be reached for this fit, since every pixel in an image represented a different coordinate in the sky with available brightness data. The results of this advanced NSB model did not only show great visual resemblance with the cloud monitor pictures, but significantly lowered the model deviation from the data.

The final evolution of the NSB tool and its model was reached with the use of actual H.E.S.S. camera monitoring data, when the collaboration started to schedule the first moonlight runs in early 2019. Roughly every second the cameras are operating, the status of every pixel is recorded. This includes the current drawn by the single PMTs, which can be converted to perceived brightness. On the positive side, it allowed to further improve the model by comparing its output directly to brightness measurements of the actual camera hardware, but had the drawback of being stuck with the pointing nature of the telescopes. A typical H.E.S.S. observation run is 28 minutes in tracking mode, where the telescopes follow a coordinate in the sky. Unfortunately, the moon period is too long compared to that time. During this half hour its position relative to the stars as seen from the Earth does not change significantly, which means that on the one hand a run provides a lot of data due to the 1 Hz monitoring rate, but on the other hand does not cover the parameter space (especially the source moon separation) well. On top of that, the extreme cases like strong moon phases or low source altitudes were not covered by these runs. Nevertheless, more than 70 runs with a total observation time of 1610 minutes went into the calculations, with a big part being data taken during the *deeper, wider, faster* campaign [Andreoni and Cooke, 2017] in June 2019. With the newly fitted scaling parameters for the individual model parts and the introduced moon horizon correction, the currents model outperformed the two previous versions. The results of the relative deviation calculations for all three models were:

| | | |
|---|---|---|
| Basic model: | $\mu = -0.2374$ | $\sigma = 0.3843$ |
| Advanced model: | $\mu = -0.0216$ | $\sigma = 0.2250$ |
| Currents model: | $\mu = -0.0046$ | $\sigma = 0.1653$ |

The values for $\mu$ and $\sigma$ are the center and full width at half maximum of the distributions. With this model, predictions of the brightness for all locations in the sky for all time slots over a whole year have been produced. These were analyzed in order to calculate the amount of available time each spot is observable below a certain threshold and the resulting theoretical time gain maps have then been masked with an averaged H.E.S.S. observation pattern in

order to quantify a realistic amount. Additionally, the integrated charge was calculated and similar maps were produced. With those, the amount of (additional) PMT aging - if looking at a specific spot - could be derived.
By comparing the lifetime reduction with the additional time through moonlight observations, a lifetime corrected time gain could be calculated. For nominal gain settings of 80 p.e./ADC, this value was positive up to a NSB limit of 230 MHz with a maximum of $\approx 56$ h/year at 125 MHz. The corresponding maximal absolute time gain is 215 h/year. This is called the conservative limit approach. The so called aggressive balanced approach sets the limit at the point where the lifetime corrected time gain is equal to zero and turns out to yield 446 h/year absolute extra time.
The average amount of total data runs with H.E.S.S. is 735 h/year in the considered years (2013-2016), which means the aggressive balanced approach yields an extra of up to 60% at nominal gain settings. One has to keep in mind that the definition of this strategy was to be at zero lifetime corrected time gain, which means the PMTs reach their end of life up to 60% faster as well. With this approach the same total amount of data will be taken as without moonlight observations, but in a much quicker time.
The last discussed limit was the technical possible maximum. Pushing the PMTs to the edge of their specifications, it would be possible to gain an absolute of 648.5 h/year but effectively means losing 131.9 h/year if taking the lifetime reduction into account. This translates to a theoretical increase of the duty cycle by 88%, but of course should only be considered if money and PMT supply does not play a role, which it certainly does.
Finally, those three strategies were investigated depending on a reduction in gain. Naturally all time gain values increase, since in the extreme case of zero gain, where the cameras are turned off completely, one can point them anywhere - even at daytime.[*] To which extend the gain might be reduced in the future is a decision beyond the scope of this work, as it requires thorough MC studies of the calibration and analysis chain. The conclusion of this work is that moonlight observations are not only feasible but indeed provide a huge increase of available observation time, even at nominal gain setting. Those extra hours do not have to be observed all the time, but they e.g. allow for many more targets of opportunity the system could respond to. The famous GRB 190114C, which was observed by the *MAGIC* collaboration during moonlight, would have been a viable target for H.E.S.S.

[*]Except that having the sun being focused by $107m^2$ mirrors is generally spoken not a very sane idea.

## 5.2. Machine Learning

The second main chapter of this work evaluated the possibility to use machine learning algorithms as an alternative way of analyzing the data already recorded. We like to call our data "images", as they can be easily visualized by color-coding every pixels (the single PMTs) response and showing the result with the proper pixel arrangement on screen. However, this arrangement is the main difference to conventional image data: Instead of orthogonal pixel positions, the H.E.S.S. cameras have an underlying hexagonal grid since the PMTS are simply spoken tubes that are most densely packed this way. In order to use any of the available machine learning frameworks, one has to transform those images in a proper way to be used as an input or rewrite the way the framework defines an image. In this work the first approach of resampling the images was taken: Linear interpolation and rebinning. Interpolation is slower but more versatile. Rebinning is by far the quickest method as it can be described by a matrix-vector multiplication as long as the pixel map is unchanged. This happens when dealing with real data, where each pixel location is virtually changed in order to introduce the pointing corrections of the telescope.

These adjustments do not throw off the standard analysis techniques like the calculation of the *Hillas* parameters via the image moments, since the input to those are the coordinate pairs of each pixel anyway. They do not have to be in any special (or even a regular square) grid for the calculations to work out. The same holds for the *ImPACT* and *Model* analysis.

After preprocessing, the next hurdle was the fact that an air shower event is seen by up to four telescopes, which had to be combined somehow in order be analyzed together. Inspiration was taken from other image based machine learning tasks and the two solutions were: Either interpret the single images as a sequence or treat them as color channels of the same image. The first showed to be better suited for classification tasks. This can be interpreted as the model splitting its decision in four parts, where each is influenced by the ones taken before and was implemented using **L**ong **S**hort **T**erm **M**emory Cells. For the regression task of predicting the shower origins as continuous coordinate values, expressing the event as an image with four channels worked best.

A caveat which was not mentioned yet is the cases where not all four telescopes were triggered. In those events, the corresponding images were set to pixel values of all zero. This might not be the best technique and further investigation in that direction is needed.

The performance of the deep learning models at Monte Carlo simulations were very promising and the classifier clearly outperformed the traditional H.E.S.S. method. This was shown using the decision-threshold independent measure of **A**rea **U**nder the **C**urve, which can be interpreted as the probability that a randomly chosen signal event gets a higher classification score than a randomly picked background event. No classification threshold cut optimization is needed. The results can be found in table 5.1:

| Classifier | trained on | tested on | AUC |
|---|---|---|---|
| CRNN | no cuts | preselected | 0.9915 |
| CRNN | no cuts | no cuts | 0.9875 |
| *ImPACT* BDT | preselected | preselected | 0.9642 |

Table 5.1.: AUC results of CRNN and BDT. To emphasize the possibility of successfully classifying events, which the standard pipeline discards through preselection cuts, the CRNN was trained without any cuts in *local distance* and *size*.

The model for direction reconstruction was clearly better than the *Hillas* analysis on simulated events but could not outperform the two likelihood fitting methods, even if trained only on preselected events. In fact, many of the development models during hyperparameter optimization could not even reach *Hillas* level of reconstruction, which shows the limitations of machine learning techniques.
When evaluating the two models with real H.E.S.S. data, the deep learning classifier was still able to perform better than the standard one, increasing the signal to noise ratio of the PKS 2155-304 analysis from 11.6 to 12.4. The network for direction reconstruction however dropped to the level of the *Hillas* analysis with $R68 = 0.102°$ for both, where *ImPACT* manages to reach $R68 = 0.067°$ on the data of PKS 2155-304.
The reasons for this lack of performance are still not fully understood, but possible explanations include that the simulations are not suitable for machine learning training (as a model was able to distinguish between simulated and real data), or the task cannot be approximated well enough by these algorithms (demonstrated by the toy example of adding and multiplying numbers).

# Appendix

## A. Plots and Images

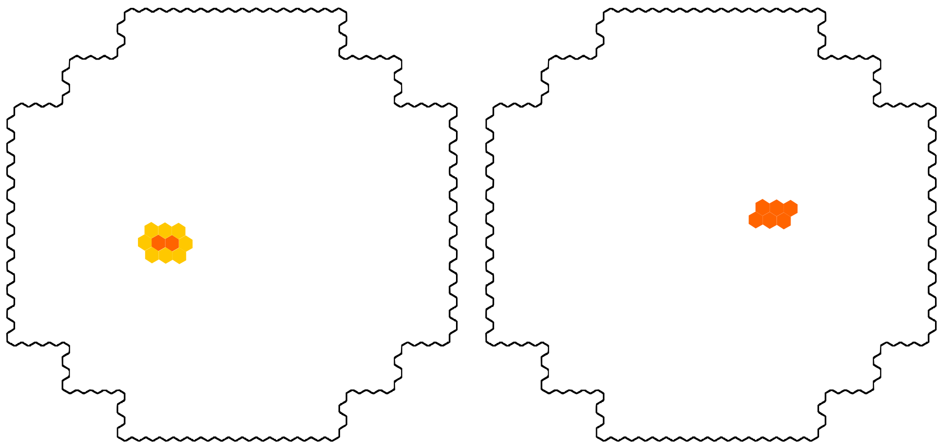


Figure 1.: Illustration of possible 60 p.e. shower images that would survive the tailcut cleaning. On the left two pixels at 10 p.e. surrounded by 8 pixels with 5 p.e., or on the right 6 pixels with 10 p.e.

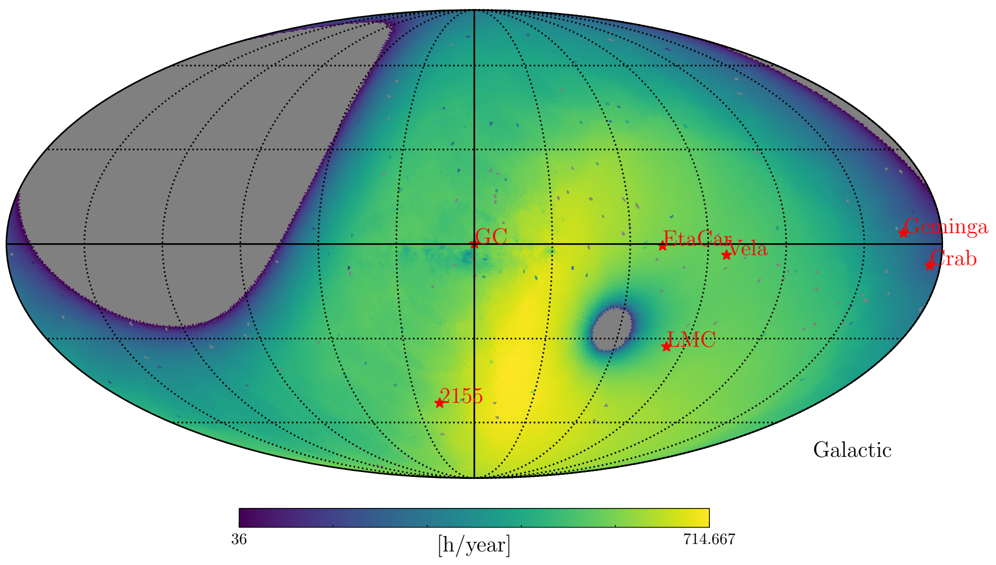


Figure 2.: The potential extra time to be gained per year for every HEALPix, when setting the brightness limit to 550 MHz, the minimum source altitude to 30° and allowing the Moon to be above the horizon.

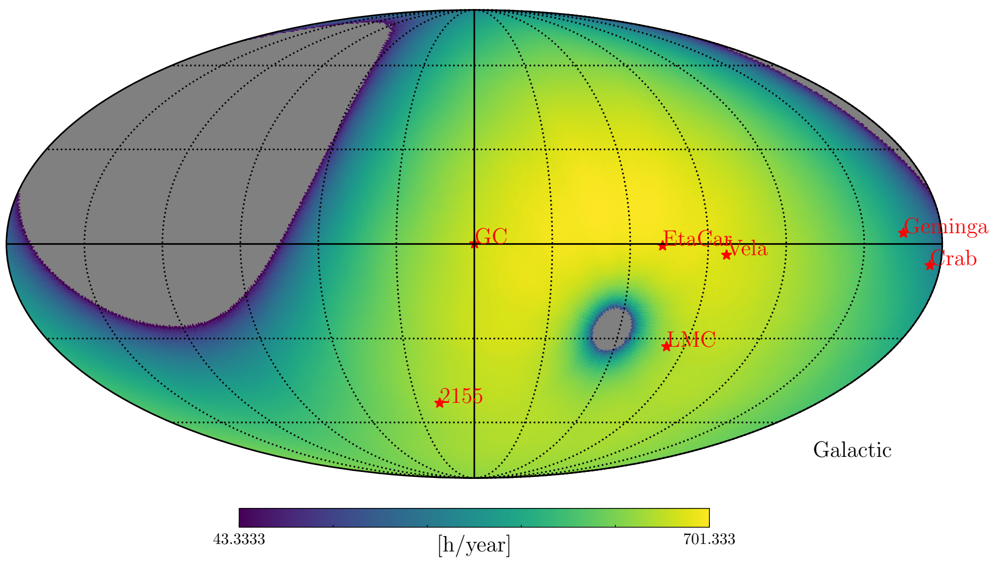
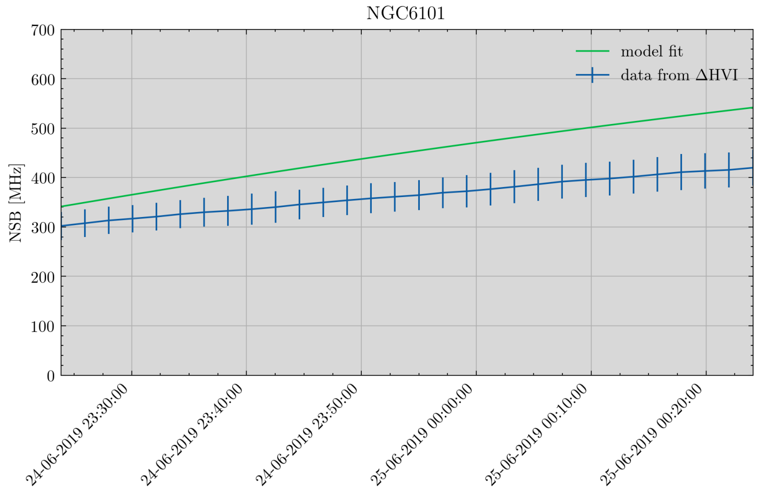
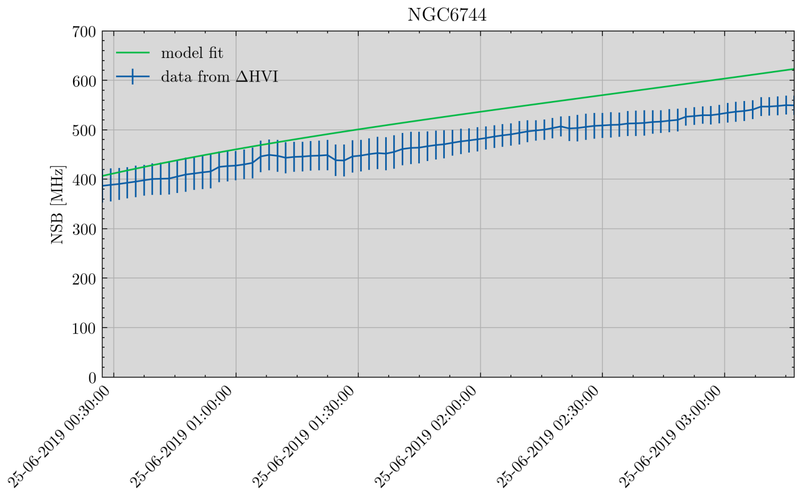


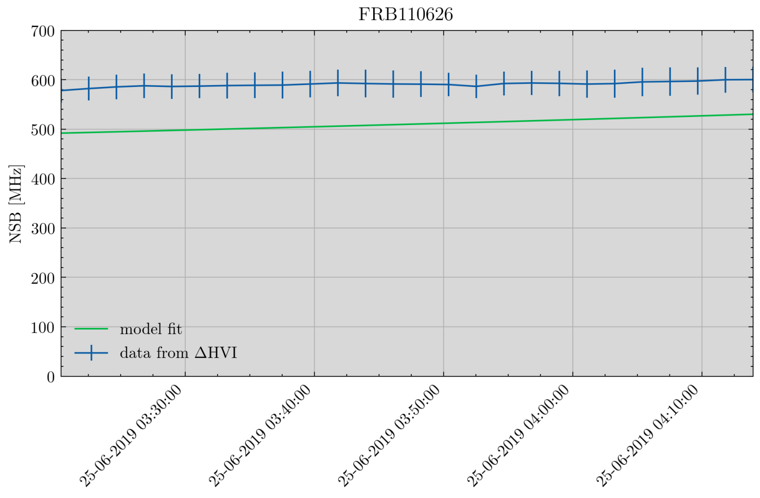
Figure 3.: The available dark time without moon per year for every HEALPix, when setting the minimum source altitude to 30°.



(a)

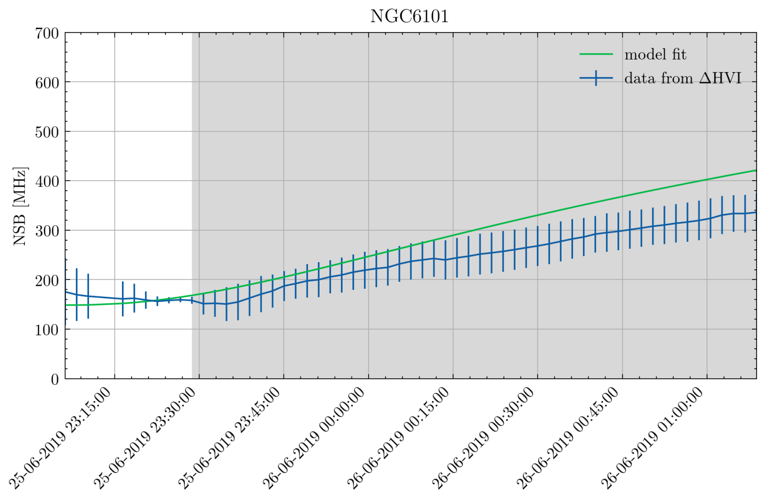


(b)

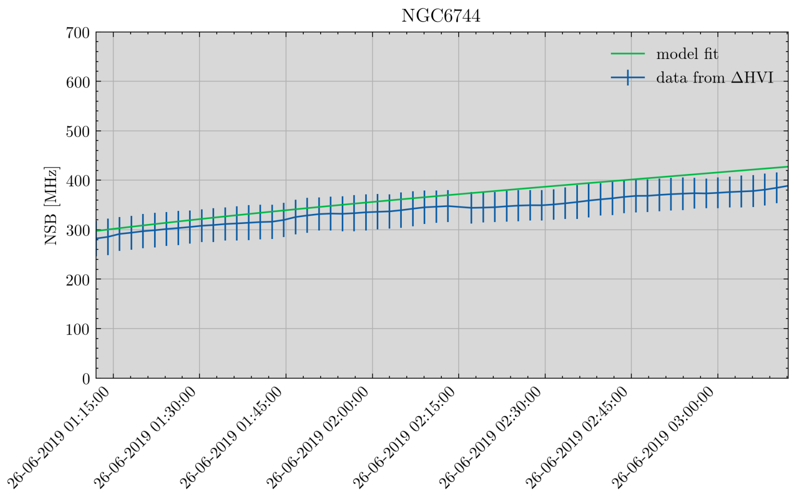


(c)

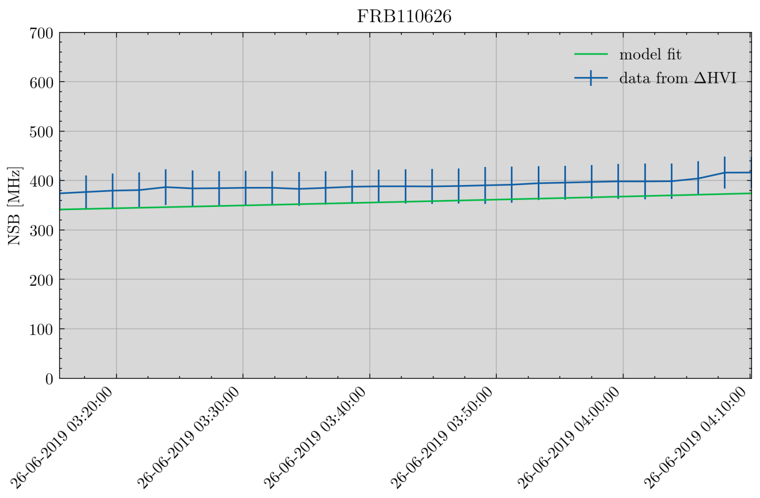
Figure 4.: NSB data and model fit (DWF first night). Times with moon above the horizon are shaded gray.



(a)

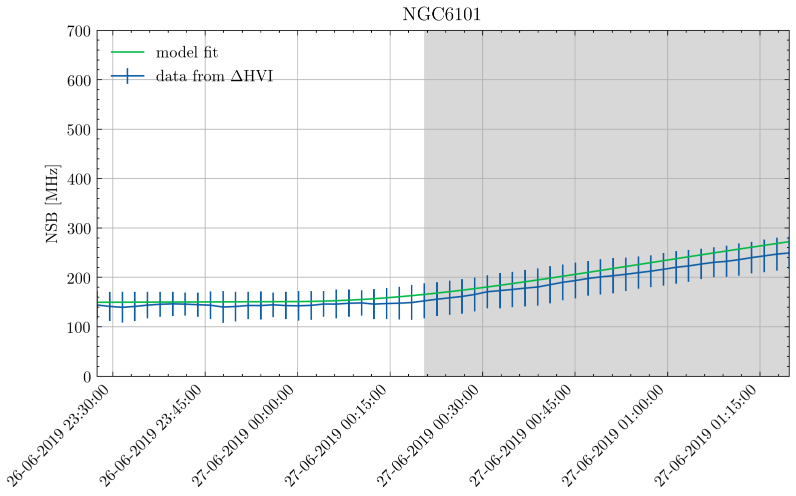


(b)

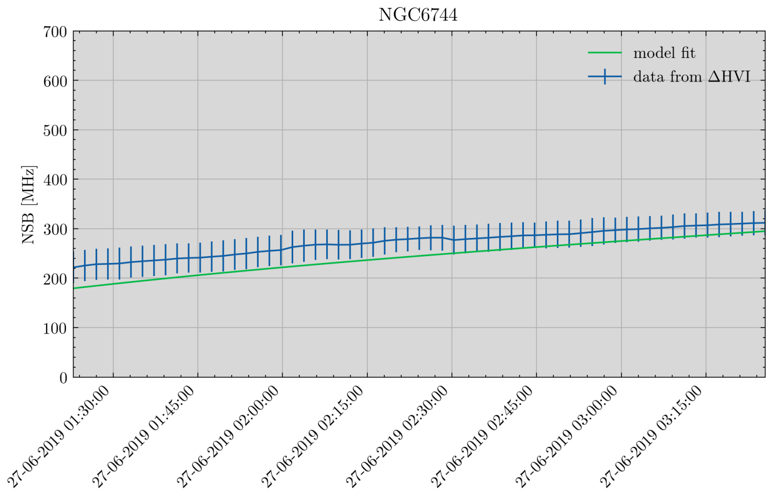


(c)

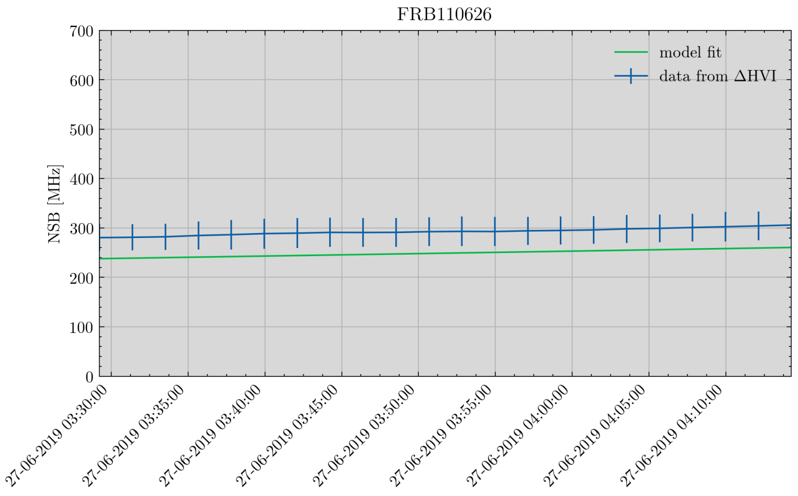
Figure 5.: NSB data and model fit (DWF second night). Times with moon above the horizon are shaded gray.



(a)

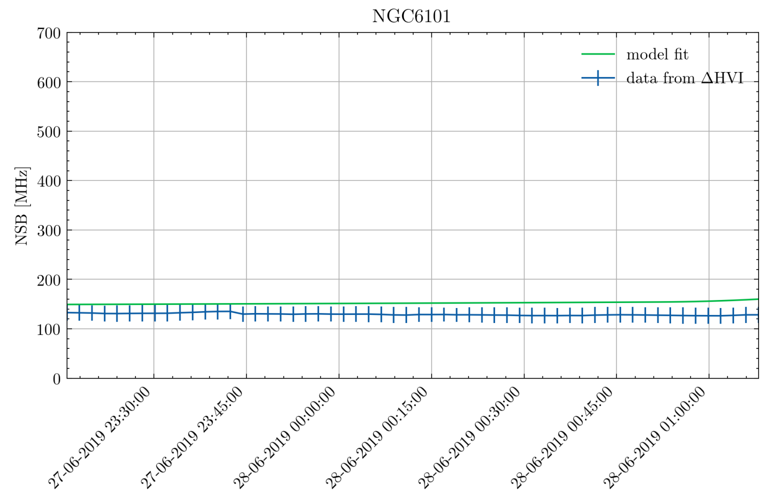


(b)

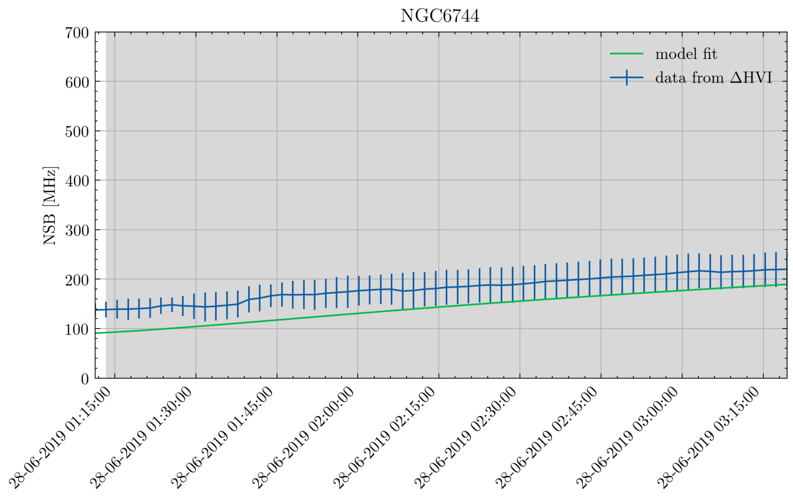


(c)

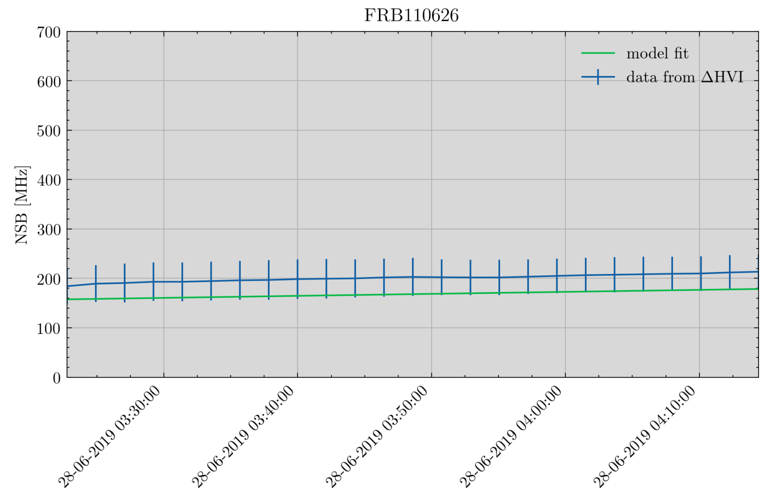
Figure 6.: NSB data and model fit (DWF third night). Times with moon above the horizon are shaded gray.



(a)

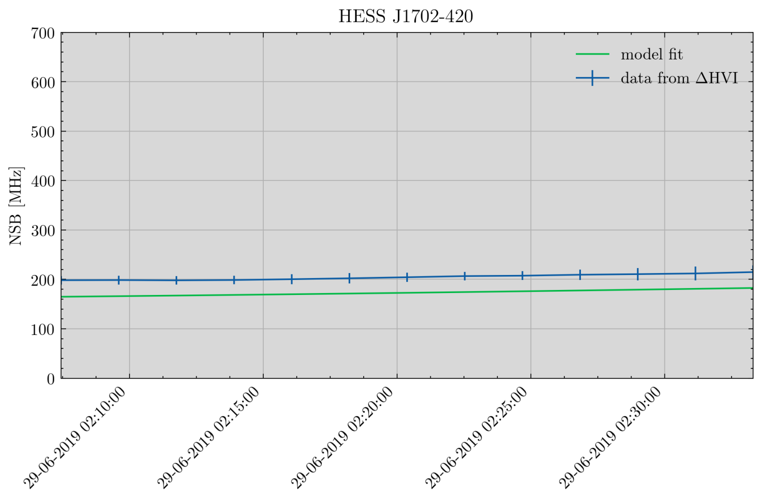


(b)

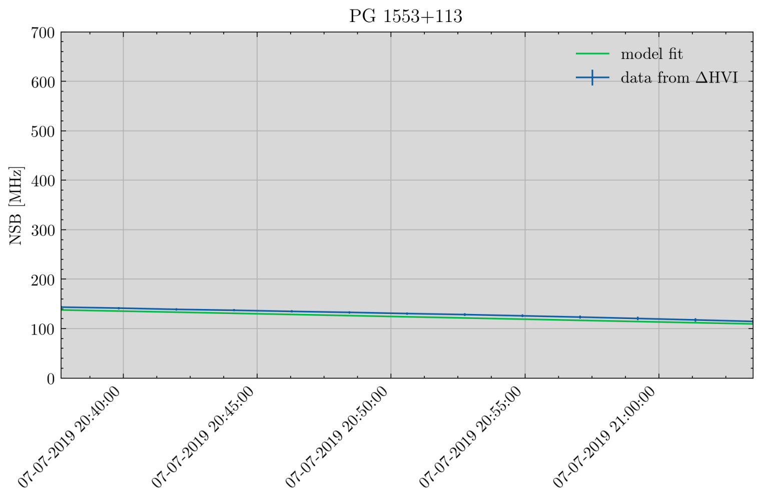


(c)

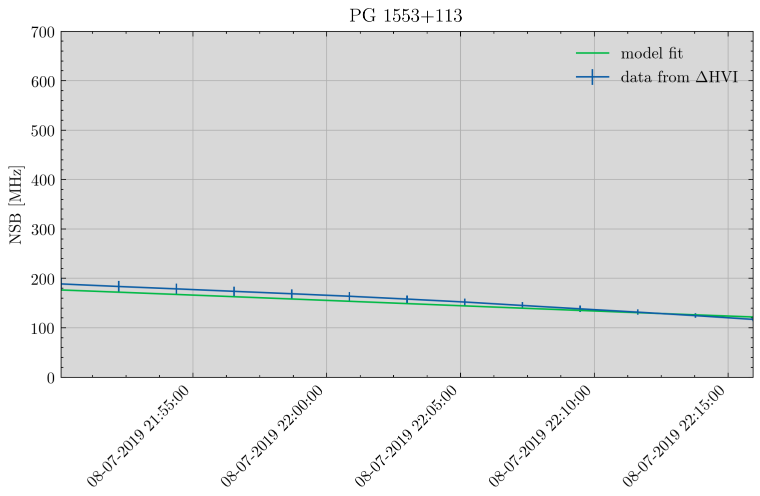
Figure 7.: NSB data and model fit (DWF fourth night). Times with moon above the horizon are shaded gray.



(a)

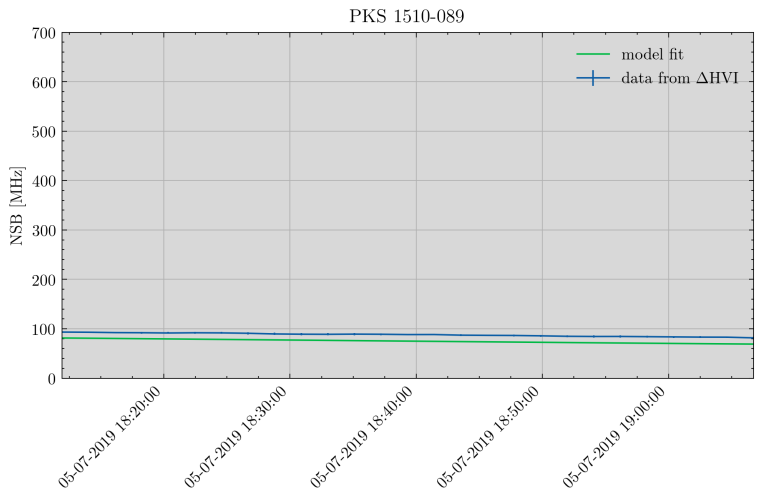


(b)

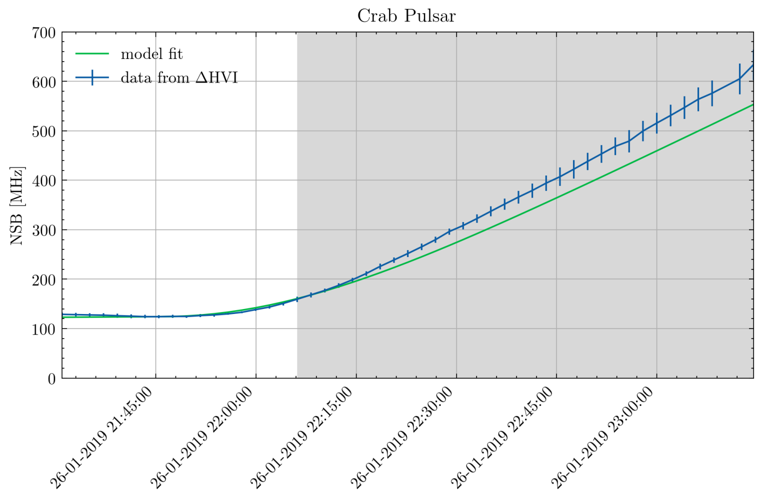


(c)

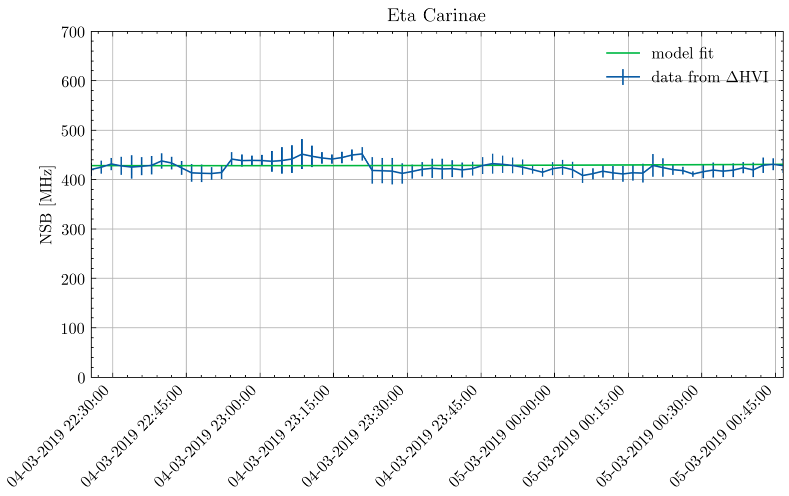
Figure 8.: NSB data and model fit (auxiliary data 1). Times with moon above the horizon are shaded gray.



(a)



(b)



(c)

Figure 9.: NSB data and model fit (auxiliary data 2). Times with moon above the horizon are shaded gray.

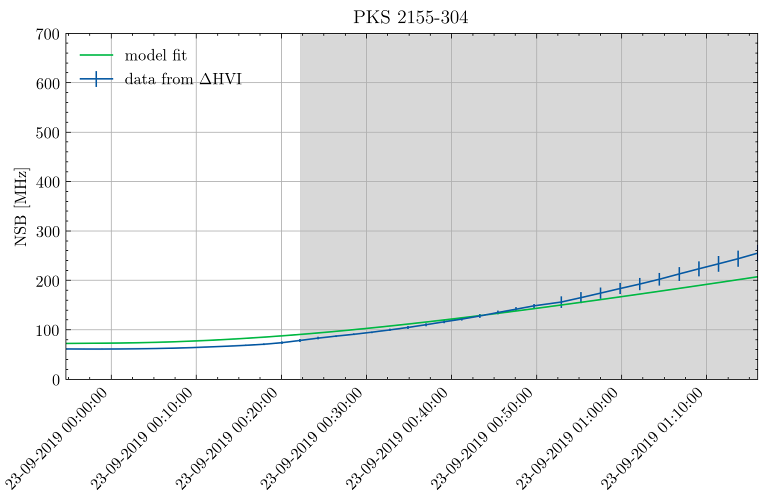
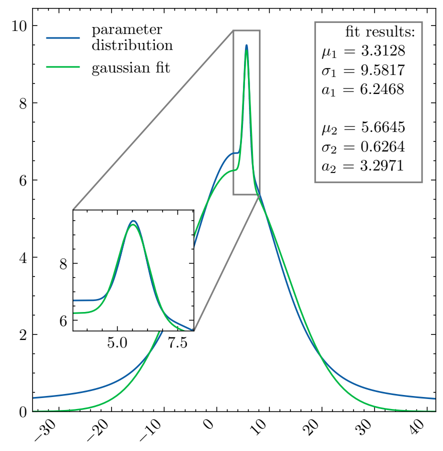
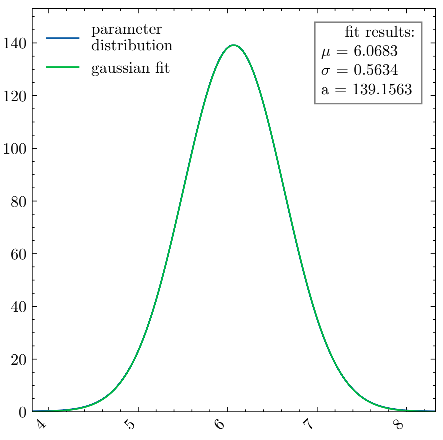


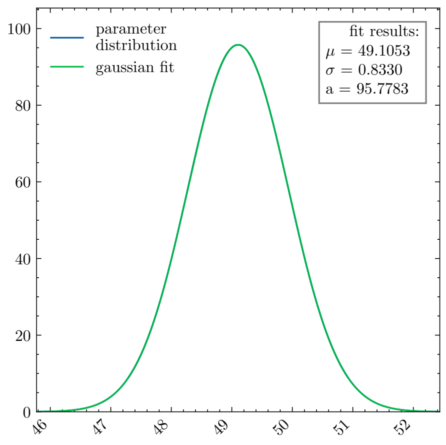
Figure 10.: NSB data and model fit (auxiliary data 3). Times with moon above the horizon are shaded gray.



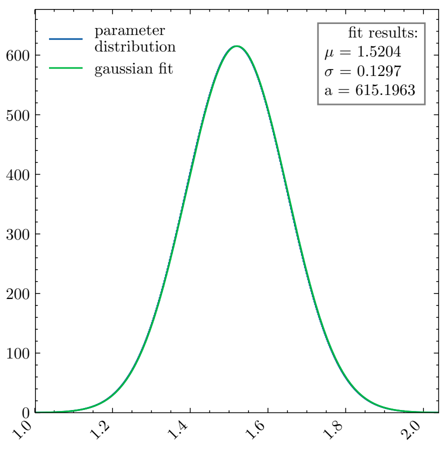
(a) Distribution of A



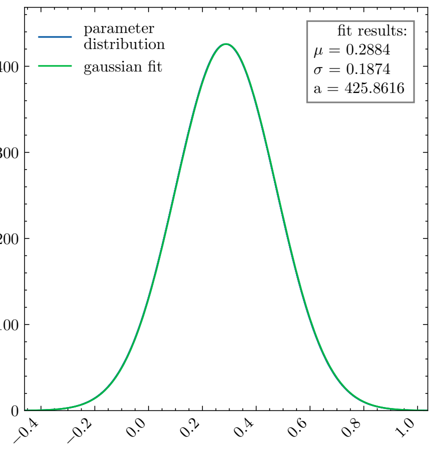
(b) Distribution of B



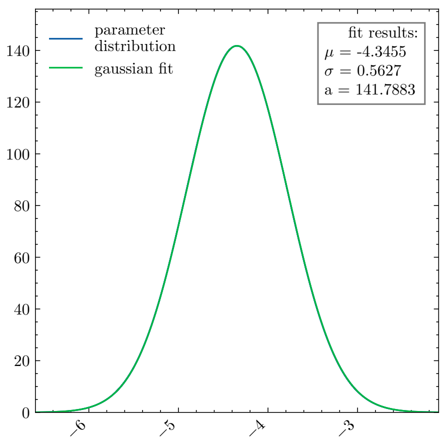
(c) Distribution of $p_0$



(d) Distribution of $p_2$

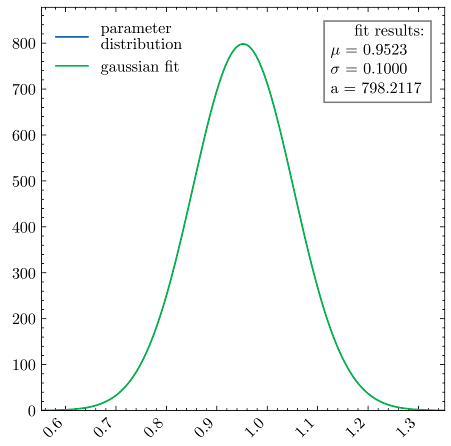


(e) Distribution of k

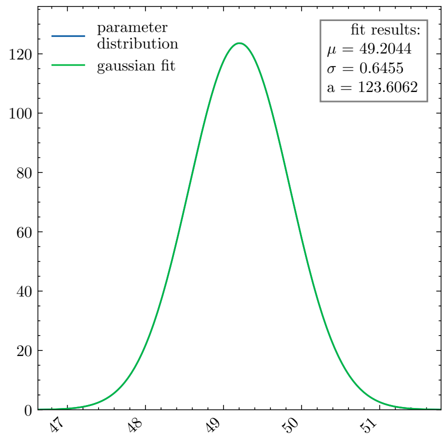


(f) Distribution of h

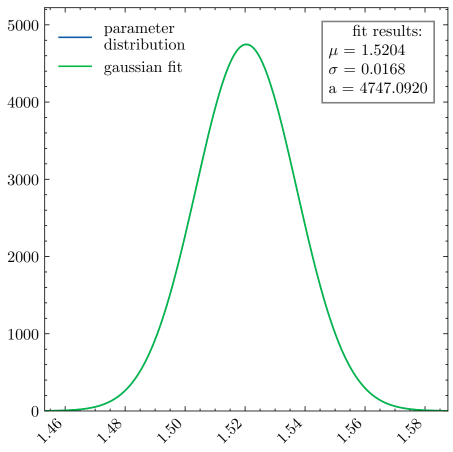
Figure 11.: Fit parameter distributions of the unstable, thus unused fit.



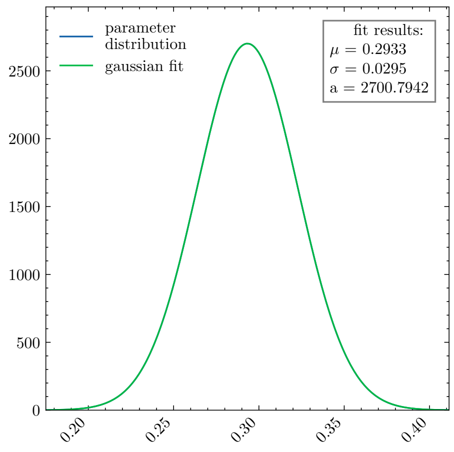
(a) Distribution of $p_1$



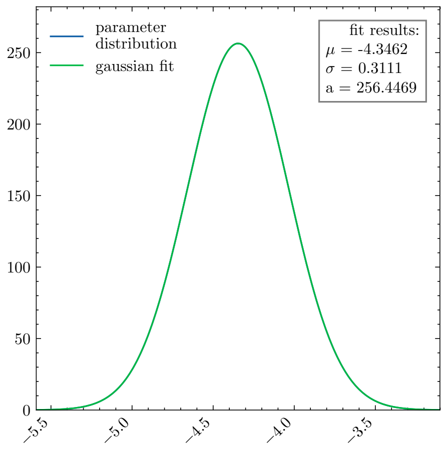
(b) Distribution of $p_0$



(c) Distribution of $p_2$

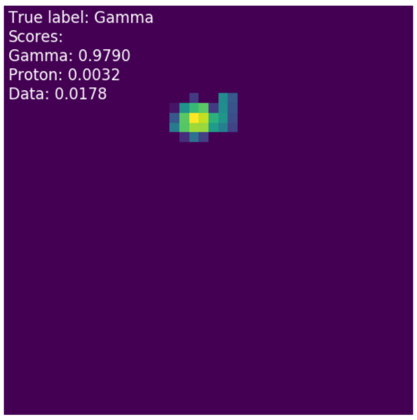


(d) Distribution of k

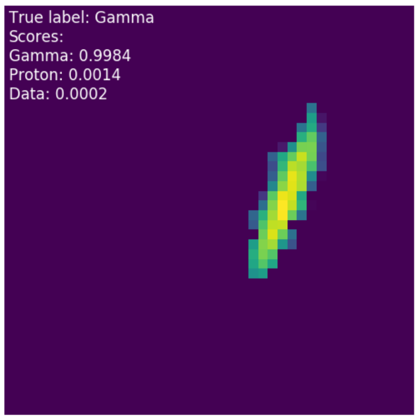


(e) Distribution of h

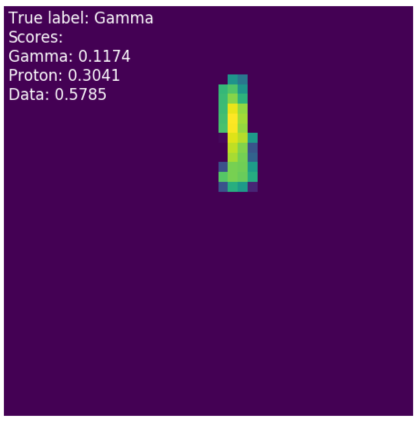
Figure 12.: Fit parameter distributions of the final, stable fit



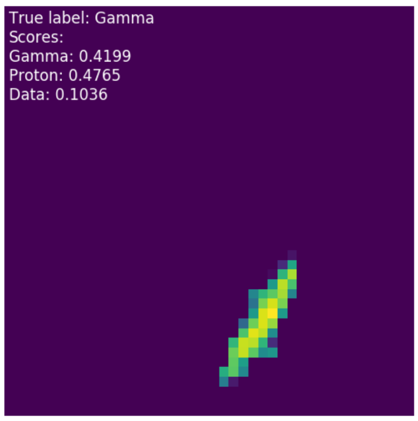
(a) correct prediction



(b) correct prediction

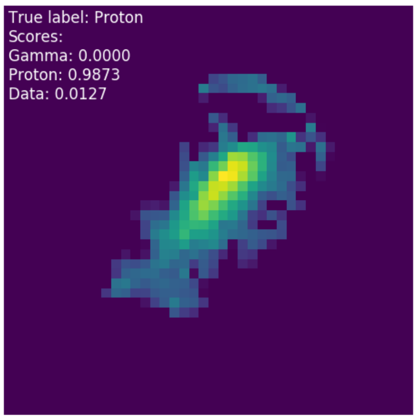


(c) wrong prediction

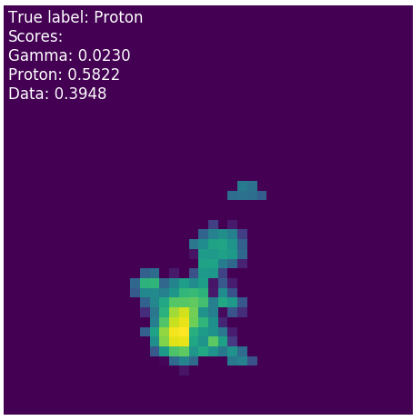


(d) wrong prediction

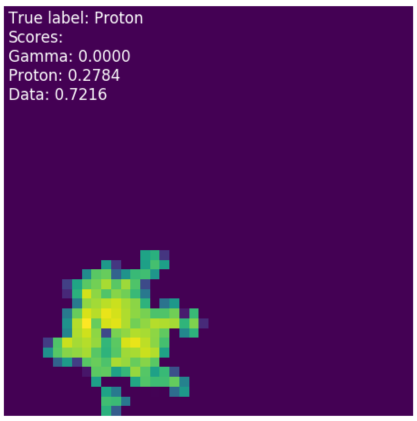
Figure 13.: Classification results of simulated gamma showers.



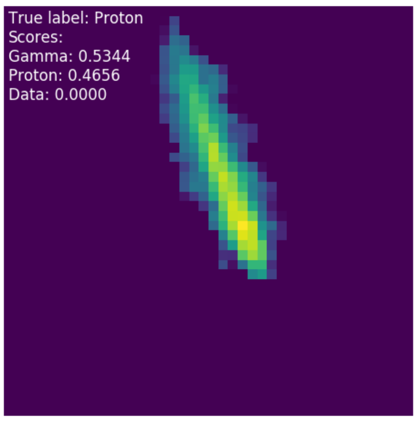
(a) correct prediction



(b) correct prediction

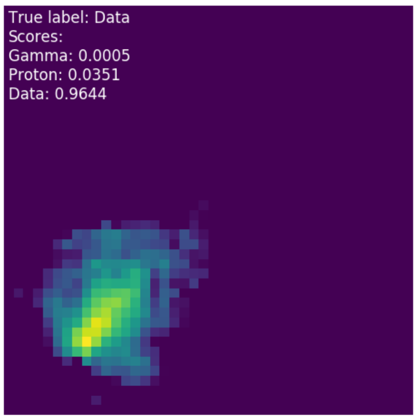


(c) wrong prediction

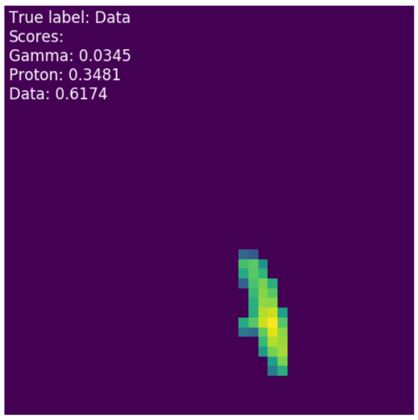


(d) wrong prediction

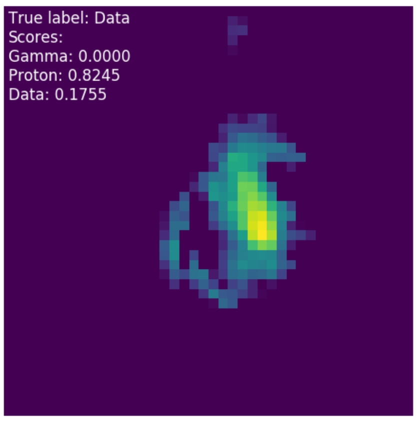
Figure 14.: Classification results of simulated proton showers.



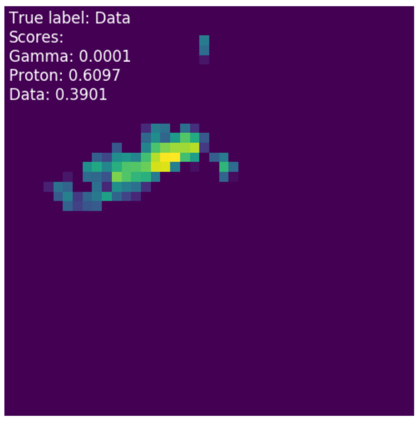
(a) correct prediction



(b) correct prediction



(c) wrong prediction



(d) wrong prediction

Figure 15.: Classification results of data showers.

# B. Tables

| run | target | date | duration [minutes] |
|---|---|---|---|
| 150635 | HESS J1702-420 | 29.06.2019 02:06 | 28.0 |
| 151093 | PKS 1510-089 | 05.07.2019 18:10 | 28.0 |
| 151096 | PKS 1510-089 | 05.07.2019 18:39 | 28.0 |
| 151172 | PKS 1510-089 | 07.07.2019 20:21 | 12.7 |
| 151174 | PKS 1510-089 | 07.07.2019 21:07 | 18.0 |
| 151173 | PG 1533+113 | 07.07.2019 20:36 | 28.0 |
| 151201 | PG 1533+113 | 08.07.2019 21:48 | 28.0 |
| 145077 | Crab Nebula | 26.01.2019 21:29 | 28.0 |
| 145082 | Crab Nebula | 26.01.2019 21:58 | 28.0 |
| 145083 | Crab Nebula | 26.01.2019 22:26 | 0.8 |
| 145084 | Crab Nebula | 26.01.2019 22:28 | 28.0 |
| 145086 | Crab Nebula | 26.01.2019 22:56 | 10.5 |
| 145087 | Crab Nebula | 26.01.2019 23:07 | 0.7 |
| 145088 | Crab Nebula | 26.01.2019 23:11 | 0.1 |
| 145089 | Crab Nebula | 26.01.2019 23:13 | 8.1 |
| 146753 | Eta Carinae | 04.03.2019 22:24 | 28.0 |
| 146754 | Eta Carinae | 04.03.2019 22:53 | 28.0 |
| 146755 | Eta Carinae | 04.03.2019 23:22 | 25.1 |
| 146756 | Eta Carinae | 04.03.2019 23:50 | 28.0 |
| 146757 | Eta Carinae | 05.03.2019 00:19 | 28.0 |
| 153235 | PSR J1723-2837 | 02.09.2019 17:59 | 28.0 |
| 153238 | PSR J1723-2837 | 02.09.2019 18:28 | 28.0 |
| 153245 | PSR J1723-2837 | 02.09.2019 18:57 | 28.0 |
| 153248 | PSR J1723-2837 | 02.09.2019 19:26 | 28.0 |
| 153250 | PSR J1723-2837 | 02.09.2019 19:54 | 11.2 |
| 153897 | PKS 2155-304 | 22.09.2019 23:53 | 28.0 |
| 153898 | PKS 2155-304 | 23.09.2019 00:22 | 28.0 |
| 153899 | PKS 2155-304 | 23.09.2019 00:51 | 25.4 |

Table 1.: Run list of model fit data part 1 (auxiliary)

| run | target | date | duration [minutes] |
|---|---|---|---|
| 150504 | NGC6101 | 24.06.2019 23:22 | 4.5 |
| 150505 | NGC6101 | 24.06.2019 23:28 | 28.0 |
| 150506 | NGC6101 | 24.06.2019 23:57 | 28.0 |
| 150507 | NGC6744 | 25.06.2019 00:26 | 28.0 |
| 150508 | NGC6744 | 25.06.2019 00:54 | 28.0 |
| 150509 | NGC6744 | 25.06.2019 01:23 | 28.0 |
| 150510 | NGC6744 | 25.06.2019 01:52 | 28.0 |
| 150511 | NGC6744 | 25.06.2019 02:21 | 28.0 |
| 150512 | NGC6744 | 25.06.2019 02:50 | 28.0 |
| 150513 | FRB110626 | 25.06.2019 03:19 | 28.0 |
| 150514 | FRB110626 | 25.06.2019 03:48 | 27.0 |
| 150534 | NGC6101 | 25.06.2019 23:05 | 5.0 |
| 150535 | NGC6101 | 25.06.2019 23:15 | 28.0 |
| 150536 | NGC6101 | 25.06.2019 23:44 | 28.0 |
| 150537 | NGC6101 | 26.06.2019 00:13 | 28.0 |
| 150538 | NGC6101 | 26.06.2019 00:41 | 28.0 |
| 150539 | NGC6744 | 26.06.2019 01:10 | 5.0 |
| 150540 | NGC6744 | 26.06.2019 01:16 | 28.0 |
| 150541 | NGC6744 | 26.06.2019 01:45 | 28.0 |
| 150542 | NGC6744 | 26.06.2019 02:16 | 28.0 |
| 150543 | NGC6744 | 26.06.2019 02:45 | 28.0 |
| 150544 | FRB110626 | 26.06.2019 03:14 | 28.0 |
| 150545 | FRB110626 | 26.06.2019 03:43 | 28.0 |
| 150568 | NGC6101 | 26.06.2019 23:26 | 28.0 |
| 150569 | NGC6101 | 26.06.2019 23:55 | 28.0 |
| 150570 | NGC6101 | 27.06.2019 00:23 | 28.0 |
| 150571 | NGC6101 | 27.06.2019 00:52 | 28.0 |
| 150572 | NGC6744 | 27.06.2019 01:21 | 10.0 |
| 150573 | NGC6744 | 27.06.2019 01:32 | 28.0 |
| 150574 | NGC6744 | 27.06.2019 02:01 | 28.0 |
| 150575 | NGC6744 | 27.06.2019 02:29 | 28.0 |
| 150576 | NGC6744 | 27.06.2019 02:58 | 28.0 |
| 150577 | FRB110626 | 27.06.2019 03:28 | 28.0 |
| 150578 | FRB110626 | 27.06.2019 03:57 | 17.8 |
| 150596 | NGC6101 | 27.06.2019 23:14 | 28.0 |
| 150597 | NGC6101 | 27.06.2019 23:43 | 28.0 |
| 150598 | NGC6101 | 28.06.2019 00:12 | 28.0 |
| 150599 | NGC6101 | 28.06.2019 00:40 | 28.0 |
| 150600 | NGC6744 | 28.06.2019 01:10 | 28.0 |
| 150601 | NGC6744 | 28.06.2019 01:39 | 28.0 |
| 150602 | NGC6744 | 28.06.2019 02:07 | 28.0 |
| 150603 | NGC6744 | 28.06.2019 02:36 | 28.0 |
| 150604 | NGC6744 | 28.06.2019 03:05 | 15.0 |
| 150605 | FRB110626 | 28.06.2019 03:21 | 28.0 |
| 150606 | FRB110626 | 28.06.2019 03:50 | 25.1 |

Table 2.: Run list of model fit data part 2 (DWF)

| Limit [MHz] | $T_m$ [h/year] | $T_d$ [h/year] | $T_{d'+m}$ [h/year] | $Q_m$ [C/year] | $Q_d$ [C/year] | $Q_{d'+m}$ [C/year] | $\tau$ | $T_m^*$ [h/year] |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.0 | 735.1 | 0.0 | 0.0 | 17.9 | 1.00 | 0.0 |
| 50 | 0.0 | 0.0 | 735.1 | 0.0 | 0.0 | 17.9 | 1.00 | 0.0 |
| 60 | 2.2 | 17.7 | 737.3 | 0.0 | 0.2 | 17.9 | 1.00 | 1.2 |
| 70 | 27.9 | 124.6 | 763.0 | 0.3 | 1.5 | 18.2 | 0.98 | 14.0 |
| 80 | 68.0 | 230.9 | 803.1 | 0.9 | 2.9 | 18.8 | 0.95 | 30.4 |
| 90 | 106.2 | 293.2 | 841.3 | 1.5 | 3.9 | 19.3 | 0.92 | 42.3 |
| 100 | 139.2 | 328.8 | 874.3 | 2.0 | 4.5 | 19.9 | 0.90 | 49.7 |
| 110 | 170.0 | 363.7 | 905.1 | 2.6 | 5.1 | 20.5 | 0.87 | 54.1 |
| 120 | 200.0 | 401.6 | 935.1 | 3.3 | 5.9 | 21.1 | 0.85 | 56.0 |
| 130 | 229.8 | 440.8 | 964.8 | 3.9 | 6.8 | 21.8 | 0.82 | 55.8 |
| 140 | 256.8 | 473.1 | 991.9 | 4.6 | 7.6 | 22.5 | 0.80 | 54.0 |
| 150 | 283.4 | 508.1 | 1018.4 | 5.3 | 8.5 | 23.2 | 0.77 | 50.6 |
| 160 | 308.4 | 535.8 | 1043.4 | 6.0 | 9.3 | 23.9 | 0.75 | 46.1 |
| 170 | 331.9 | 560.8 | 1067.0 | 6.7 | 10.1 | 24.6 | 0.73 | 40.8 |
| 180 | 353.7 | 580.6 | 1088.8 | 7.4 | 10.7 | 25.3 | 0.71 | 34.9 |
| 190 | 373.0 | 594.8 | 1108.1 | 8.0 | 11.2 | 25.9 | 0.69 | 28.9 |
| 200 | 391.8 | 610.9 | 1126.8 | 8.7 | 11.7 | 26.6 | 0.67 | 22.4 |
| 210 | 410.6 | 629.0 | 1145.7 | 9.4 | 12.4 | 27.3 | 0.65 | 15.3 |
| 220 | 428.8 | 644.2 | 1163.9 | 10.1 | 13.0 | 28.0 | 0.64 | 7.8 |
| 230 | 445.4 | 655.8 | 1180.4 | 10.8 | 13.5 | 28.7 | 0.62 | 0.6 |
| 240 | 461.9 | 669.1 | 1197.0 | 11.5 | 14.1 | 29.4 | 0.61 | -7.0 |
| 250 | 476.8 | 677.0 | 1211.9 | 12.2 | 14.4 | 30.0 | 0.59 | -14.3 |
| 260 | 490.0 | 682.6 | 1225.1 | 12.8 | 14.7 | 30.7 | 0.58 | -21.0 |
| 270 | 501.9 | 688.0 | 1237.0 | 13.4 | 14.9 | 31.2 | 0.57 | -27.3 |
| 280 | 513.3 | 692.9 | 1248.4 | 13.9 | 15.2 | 31.8 | 0.56 | -33.5 |
| 290 | 523.8 | 699.5 | 1258.9 | 14.5 | 15.5 | 32.3 | 0.55 | -39.5 |
| 300 | 534.4 | 705.3 | 1269.5 | 15.0 | 15.8 | 32.9 | 0.54 | -45.8 |
| 325 | 556.7 | 713.8 | 1291.7 | 16.3 | 16.3 | 34.2 | 0.52 | -59.6 |
| 350 | 575.6 | 719.8 | 1310.7 | 17.5 | 16.7 | 35.3 | 0.51 | -72.3 |
| 375 | 591.7 | 725.6 | 1326.8 | 18.5 | 17.1 | 36.4 | 0.49 | -83.7 |
| 400 | 605.1 | 728.9 | 1340.2 | 19.5 | 17.3 | 37.3 | 0.48 | -93.7 |
| 425 | 616.0 | 730.7 | 1351.0 | 20.3 | 17.4 | 38.2 | 0.47 | -102.3 |
| 450 | 625.3 | 732.9 | 1360.4 | 21.0 | 17.6 | 38.9 | 0.46 | -110.1 |
| 500 | 638.5 | 734.0 | 1373.6 | 22.2 | 17.7 | 40.0 | 0.45 | -122.0 |
| 550 | 648.5 | 734.3 | 1383.6 | 23.1 | 17.7 | 41.0 | 0.44 | -131.9 |
| 600 | 657.8 | 734.5 | 1392.9 | 24.1 | 17.7 | 42.0 | 0.43 | -141.9 |
| 750 | 679.8 | 734.7 | 1414.9 | 26.8 | 17.8 | 44.6 | 0.40 | -168.6 |
| 1000 | 700.9 | 735.0 | 1436.0 | 30.1 | 17.8 | 47.9 | 0.37 | -199.8 |
| 10000 | 723.0 | 735.1 | 1458.1 | 35.5 | 17.9 | 53.4 | 0.33 | -247.2 |
| 100000 | 723.0 | 735.1 | 1458.1 | 35.5 | 17.9 | 53.4 | 0.33 | -247.2 |

Table 3.: Summary of all calculated values of the NSB chapter.

| Component | Details |
|---|---|
| Mainboard | Fujitsu (R) D3417-A2 |
| CPU | Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, 4 cores / 8 threads |
| RAM | $4 \times 16$ GB DDR4 @ 2133 MHz |
| ROM | $3 \times 2$ TB HDD |
|  | $2 \times 128$ GB NVMe SSD in RAID 1 |
| GPU | NVIDIA (R) GeForce GTX 1080 with 8 GB VRAM |

Table 4.: Specifications of the system used for all necessary computations on the NSB model and the machine learning tasks

# Bibliography

Abadi, M., A. Agarwal, P. Barham, et al.
2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abraham, J., , et al.
2010. Measurement of the depth of maximum of extensive air showers above $10^{18}$ eV. *Phys. Rev. Lett.*, 104:091101.

Achterberg, A., Y. A. Gallant, J. G. Kirk, and A. W. Guthmann
2001. Particle acceleration by ultrarelativistic shocks: theory and simulations. *Monthly Notices of the Royal Astronomical Society*, 328(2):393–408.

Aharonian, F., A. Akhperjanian, K.-M. Aye, et al.
2004. Calibration of cameras of the h.e.s.s. detector. *Astroparticle Physics*, 22(2):109 – 125.

Aharonian, F., A. G. Akhperjanian, A. Bazer-Bachi, et al.
2007a. Hess observations of the supernova remnant rx j0852. 0–4622: Shell-type morphology and spectrum of a widely extended very high energy gamma-ray source. *The Astrophysical Journal*, 661(1):236.

Aharonian, F., A. G. Akhperjanian, A. R. Bazer-Bachi, et al.
2007b. An exceptional very high energy gamma-ray flare of PKS 2155-304. *The Astrophysical Journal*, 664(2):L71–L74.

Ahnen, M., S. Ansoldi, L. Antonelli, et al.
2017. Performance of the magic telescopes under moonlight. *Astroparticle Physics*, 94:29 – 41.

Andreoni, I. and J. Cooke
2017. The deeper wider faster programme: Chasing the fastest bursts in the universe. *Proceedings of the International Astronomical Union*, 14(S339):135–138.

Archambault, S. et al.
2017. Gamma-ray observations under bright moonlight with VERITAS. *Astroparticle Physics*, 91:34–43.

Bell, A.
2013. Cosmic ray acceleration. *Astroparticle Physics*, 43:56 – 70. Seeing the High-Energy Universe with the Cherenkov Telescope Array - The Science Explored with the CTA.

Berge, D., S. Funk, and J. Hinton
2007. Background modelling in very-high-energy $\gamma$-ray astronomy. *Astronomy and Astrophysics*, 466(3):1219–1229.

Bernlohr, K.
2008. Simulation of Imaging Atmospheric Cherenkov Telescopes with CORSIKA and sim_telarray. *Astropart. Phys.*, 30:149–158.

Branch, M. A., T. F. Coleman, and Y. Li
1999. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23.

Bryson, A. and Y. Ho
1969. *Applied optimal control: optimization, estimation, and control*, Blaisdell book in the pure and applied sciences. Blaisdell Pub. Co.

Bykov, A. M., D. C. Ellison, A. Marcowith, and S. M. Osipov
2018. Cosmic ray production in supernovae. *Space Science Reviews*, 214(1):41.

de Naurois, M.
2012. *Very High Energy astronomy from H.E.S.S. to CTA. Opening of a new astronomical window on the non-thermal Universe.* Habilitation à diriger des recherches, Université Pierre et Marie Curie - Paris VI.

de Naurois, M. and L. Rolland
2009. A high performance likelihood reconstruction of $\gamma$-rays for imaging atmospheric cherenkov telescopes. *Astroparticle Physics*, 32(5):231 – 252.

Fermi, E.
1949. On the origin of the cosmic radiation. *Phys. Rev.*, 75:1169–1174.

Gaia Collaboration et al.
2016. Gaia data release 1 - summary of the astrometric, photometric, and survey properties. *A&A*, 595.

Giavitto, G., T. Ashton, A. Balzer, et al.
2015. A major electronics upgrade for the hess cherenkov telescopes 1-4. *arXiv preprint arXiv:1509.01232*.

Gorski, K. M., B. D. Wandelt, E. Hivon, and others.
1999. The healpix primer. *arXiv preprint arXiv:astro-ph/9905275.*

Hand, D. J. and R. J. Till
2001. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186.

Hanley, J. A. and B. J. McNeil
1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36. PMID: 7063747.

Hebb, D. O.
1949. *The Organization of Behaviour.* New York: Wiley & Sons.

Heck, D., J. Knapp, J. N. Capdevielle, et al.
1998. *CORSIKA: a Monte Carlo code to simulate extensive air showers.*

Hess, V. F.
1912. Über beobachtungen der durchdringenden strahlung bei sieben freiballonfahrten. *Physikalische Zeitschrift*, 13:1084–1091.

H.E.S.S. Collaboration et al.
2018. Deeper h.e.s.s. observations of vela junior (rx j0852.0-4622): Morphology studies and resolved spectroscopy. *Astronomy Astrophysics*, 612:A7.

H.E.S.S. Collaboration et al.
2020. Detection of very-high-energy emission from the colliding wind binary with h.e.s.s. *Astronomy and Astrophysics*, 635:A167.

Hillas, A. M.
1985. Cerenkov Light Images of EAS Produced by Primary Gamma Rays and by Nuclei. In *19th International Cosmic Ray Conference (ICRC19), Volume 3*, volume 3 of *International Cosmic Ray Conference*, P. 445.

Hillig, C.
2019. Analyzing the cosmic ray spectrum based on imaging atmospheric cherenkov telescope data using deep learning methods. *Friedrich-Alexander-Universität Erlangen-Nürnberg.*

Hofmann, W.
1997. Measuring gamma-ray energy spectra with the hegra iact system. *arXiv preprint arXiv:9710297.*

Holch, T. L., I. Shilon, M. Büchele, et al.
2018. Probing Convolutional Neural Networks for Event Reconstruction in $\gamma$-Ray Astronomy with Cherenkov Telescopes. *PoS*, ICRC2017:795.

Knapp, J., D. Heck, and G. Schatz
1996. Comparison of hadronic interaction models used in air shower simulations and of their influence on shower development and observables.

Krisciunas, K. and B. E. Schaefer
1991. A model of the brightness of moonlight. *Publications of the Astronomical Society of the Pacific*, 103(667):1033.

LeCun, Y.
1985. Une procedure d'apprentissage pour reseau a seuil asymmetrique (a learning scheme for asymmetric threshold networks). In *Proceedings of Cognitiva 85, Paris, France*, Pp. 599–604.

LeCun, Y., B. Boser, J. S. Denker, et al.
1989. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551.

Li, T. P. and Y. Q. Ma
1983. Analysis methods for results in gamma-ray astronomy. *The Astrophysical Journal*, 272:317–324.

Matthews, J.
2004. A heitler model of extensive air showers. *Astroparticle Physics*, 22:387–397.

McMillan, P. J.
2011. Mass models of the Milky Way. *Monthly Notices of the Royal Astronomical Society*, 414(3):2446–2457.

Mirzoyan, R. et al.
2019. First time detection of a grb at sub-tev energies; magic detects the grb 190114c. *The Astronomer's Telegram: ATel 12390.*

Mitchel, T. M.
1997. *Machine Learning.* McGraw-Hill Science/Engineering/Math.

Nievas, M.
2013. Absolute photometry and night sky brightness with all-sky cameras. *Departamento de Astrofísica y Ciencias de la Atmósfera UNIVERSIDAD COMPLUTENSE DE MADRID.*

Parsons, R. and J. Hinton
2014. A monte carlo template based analysis for air-cherenkov arrays. *Astroparticle Physics*, 56:26 – 34.

Parsons, R. D., C. Bleve, S. S. Ostapchenko, and J. Knapp
2011. Systematic uncertainties in air shower measurements from high-energy hadronic interaction models. *Astroparticle Physics*, 34(11):832–839.

Patrignani, C. et al. (Particle Data Group)
2016. 2017 review of particle physics. *Chin. Phys.*, 40.

Rosenblatt, F.
1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Reviews*, 65:386–408.

Schwarzschild, B.
2010. The highest-energy cosmic rays may be iron nuclei. *Physics Today*, 63.

Shilon, I., M. Kraus, M. Büchele, et al.
2019. Application of deep learning methods to analysis of imaging atmospheric cherenkov telescopes data. *Astroparticle Physics*, 105:44 – 53.

Steppa, C. and T. L. Holch
2019. Hexagdly—processing hexagonally sampled data with cnns in pytorch. *SoftwareX*, 9:193 – 198.

The Pierre Auger Collaboration et al.
2007. Correlation of the highest-energy cosmic rays with nearby extragalactic objects. *Science*, 318(5852):938–943.

Veh, J.
2018. *Dark Matter $\gamma$-line search in the Galactic Centre with H.E.S.S. using an On-Off technique*. Phd thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg.

Wulf, T.
1910. Observations on the radiation of high penetration power on the eiffel tower. *Physikalische Zeitschrift*, 11:811.

# List of Figures

# Acknowledgments

Sieht so aus als hätten wir es geschafft. Wir, weil so eine Arbeit sich nicht von alleine schreibt. Also ja, das schreiben blieb einzig und alleine mir überlassen, aber die Arbeit hinter der Arbeit ist eine Aufgabe, bei der man Hilfe aus allen Richtungen dankend annimmt. Deshalb geht mein Dank in keiner besonderen Reihenfolge an

Prof. Dr. Stefan Funk für sein fortlaufendes Vertrauen in mich und meine Arbeit, für die zahlreichen und hilfreichen Diskussionen egal ob bei dir im Büro, bei mir oder am Flur. Nie vergessen werde ich, wie du durch den obersten Flur am ECAP schlenderst, dein Blick durch unsere offene Bürotür fällt und du plötzlich stehen bleibst mit einem "Ha! Matthias, ich wollte dich fragen: Wie steht's denn eigentlich um...". Die Mischung aus Freiraum und Vorgabe war stets wohl getroffen und hat mir immer geholfen.

dich Peter. Ich hätte mir keinen besseren Büropartner wünschen können. Als wir gemeinsam angefangen haben hatten wir nicht einmal ordentliche Möbel in unserem Büro. Über die Zeit haben wir so viel mehr daraus gemacht als nur einen Arbeitsplatz und einen Ort bzw. eine Atmosphäre geschaffen die ihresgleichen sucht. Du warst ein großartiger Kollege und bist und bleibst ein großartiger Freund.

die gesamte Gamma-Gruppe des ECAP. Ihr seid ein mega Haufen mit nahezu familiärer Umgebung. Danke euch allen für die schöne Zeit und eure Unterstützung bei Problemen egal ob groß oder klein. Ein besonderer Dank geht noch an David als meinen längsten Begleiter vom Beginn des Studiums bis weit darüber hinaus. Unsere gemeinsame Zeit (und vor allem Namibia) werde ich immer in guter Erinnerung behalten. Zu guter Letzt danke ich noch Johannes V. als universal-Ansprechpartner in Fragen HAP, Datenbanken, woody, Mac-Mini und allen weiteren Lebenslagen. Auf dich war immer Verlass, das schätze ich sehr.

meinen Mitstreitern im Bereich machine learning. Allen voran Idan Shilon. Danke für deine Unterstützung und dafür, dass du mir die Möglichkeit gegeben hast in dieses Thema abzutauchen, als die Kollaboration noch nicht bereit war für den Mond. Des weiteren natürlich Manuel und Tobias, die unser "hessnn" Team vollendet haben und mit denen es mir stets eine Freude war gemeinsam unsere Grafikkarten zu quälen.

meine beiden Master Studentinnen Christina und Jacky. Es war mir eine Freude euch betreuen zu dürfen und die enge Zusammenarbeit habe ich sehr geschätzt.

die Kolleginnen und Kollegen der H.E.S.S. Kollaboration. Von Anfang bis Ende hatte ich immer das Gefühl mit euch in einem Boot zu sitzen. Ich habe mich jedes Mal auf die halbjährlichen Collaboration-Meetings gefreut, sowohl für den fachlichen als auch den zwischenmenschlichen Austausch.

meine Familie. Dafür, dass ihr mich immer unterstützt habt, auch wenn's mal wieder etwas länger gedauert hat.

meine Verena. Wo du bist ist mein zu Hause. Danke für dein Vertrauen, deine Liebe und dass du immer für mich da bist. 49.01222°N 12.09882°E *Crazy Love.*

*" … They told me: You cannot simply go to Namibia and point [...] at the full moon!*

*So I went and did it… "*