Search for Ultra-High Energy Photons with the Pierre Auger Observatory using Deep Learning Techniques

von

Tobias Alexander Pan

Masterarbeit in Physik

vorgelegt der

Fakultät für Mathematik, Informatik und Naturwissenschaften

der RWTH Aachen

im Februar 2020

angefertigt im

III. Physikalischen Institut A

bei

Jun.-Prof. Dr. Thomas Bretz
Prof. Dr. Thomas Hebbeker

Contents

Chapter 1

Introduction

With energies even above 10^{20} eV [1], ultra-high energy cosmic rays (UHECRs) are the most energetic particles observed in nature. They were discovered in 1912 by Victor Hess and have since then been studied in great detail over many orders of magnitude in energy. Important questions such as where the particles come from or their composition at the highest energies are still unsolved. Although most cosmic rays are nuclei, especially protons, there are also some other particle types like photons. The most energetic photons that have been discovered so far have energies below 1 PeV [2] and no ultra-high energy photon (UHEP) with energies above 10^{17} eV has been found yet.

When interacting with the Earth's atmosphere, cosmic rays produce secondary particles inducing large cascades, called extensive air showers (EASs), which can spread over several kilometers in width and reach the surface, where they can be detected. Due to this reaction, cosmic rays can only be detected directly in space, but at the highest energies the incoming flux is too low so that the arrangement of a giant detector in space would be necessary, which is not possible today. Instead, UHECRs are detected indirectly by measuring only their remnants in the form of EASs. Nevertheless, there are different ways to measure EASs. The Pierre Auger Observatory is the largest cosmic ray observatory in the world which is able to detect EASs induced by cosmic rays with energies starting at 10^{17} eV [3]. It uses a grid of water Cherenkov detectors (WCDs), called the surface detector (SD), to detect particles which hit the ground, and multiple fluorescence telescopes, called the fluorescence detector (FD), which can detect the fluorescence light emitted by nitrogen atoms which were excited by the air shower. On the basis of these measurements it is possible to reconstruct properties of the primary particle, such as the energy and the arrival direction. In this thesis, methods are presented which allow UHEPs to be found in the background of UHECRs using simulated events of the Pierre Auger Observatory.

In recent years, machine learning techniques like boosted decision trees (BDTs) were used for this task. Although these methods achieve good precision, no event from data of the Pierre Auger Observatory has so far been classified as a photon. New methods with higher sensitivity thus have the potential of detecting a UHEP for the first time or lowering the upper limits of the photon flux. One part of the wide field of machine learning covers deep learning techniques using deep neural networks (DNNs), which have made rapid progress and achieved great successes in the last few years, for example in image recognition [4]. More and more applications in physics research also show that they are a useful tool, for example in reconstruction tasks of EASs at the Pierre Auger Observatory [5]. The reconstruction of air showers on the basis of numerous detector measurements with spatially and temporally interrelated connections is a very challenging task, which is why the development of conventional analytical evaluations can be very complicated. Once the training of the DNN is completed, deep learning methods have the potential to solve such complex problems at low computational costs, using all measurements available and, therefore, in the best case extract all possible information from the data. The first step is to train the DNN on simulated data whose true Monte Carlo output is known. Afterwards, the trained DNN can be applied to measured data.

In this thesis, the design of a DNN for the distinction between primary photons and protons is presented. Different approaches are discussed and compared to a previous analysis, where a BDT was used [6, 7]. Finally, the impact of these results on the photon flux estimation is discussed.

Chapter 2

Cosmic rays

2.1 Cosmic ray physics

The spectrum of cosmic rays exceeds eleven orders of magnitude in energy as can be seen in figure 2.1. The cosmic ray flux over this energy range can be described by a power law $\Phi \propto E^{-\gamma}$ with

$$\gamma \approx \begin{cases} 2.7 & E \lesssim 4\,\mathrm{PeV} \\ 3 & 4\,\mathrm{PeV} \lesssim E \lesssim 3\,\mathrm{EeV} \\ 2.7 & 3\,\mathrm{EeV} \lesssim E \lesssim 60\,\mathrm{EeV} \\ 4 & 60\,\mathrm{EeV} \lesssim E \end{cases} \tag{2.1}$$

starting at high fluxes of $\sim 10^4\,\mathrm{m^{-2}s^{-1}}$ at 1 GeV and ending at extremely low fluxes of $\sim 1\,\mathrm{km^{-2}\ century^{-1}}$ at $10^{20}\,\mathrm{eV}$. Cosmic rays below the so-called "knee", at 4 PeV [8], are assumed to have a galactic origin whereas cosmic rays above the so-called "ankle", at 3 EeV [9], are assumed to be extragalactic. This occurs because the deflection of the cosmic ray due to galactic magnetic fields becomes too weak to bind it within the galaxy. This effect is depending on the charge of the particle. Above 60 EeV, the spectrum strongly cuts off [9]. A possible explanation for this might be the Greisen-Zatsepin-Kuzmin (GZK) effect [10, 11] which will be explained below.

It is not completely clear yet how cosmic rays achieve such energies, although there are already many models. For example, Fermi acceleration is a mechanism describing the acceleration of charged particles in astrophysical shock waves, like in supernovae, where the particles are scattered in turbulent magnetic fields [12]. The most likely source candidates for this shock acceleration are gamma ray bursts, active galactic nuclei, pulsars and the lobes of giant radio galaxies.

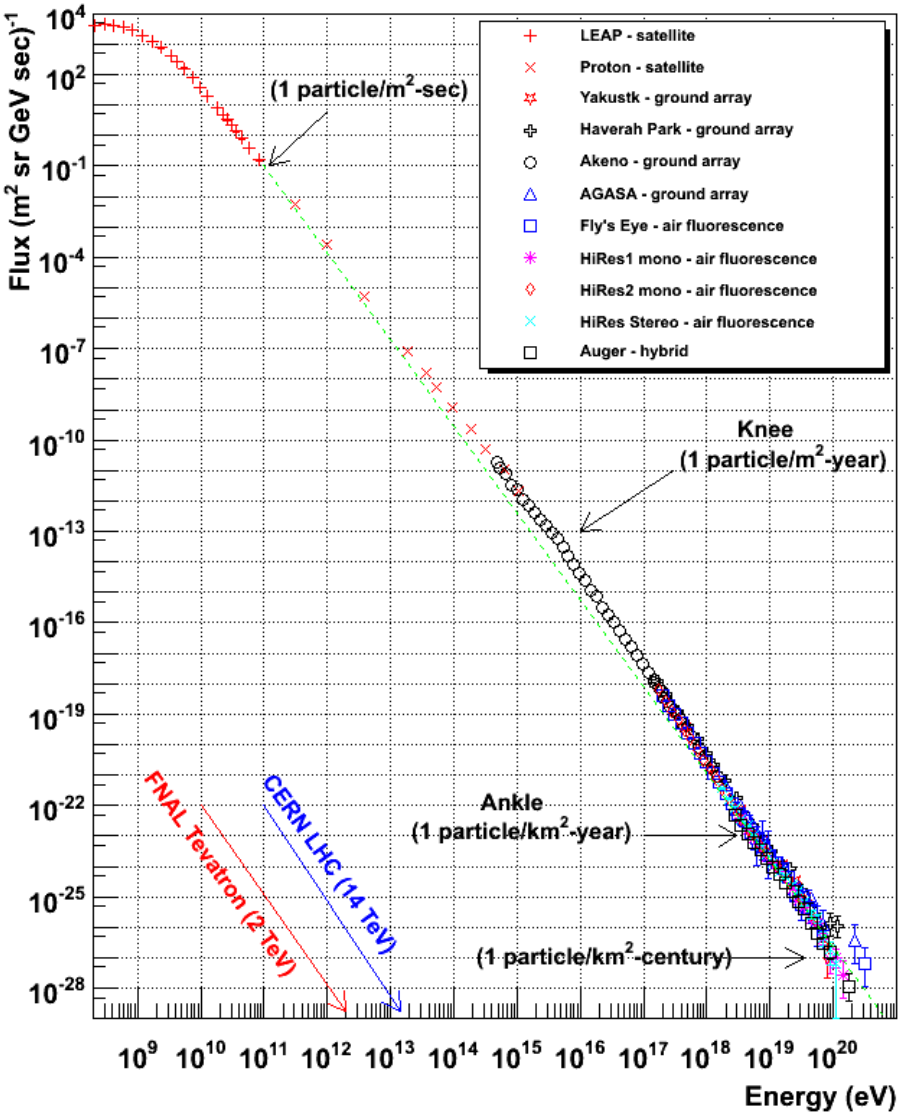


FIGURE 2.1: Cosmic ray flux as a function of energy, starting at 200 MeV up to 1 ZeV, measured by various experiments. Taken from [13].

Even though there are also many open questions regarding the composition of cosmic rays, it is known that $\sim$ 99% of all cosmic rays are nuclei and only $\sim$ 1% other particles like electrons or photons. With 90%, protons make up the largest part of nuclei, followed by helium (9%), and only 1% are heavier elements.

Due to the deflection by galactic magnetic fields, charged particles lose their directional information which is why neutral particles are essential to identify the cosmic ray sources. Neutrons decay to protons during propagation through space, and neutrinos are very hard to detect so that photons are very important for arrival direction studies and point sources analyses.

The production of UHEPs is assumed to be linked to the GZK effect [10, 11], which might be the cause for the cosmic ray cut-off. At $\sim 60\,\mathrm{EeV}$, protons start to interact with cosmic microwave background (CMB) photons ($E \sim 10^{-3}\,\mathrm{eV}$) and produce Δ^+ baryons which then decay into pions and nucleons:

$$\begin{aligned} p^+ + \gamma_{\mathrm{CMB}} \rightarrow \Delta^+ &\rightarrow p^+ + \pi^0 \\ &\rightarrow n + \pi^+ \end{aligned}$$

Ultra-high energy photons are then produced due to the decay

$$\pi^0 \rightarrow 2\gamma$$

with $\sim 10\%$ of the energy of the incoming proton.

Until now, no UHEP has been detected by the Pierre Auger Observatory. Many other production models such as super heavy dark matter, topological defect or Z-burst models [14] can already be excluded as a result of the current upper limits for the photon flux. The current upper limits of the photon flux can be seen in figure 2.2. Currently, the results cannot confirm or disprove the GZK effect, which is why improved methods to identify UHEPs are presented in this thesis.

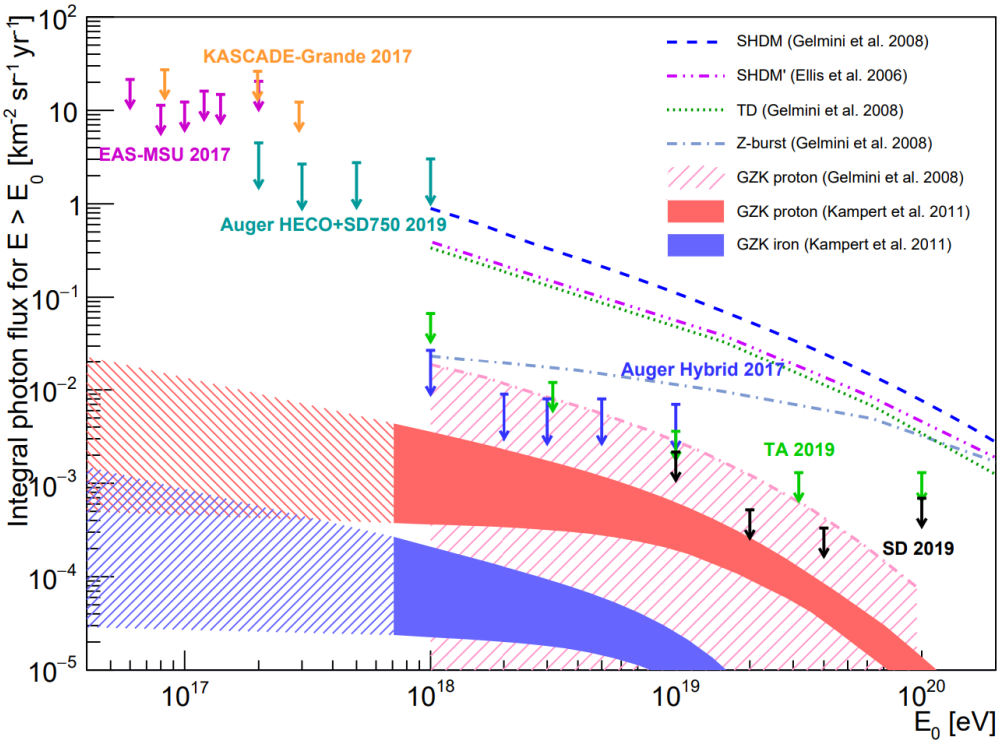


FIGURE 2.2: Photon flux limits for different experiments at 95% confidence level (90% for EAS-MSU and KASCADE-Grande), compared to model predictions. Taken from [6].

2.2 Cosmic ray-induced extensive air showers

When entering the Earth's atmosphere, UHECRs interact with a nucleus (most likely nitrogen or oxygen) and initiate a cascade of particles. The first interaction happens typically at a height of $20 - 30\,\mathrm{km}$, depending on the energy and type of the particle and, of course, with underlying statistical fluctuations because it is a probabilistic interaction. The secondary particles themselves interact with the atmosphere and produce even more particles that successively feed the cascade. With a few meters in depth, the air shower propagates through the atmosphere as a thin, slightly curved plane. The lateral extension, on the other hand, can exceed a few kilometers, again depending on the energy and type of the particle. A sketch can be seen on the right of figure 2.3.

The air shower has three main components: the hadronic, the muonic and the electromagnetic component (see figure 2.3 on the left). In principle neutrinos, mostly produced by the production and decay of muons, are a fourth component, but because they are almost unmeasurable, they are not very important, apart from the fact that the energy reconstruction of the EAS has to be corrected for the invisible neutrino energy. The three main components are described below.

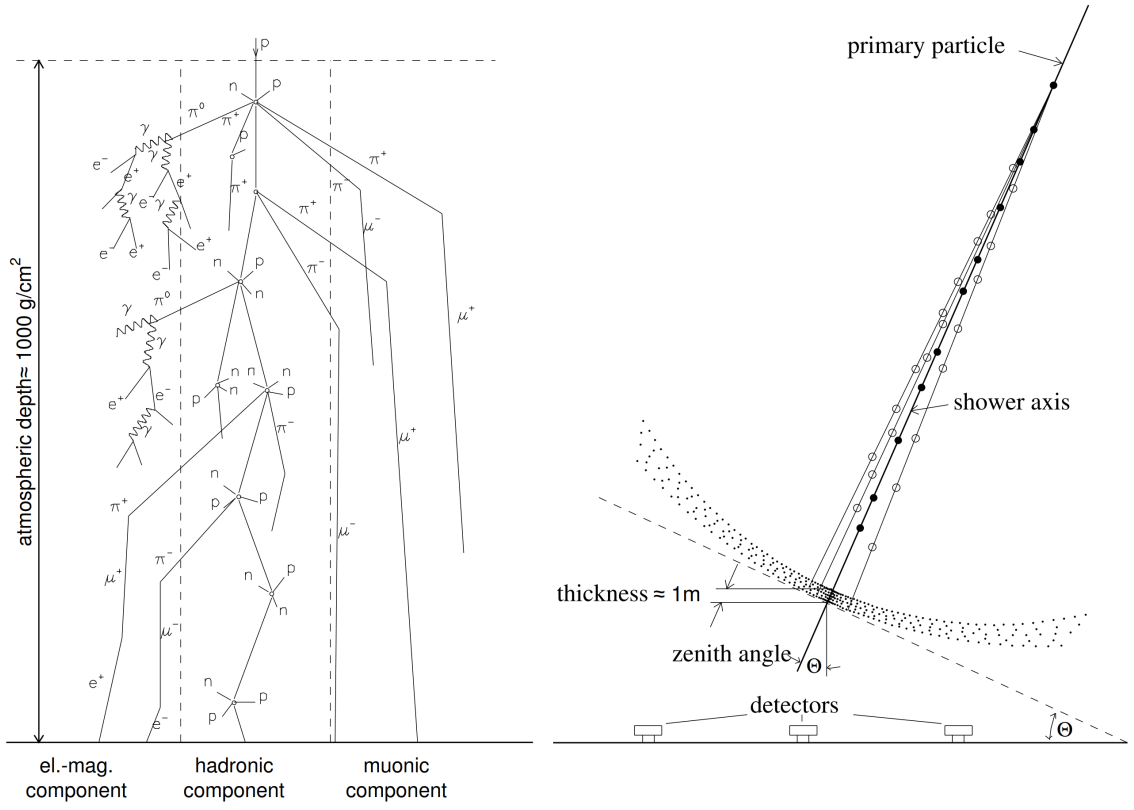


FIGURE 2.3: Left: Scheme of the different components of an extensive air shower. In real air showers different components are not spatially separated but mixed. Right: Sketch of an air shower not to scale with its thin shower front arriving with zenith angle Θ. Taken from [15].

Hadronic component. Usually initiated by a hadronic particle, the hadronic component consists mostly of baryons such as protons and neutrons, and mesons like pions and kaons. Initialization due to photons by photonuclear interaction is also possible. The cascade develops due to further scattering and interactions in the air, along with the decay of short-lived hadrons whose decay products can also contribute to the other components.

Muonic component. If mesons, especially kaons and pions, of the hadronic component decay, they often produce muons:

$$\pi^+ \to \mu^+ + \nu$$
$$\pi^- \to \mu^- + \bar{\nu}$$
$$K^+ \to \mu^+ + \nu$$
$$K^- \to \mu^- + \bar{\nu}$$

Kaons also decay into pions, which in turn decay into muons. As muons interact very little with the atmosphere, most of them reach the Earth's surface on an almost straight way. Consequently, muons normally reach the ground earlier than the electromagnetic component and still carry directional information. Most muons have relativistic energies and therefore do not decay before reaching the surface, whereas low-energy muons can decay into electrons, thus contributing to the electromagnetic component.

Electromagnetic component. When π^0 mesons decay almost immediately ($\tau = (8.52 \pm 0.18) \cdot 10^{-17}\,\mathrm{s}$ [16]) into two photons, they initiate an electromagnetic cascade.

$$\pi^0 \to 2\gamma$$

The two main processes within the electromagnetic component are the e^+e^- pair production

$$\gamma \to e^+ + e^-$$

and bremsstrahlung of electrons and positrons by interaction with the nuclei in the air. These two effects recur as long as the photons exceed the critical energy $E_{crit} \approx 37\,\mathrm{MeV}$. Below this value, they do not have enough energy for further reactions and the cascade begins to fade. These two production processes cause the electromagnetic component to increase very fast so that the number of photons, electrons and positrons by far exceeds the particle count of the other components, making the electromagnetic component the dominant part of the particle density and energy in EASs. Charged particles can excite nitrogen molecules in the atmosphere, which leads to the production of fluorescence light. Thus the electromagnetic component is also dominant in the production of this fluorescence light. The electromagnetic component can be described well by the Heitler model [17].

The slant depth X is a quantity describing the longitudinal development of EASs. It is measured in $[X] = \text{g cm}^{-2}$ and is defined by the integral over the transversed air density

$$X(x) = \int_{\infty}^{x} \rho(x')\mathrm{d}x' \tag{2.2}$$

with the height x and the air density ρ. The number of particles in the EAS can be parametrized by the Gaisser-Hillas function [18]

$$N(X) = N_{\max} \left(\frac{X - X_0}{X_{\max} - X_0} \right)^{\frac{X_{\max} - X_0}{\lambda}} \exp\left(\frac{X_{\max} - X}{\lambda} \right) \tag{2.3}$$

with the maximum number of secondary particles $N_{\max}$, the slant depth of the first interaction X_0 and the effective mean free path between interactions λ. At some point in the shower development, the particles have not enough energy for further reactions which is why the slant depth of the shower maximum $X_{\max}$ is reached. It is dependent on the energy and the type of the particle and, therefore, an important observable for the distinction between photon and proton-induced air showers. Usually, photons have a higher $X_{\max}$ than protons for the same energy and the $X_{\max}$ of heavier nuclei is even lower. Since proton and photon-induced air showers are the most alike, only the differences and distinction methods between these two types are discussed in this analysis.

2.3 Attributes of photon-induced air showers

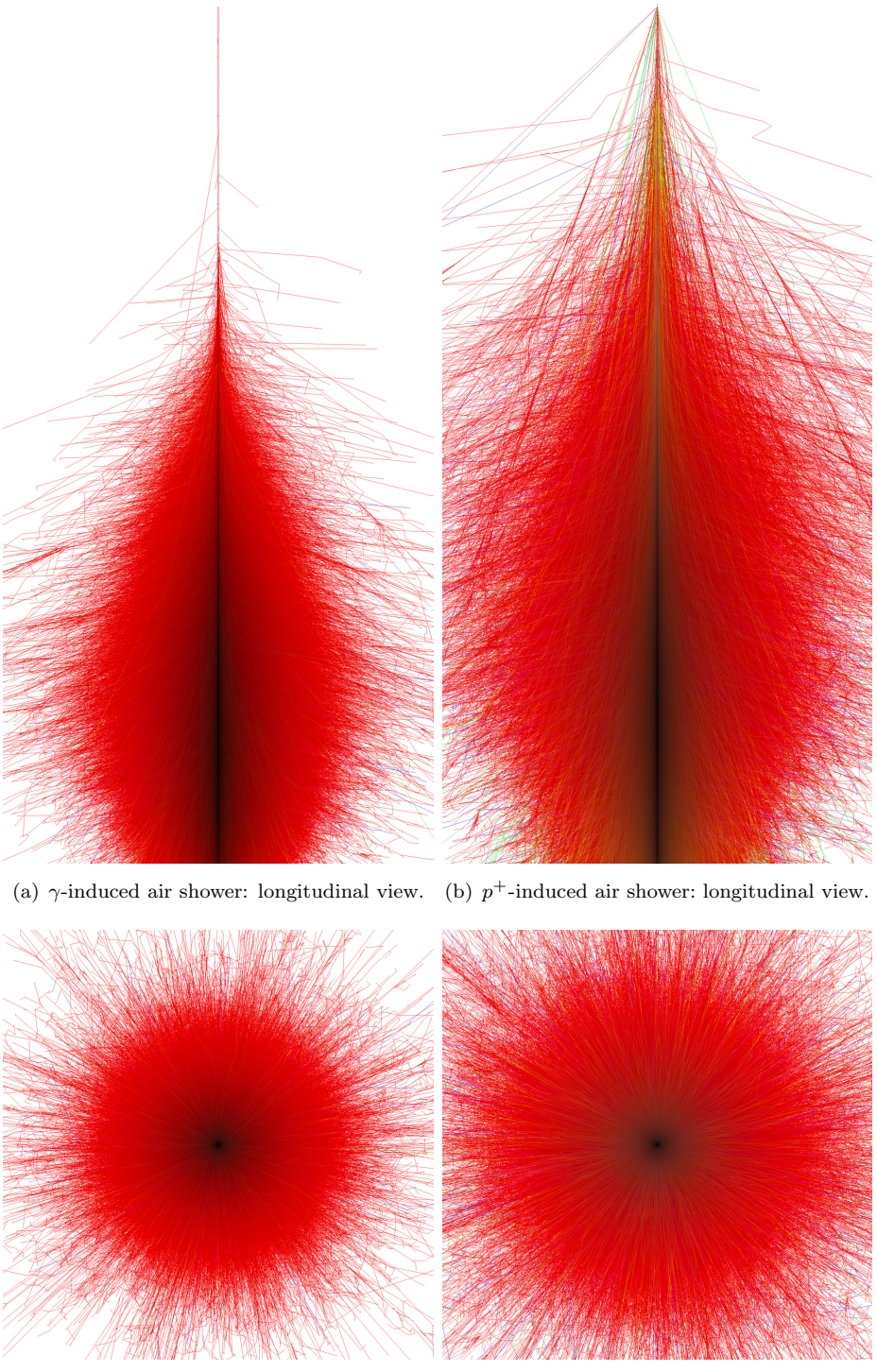
In order to find UHEPs it is crucial to know the differences between photon and proton-induced EASs. A very important difference is the slant depth of the shower maximum $X_{\max}$ which is, for the same energy, on an average higher for photon-induced showers, although an event-by-event distinction is not possible, because statistical fluctuations cause an overlap of the two distributions, though the distinction becomes clearer for higher energies. Moreover, the photon penetrates much deeper into the atmosphere before the first interaction happens. Another difference is that photon-induced showers can only produce hadrons by photonuclear interactions which causes a strongly reduced hadronic component compared to hadron-induced air showers. Since muons are mainly produced in the hadronic component, they are suppressed in photon-induced showers as well. This also means that those showers, compared to proton-induced ones, extend over a smaller area and thus are more concentrated, because hadron interactions provide a high transverse momentum which makes the muon component widespread. As a consequence, the ratio of the electromagnetic component to the muonic component is higher in photon-induced showers. At the same primary energy they therefore excite more air molecules and emit more fluorescence light. Another difference is that, due to lower muon production, less

neutrinos are produced as well. This means that photon-induced showers also have less invisible energy. For proton-induced showers the fraction of invisible energy is $\sim 10\%$, but less than 1% for photons [19]. A photon and a proton-induced air shower is shown in figure 2.4 for a better visual understanding of the differences.

Two important effects change the behavior of photon-induced EASs, which increases the difference to hadronic induced ones. These effects are the Landau-Pomeranchuk-Migdal (LPM) effect and the preshower effect which are described below.

Landau-Pomeranchuk-Migdal effect. The LPM effect [20, 21] describes the suppression of the cross section of pair production and bremsstrahlung processes for very high energies. e^+e^- pair production requires a transfer of momentum from a photon to the nucleus via a virtual photon. For higher photon energies, the transversed momentum becomes smaller and the wavelength of the virtual photon larger. If the photon has enough energy, the wavelength of the virtual photon becomes longer than the mean free path in the medium. Pair productions at multiple nuclei then destructively interfere and as a result the cross section of the process is reduced. The higher the photon energy and the material density, the higher is the suppression of the pair production cross section. A similar effect happens for bremsstrahlung of $e^{\pm}$. The density dependence of the LPM effect means an increasing suppression at lower altitudes. So the LPM effect delays the first interactions of UHEPs in the atmosphere, further delays the whole shower development and also causes higher fluctuations in the shower development resulting in a higher $X_{\max}$. These effects become relevant for UHEP energies above 1 EeV [22].

Preshower effect. UHEPs can do e^+e^- pair production before even entering the atmosphere, by interacting with the Earth's magnetic field. This is the so-called preshower effect [23]. The $e^{\pm}$ then emit photons through bremsstrahlung in the magnetic field. As a result, many electromagnetic particles that are not energetic enough for the LPM effect, enter the atmosphere. This means that the preshower effect and the LPM effect usually compete. The preshower effect is dependent on the strength of the transverse magnetic field and, therefore, on the location on Earth and the arrival direction of the UHEP. It increases for higher primary energies, but it only becomes significant at energies above $10^{19.6}$ eV. Since our simulated and analyzed energy range does not exceed $10^{19.5}$ eV, the preshower effect will not be discussed any further.



(a) γ-induced air shower: longitudinal view. (b) p^+-induced air shower: longitudinal view.

(c) γ-induced air shower: footprint view. (d) p^+-induced air shower: footprint view.

FIGURE 2.4: Comparison of a vertical photon (left) and proton-induced air shower (right). Both were simulated with CORSIKA with 10^{15} eV as the primary energy. The red lines are the electromagnetic component ($e^{\pm}$, γ), green lines are muons and blue lines are hadrons. Colors are mixed for overlapping components such that, for example, yellow means electromagnetic and muonic. It can be seen that the proton shower is broader which results in a larger footprint (bottom pictures), has more muons (more green and yellow) and has the first interaction much earlier (top pictures). Taken from [24].

2.4 Detection principles

Cosmic rays can be measured directly with detectors in space, but above energies of TeV, the particle flux decreases to such low values that the detector size has to be much bigger for reasonable statistics and this is infeasible. Instead, UHECR are measured indirectly by measuring the induced EAS.

There are many different ways to measure an air shower. The most important methods can be seen in figure 2.5. The particles reaching the ground can be measured by an array of particle detectors. The air shower also emits fluorescence and Cherenkov light, which can be detected by telescopes. There are also photons in the radio range, which are emitted by the air shower due to the coupling of electrons and positrons to the Earth's magnetic field and the charge separation inside the shower. These photons can be detected by an array of radio antennas. Large neutrino detectors are suitable to measure the otherwise almost unmeasurable neutrino component of the shower.

All these detectors measure different particle types and stages in the shower development and they therefore offer different advantages and disadvantages. For a precise measurement of the air shower and its properties, it is important to apply multiple methods.

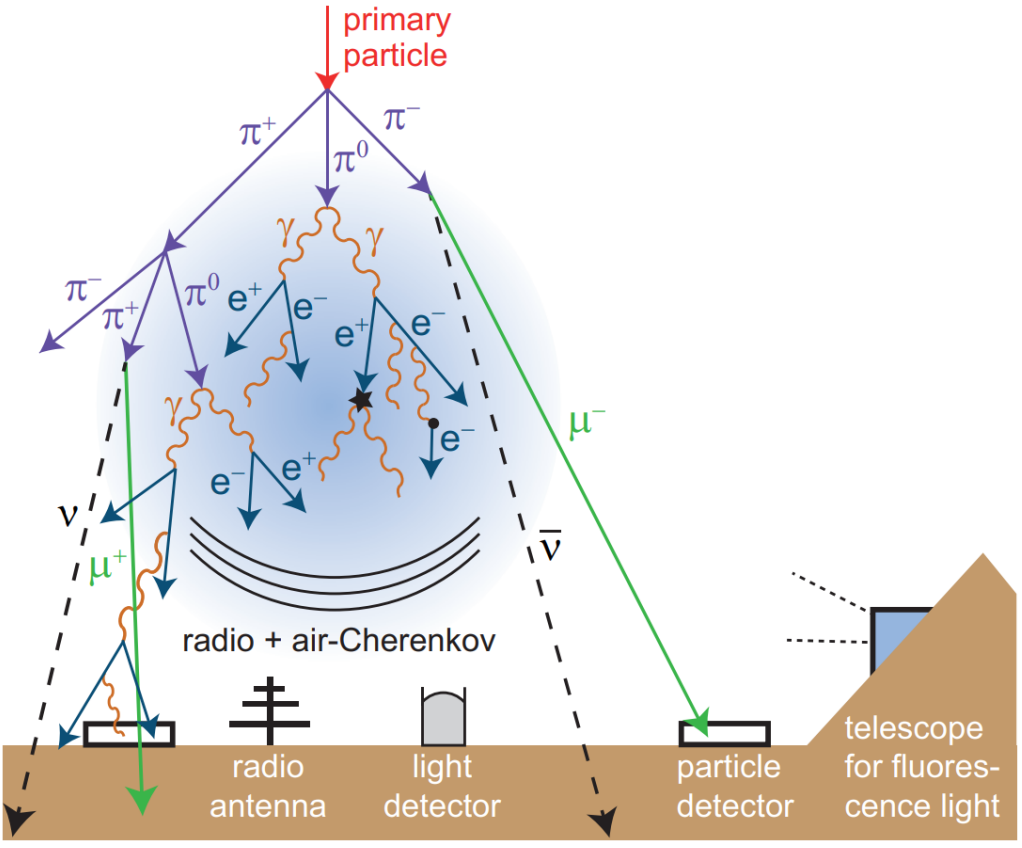


FIGURE 2.5: Sketch of different detection techniques of an air shower. Taken from [25].

Chapter 3

The Pierre Auger Observatory

The Pierre Auger Observatory is the largest cosmic ray observatory in the world; it is located 1400 m above sea level near Malargüe, Argentina. The location and structure of the observatory can be seen in figure 3.1. It measures UHECR-induced EAS with primary energies above 10^{17} eV by using a hybrid measurement method. There are 1660 water-Cherenkov detectors spread over an area of $\sim 3000\,\mathrm{km}^2$, supplemented by five fluorescence detector buildings with six telescopes, each located on the edges of the SD array overlooking it. While the FD is measuring the shower development in the air, the SD is measuring the shower footprint on the ground. The combination of these two methods leads to an accurate reconstruction of the EAS and thus of the primary particle. Although the Pierre Auger Observatory includes some other detection techniques such as, for example, radio antennas, only the details of the SD and FD will be discussed in the following, because only these are used for this analysis.

3.1 The Surface Detector

The SD consists of 1660 identical surface water-Cherenkov detector (WCD) stations arranged on a triangular grid with 1.5 km spacing spread over an area of $\sim 3000\,\mathrm{km}^2$ [26]. A picture and a sketch of one of these detectors are shown in figure 3.2. Each detector consists of a lightproof tank filled with 12 tons of highly purified water. When a charged particle enters the tank at a speed faster than the speed of light in water, it produces Cherenkov light. This light can be reflected by the inner walls and measured by three photomultiplier tubes (PMTs) mounted inside the tank on the top looking downwards. The intensity of the resulting signal can be measured with a good time resolution. It outputs for each PMT a signal trace with 768 time steps where each step is equal to 25 ns. The signal is usually calibrated to vertical equivalent muons (VEM) where one VEM is

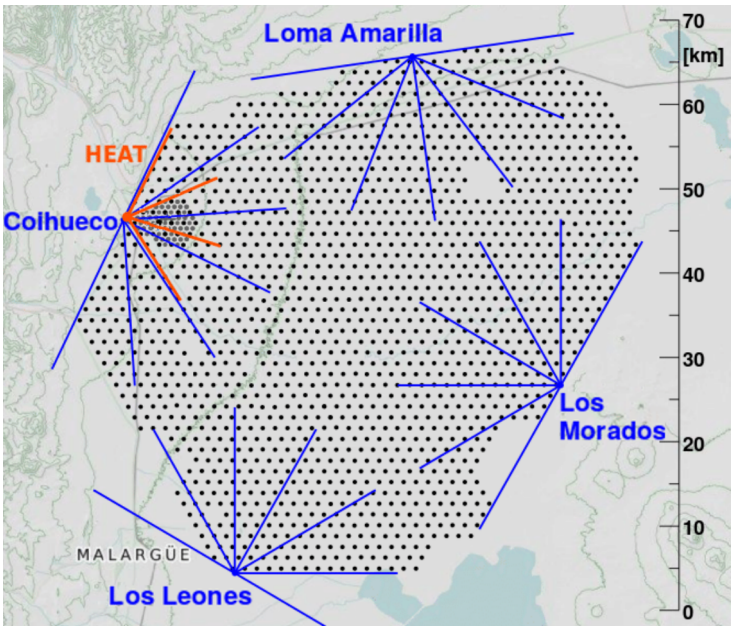
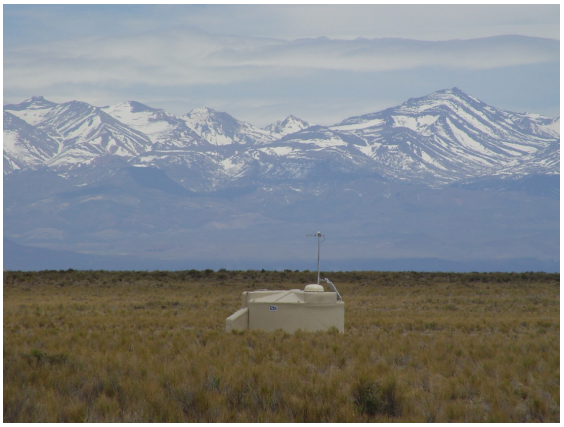
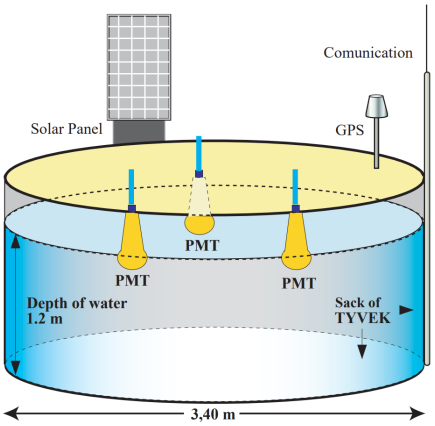


FIGURE 3.1: Location and structure of the Pierre Auger Observatory on the map. Each black dot represents one of the surface detector stations. The blue lines mark the field of view of the six telescopes for each of the four fluorescence detector enclosures. Taken from [27].

the signal produced by a vertical muon crossing the station. Additionally, the arrival time is determined with a trigger as soon as a certain signal strength is exceeded. When the charged particles in the EAS reach the ground, they produce Cherenkov light in the tanks. The more energetic the primary particle is, the higher the number of secondary particles in the EAS is. Therefore, more stations are hit and the signals in every station are higher. Using the arrival time and the signal of multiple stations, it is possible to reconstruct, for example, the arrival direction and the primary energy of the cosmic ray. Since the SD stations are lightproof and operational at day and night, they have a duty cycle close to 100%.



(a) Taken from [3].

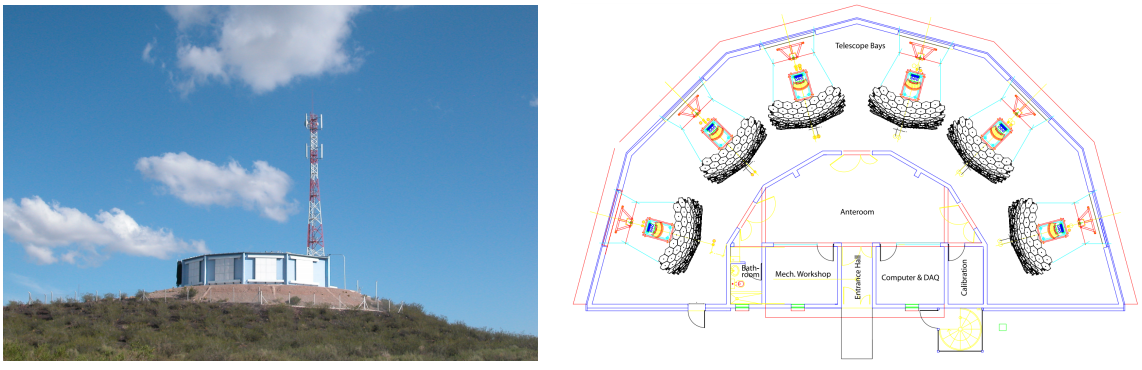


(b) Taken from [28].

FIGURE 3.2: Picture (left) and sketch (right) of one single water-Cherenkov detector.

3.2 The Fluorescence Detector

The FD consists of four buildings (Coihueco, Loma Amarilla, Los Morados and Los Leones) located on the edges of the SD [29]. Each building has 6 telescopes, each with a field of view of $30° \times 30°$ starting at a minimum elevation of 1.5° above the horizon. Thus each building has a total field of view of 180°, which allows the whole SD array to be overviewed. A picture and a sketch of a FD building are shown in figure 3.3. When the electromagnetic component of EASs interacts with the atmosphere, the atoms in the air are excited. These excited atoms can then spontaneously emit light during their de-excitation, so-called fluorescence light. Because the Earth's atmosphere mostly consists of nitrogen, most of the fluorescence light is in the UV range. This isotropic light can then be detected by the FD telescopes which have apertures with UV-passing filters to reduce background light. The transmitted light is then focused on the camera by a $10\,\mathrm{m}^2$ mirror. The camera contains 22 rows and 20 columns of pixels. Each of these 440 pixels is a PMT with a field of view of 1.5°. Like the SD, each pixel measures a signal trace proportional to the amount of light and the arrival time of the signal. Since the fluorescence light intensity is very weak, the FD only works properly in clear and moonless nights, which results in a reduced duty cycle of only $\sim 15\%$. Nevertheless, the FD-recorded signals allow the longitudinal shape of the shower development to be observed which makes the reconstruction of X_{max} possible. Furthermore, since the fluorescence light is a measurement of the energy deposited on the atmosphere, it is used for energy calibration purposes because it is proportional to the primary cosmic ray energy. Thus the measurements of combined SD and FD events, called hybrid events, can in principle compensate the lower exposure due to the reduced duty cycle by furnishing additional information, thus improving reconstruction.



(a) Picture of the fluorescence detector enclosure Los Leones with closed shutters. Taken from [3].

(b) Schematic top-down view of the FD building with six fluorescence telescopes. Taken from [29].

FIGURE 3.3: Picture (left) and sketch (right) of one of the four FD buildings.

3.3 The Infill array

In order to be sensitive to energies down to 10^{17} eV, the so-called infill array was constructed. It is an area of 24 km^2 located in the north east of the SD close to Coihueco, where additional SD stations were built to create an array with half the distances (750 m) between each of the stations. A map of the infill array can be seen in figure 3.4. Closely located to Coihueco are the High Elevation Auger Telescopes (HEAT). These are three additional FD telescopes, the only difference being that they are tilted towards higher angles in order to cover the field of view above Coihueco. A picture and sketch of HEAT can be seen in figure 3.5. The extension of Coihueco by HEAT is called HECO. When the larger field of view of HECO and the denser SD grid close to it are used, it is possible to detect EASs of less energetic primary particles, because they produce less signals, trigger less stations and penetrate less deeply into the atmosphere. Since the cosmic ray flux is much higher at lower energies, the smaller area of the infill array compared to the main array is still sufficient for a high exposure and good statistics.

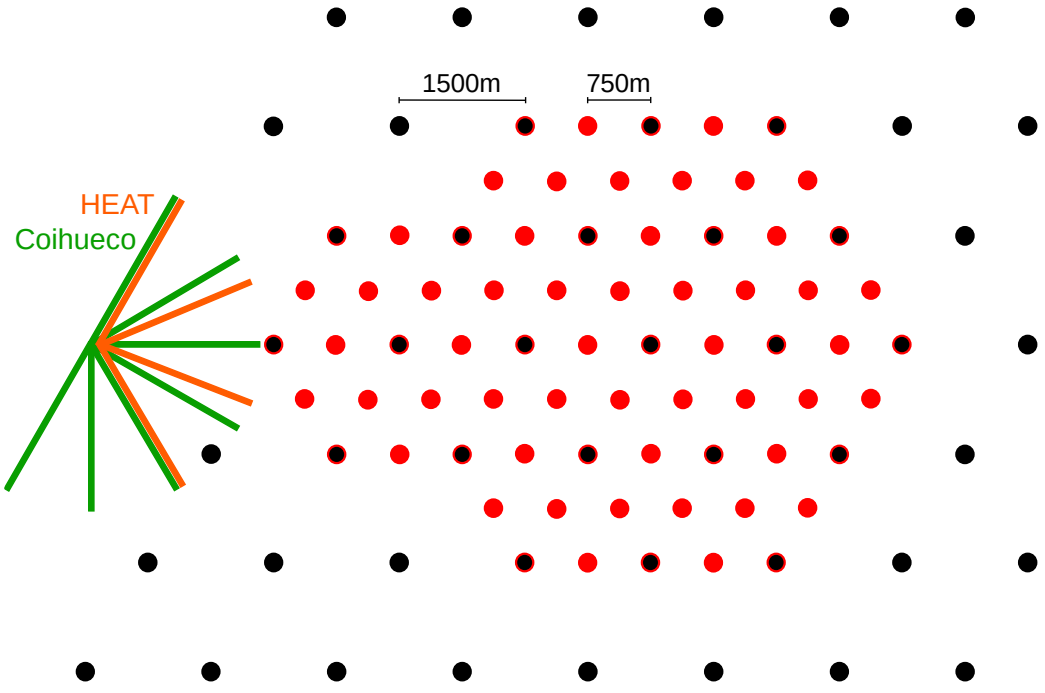
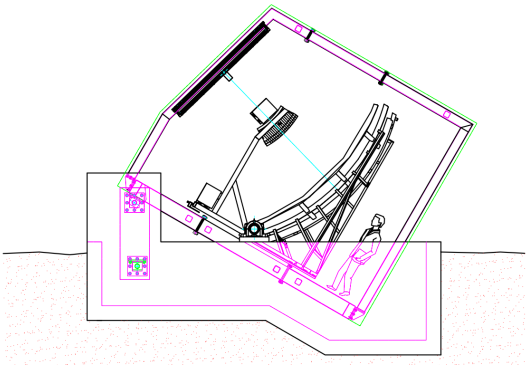


FIGURE 3.4: Map of the infill array. The red dots are the 49 SD stations that were added with a spacing of 750 m. The black dots with a red ring are stations which belong to both the infill and the large main array. Thus the infill consists of 71 stations in total. The black dots belong to the main array only. The locations of Coihueco (green) and HEAT (orange) with their fields of view have also been marked. Although all telescopes have the same field of view, HEAT sees a larger range of azimuth angles because of the higher inclination. The location of the infill array within the Pierre Auger Observatory can be seen in figure 3.1.

(a) Photo of the three HEAT telescopes in tilted mode.



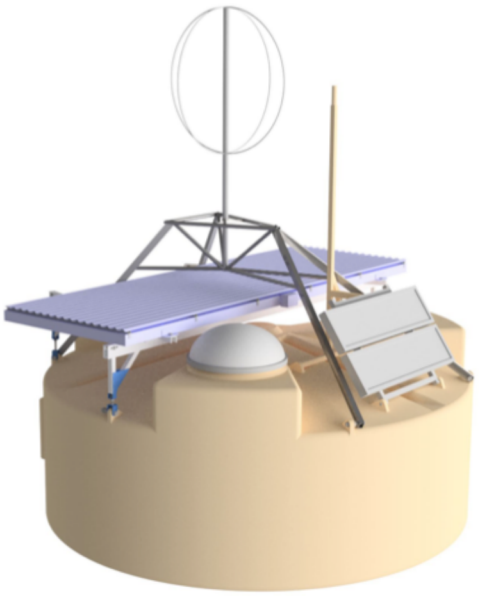
(b) Sketch of one HEAT building in tilted mode.

FIGURE 3.5: Picture (left) and sketch (right) of HEAT. Taken from [3].

3.4 AugerPrime Upgrade

A big upgrade to the Pierre Auger Observatory, called AugerPrime, is currently under construction [30]. A key feature of this upgrade is the deployment of a Surface Scintillator Detector (SSD) on top of each WCD. A picture and a sketch of an upgraded SD station are shown in figure 3.6.

(a) One station of the AugerPrime Engineering Array with a SSD on top of the WCD.



(b) Sketch of the final version of a SD station including the WCD, SSD and radio antenna.

FIGURE 3.6: Picture (left) and sketch (right) of one upgraded surface detector station. Taken from [31].

The complementary measurement with this plastic scintillator furnishes different responses to the muonic and electromagnetic components of the air shower compared to the WCD. The SSD is more sensitive to electromagnetic particles whereas the WCD detects more muons. This means that, with both measurements, a separation between these two components is possible. Since the fraction of these components is different for photon and hadron-induced EASs, a much better separation power is expected with AugerPrime. To estimate this possible improvement, a method using data which are comparable to those obtained by AugerPrime will also be discussed in this analysis (see section 7.6). Other features of the upgrade are an Underground Muon Detector (UMD) in the infill array, upgraded new electronics for the SD and a new operation mode of the FD to increase its duty cycle, which will increase the statistics of hybrid events.

3.5 Standard event reconstruction

When an extensive air shower hits the Pierre Auger Observatory, a coincident signal detection in multiple SD stations or FD pixels triggers the readout of the shower data. The geometry of such an event can be seen in figure 3.7. For each SD station or FD pixel, respectively, the arrival time, the signal-over-time and calibration information are stored. Using complex reconstruction algorithms, the air shower can be reconstructed and quantities like the primary energy, the arrival direction and X_{max} can be determined. There are methods which use only SD or FD data. These methods are mainly used when only one of the two detectors triggered, e.g. during daytime when no FD data is available. However, for a better reconstruction with lower uncertainties, it is meaningful to do a hybrid reconstruction taking SD and FD data into account. With the amount of signal and its arrival time in the SD, the lateral distribution of the air shower and from this the shower core position on the ground can be reconstructed. This allows the arrival direction of the cosmic ray to be determined, so that a reconstruction of the zenith and azimuth angles of the shower axis is possible. Since the FD measures the longitudinal shower development in the atmosphere, the profile of the deposited energy and with it X_{max} can be reconstructed. With the amount of signal in the SD or FD, the primary energy can be calculated. Corrections must be applied, because - although the signal is proportional to the primary energy - particles such as neutrinos do not contribute any signal. Since these reconstructions and corrections depend, among other things, on the type of the cosmic ray, the weather and the fluorescence yield, systematic uncertainties may occur. The statistical uncertainty of the hybrid energy reconstruction is in the order of $\sim 10\%$, for X_{max} it is $\sim 20\,\mathrm{g\ cm^{-2}}$ and the arrival direction can be reconstructed within 1° [32].

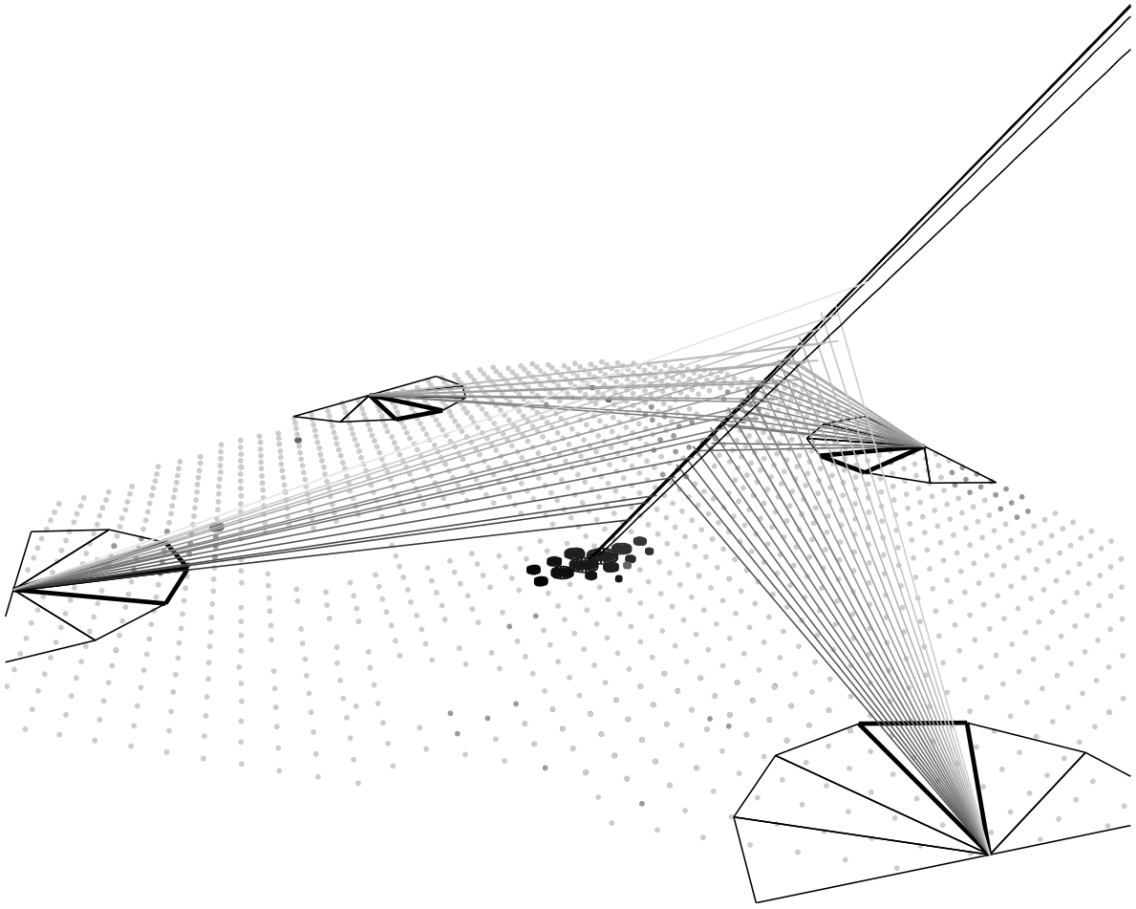


FIGURE 3.7: Example of a reconstructed event observed by one telescope at each telescope site in coincidence with the surface detector. Taken from [3].

Chapter 4

Simulation

Before applying machine learning algorithms to data, it is necessary to train and test them in simulated detector measurements, as the true attributes of the air shower and the primary particle, especially the type of the particle, are known. In order to simulate the shower development in the Earth's atmosphere, the COsmic Ray SImulations for KAscade (CORSIKA) [33] was used, which was developed for the KASCADE experiment [34] in Karlsruhe. These simulated air showers are then given to the Offline Software Framework [35], which is the standard simulation and reconstruction framework of the Pierre Auger Observatory. It simulates, for each CORSIKA air shower, all detector measurements and then reconstructs all shower parameters.

4.1 CORSIKA

CORSIKA is a useful tool for the simulation of air showers. It simulates the primary particle propagating through the Earth's atmosphere and its interactions with the air molecules. Furthermore, the propagations and interactions of all secondary particles including photons, electrons and positrons, muons and hadrons are simulated individually. This approach makes CORSIKA simulations both accurate and computationally expensive, nevertheless there are some mechanisms to reduce calculations. For example, neutrinos will not be simulated after creation because, most probably, they would not interact anyway. The same is valid for all particles below a certain energy threshold, as they would not make any significant contribution to the shower anymore. The simulations of hadronic interactions are problematic, because they are based on research at the LHC. This means that those hadronic interaction models must be interpolated over several orders of magnitude in energy. Different hadronic interaction models are available, namely EPOS LHC, QGSJetII-04 and Sibyll 2.3c, using different approaches and interpolations. EPOS LHC is

used here, because it is, currently, the best fitting model compared to data [36]. In order to be able to assess possible artefacts of the simulation and to make sure that the developed methods are robust against them, smaller samples using QGSJetII-04 and Sibyll 2.3c were simulated. At low energies, Fluka2011.2x [37] was used as the hadronic interaction model. The simulation sample was taken from the Napoli+Praha library [38].

In total, 36 000 CORSIKA air showers were simulated using EPOS LHC with a primary energy ranging from $10^{17.0}$ eV up to $10^{19.5}$ eV equally distributed in logarithmic scale (E^{-1} spectrum). The arrival direction is isotropically distributed on a spherical surface with a zenith angle of up to $\theta = 65°$. Half the simulated primary particles were photons and the other half were protons, because, as already mentioned, we focus on the distinction between photons and protons since they are most alike. For QGSJetII-04 and Sibyll 2.3c, 5000 proton-induced air showers were simulated, each with the same energy and zenith distribution. Photons were not simulated in the QGSJetII-04 and Sibyll 2.3c test sample.

4.2 Offline Software Framework

The Offline Software Framework [35] is the standard tool for detector simulation and air shower reconstruction of the Pierre Auger Observatory. After the development of the air shower in the Earth's atmosphere has been simulated using CORSIKA, the output is given to Offline which then simulates all measurements of all detectors of the Pierre Auger Observatory. In this step, the Cherenkov light induced by charged particles of the air shower in the water of the SD tank is simulated, as well as its propagation through the water, reflections at the walls, its measurement by the three PMTs up to the digitalization of the signal in the electronics. Furthermore, it is simulated how the fluorescence light reaches the FD telescope, is reflected in the mirrors, measured in the PMTs and again how the signal trace is digitalized.
Offline also provides reconstruction tools which can then be applied to the fully simulated event. In this step, for example, the primary energy, the arrival direction and X_{max} are reconstructed. In the end, a sample of events covering all detector measurements as well as reconstructed high-level observables is obtained.

The shower core is selected to be randomly distributed inside the infill array, because, for the chosen energy range, the infill array is much more sensitive. In order to increase the statistics, each CORSIKA shower is used five times in Offline, with different random core positions to make sure that the events are not too similar. So in the end, the number of events in the sample is five times that of the CORSIKA showers. Simulation of even more CORSIKA showers would be too time-consuming and would require much more space to store.

4.3 Data set selection

Before applying the methods used to the simulated data sample, several cuts have to be made to remove events which are unwanted for different reasons. Some cuts only reject certain SD tanks or FD pixels, others will remove the whole event. Of course, the same cuts would have to be applied when switching to data. In table 4.1 all cuts that are made are listed with a short description.

Cut	Description
Golden Hybrid	The event must be fully reconstructable with SD and FD to guarantee that all high level observables are reconstructed properly.
SD Rejection Status = 0	An SD signal is only accepted if the so-called rejection status is equal to zero which means that there is no rejection like, for example, no trigger or no calibration data. Otherwise, the SD tank will be ignored.
SD ID	SD tanks which do not belong to the infill or main array are ignored, e.g. if they are off grid or test stations.
Length of VEM trace > 100 entries	Some VEM traces of the SD are shorter than 100 entries (normal ones have 768). SD tanks with this issue will be ignored, because at least 100 entries in the signal trace are needed for the deep learning methods.
Has SD VEM trace	If no SD tank survives the upper cuts, the whole event will be removed, because then there is no useful SD signal.
Inner radius cut: Distance of SD tank to shower axis > 100 m	Due to the huge amount of particles and their computational costs, CORSIKA reduces the accuracy of the simulation inside a certain radius around the shower axis. In this case, this radius is 100 m. Due to the fact that a proper simulation of SD signals inside this radius cannot be guaranteed, the whole event is removed if the closest station is less than 100 m away from the shower axis. Mere removal of this SD tank is not useful, because it is the most important tank, because it has most signals and is therefore crucial for energy reconstruction. The impact of this so-called inner radius cut can be seen in figure 4.1.
FD Pixel Status = 4	If the status of an FD pixel is not 4, which means that it is part of the time fit, it will be ignored.

Table 4.1: List of all cuts applied to the simulated data sample

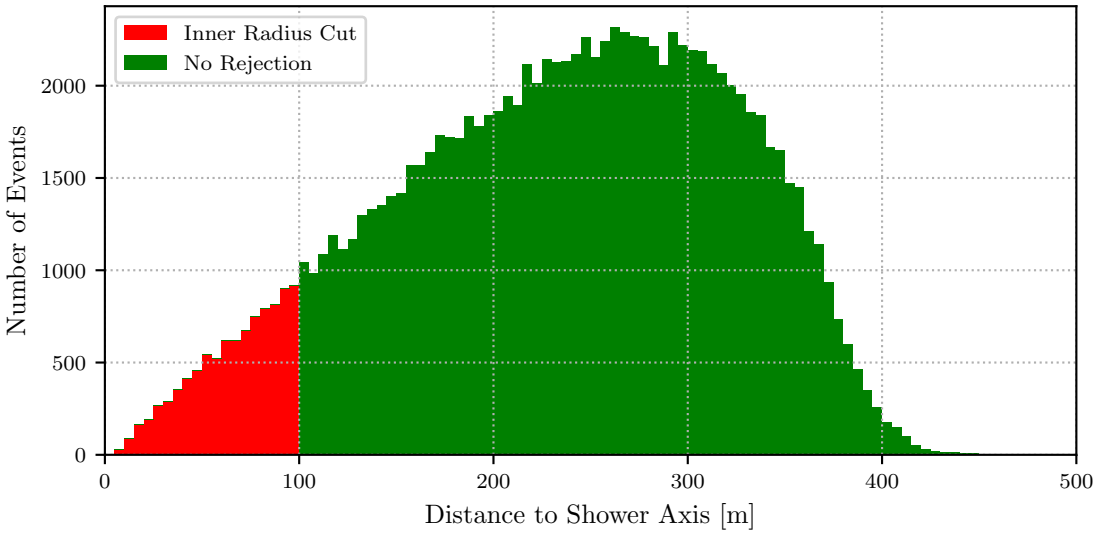


FIGURE 4.1: Histogram of the distance of the closest SD tank to the shower axis. With the inner radius cut, all events are removed where this distance is smaller than 100 m.

The numbers of events after all cuts have been made and data selection has been done are shown in table 4.2. Due to the golden hybrid cut, a large fraction of events in the first energy bin was cut out. For these low-energy events, it is much harder to reconstruct all shower properties properly, as they do not trigger enough SD stations or FD pixels.

	EPOS LHC		QGSJetII-04	Sibyll 2.3c
Energy bin	γ	p^+	p^+	p^+
$10^{17.0}\,\text{eV} - 10^{17.5}\,\text{eV}$	3279	2560	343	378
$10^{17.5}\,\text{eV} - 10^{18.0}\,\text{eV}$	12 286	12 585	1689	1775
$10^{18.0}\,\text{eV} - 10^{18.5}\,\text{eV}$	14 613	14 630	2040	2076
$10^{18.5}\,\text{eV} - 10^{19.0}\,\text{eV}$	15 263	15 193	1980	2081
$10^{19.0}\,\text{eV} - 10^{19.5}\,\text{eV}$	15 264	14 944	2087	2082
Total	60 705	59 912	8139	8392

TABLE 4.2: Number of simulated air shower events per energy bin and particle type and hadronic interaction model after data selection.

4.4 Preprocessing

In all methods that will be discussed later, the output will be the type of the particle, precisely whether it is a proton or a photon. The input, on the other hand, will vary from approach to approach. The inputs that will be used are the following: for every SD tank as well as for every FD pixel the signal traces over time, the arrival time of the signal and the total signal. Furthermore, the zenith angle of the shower axis, the calorimetric energy and the depth of the shower maximum $X_{\max}$, all reconstructed by FD, the number of triggered SD stations and the SD related observable S_b are used.

In order to optimize the results of machine learning and, especially, deep learning methods, preprocessing of the data samples is very important. This includes the geometry and shape of multidimensional input arrays as well as numerical ranges. Is has been shown that the results turn out to be better when the ranges of numbers in the input and output arrays are not too large. Therefore, the normalization and transformation of the data is described below.

Type of particle. To translate the type of the primary particle to different values it has been chosen that a photon (signal) has a value of 1 and a proton (background) has a value of 0. Later, all methods will output a float value between 0 and 1 that reflects the uncertainty of the method or how photon-like an event looks like.

Geometry of the SD. Using all SD tanks as an input would be computationally inefficient, especially because most tanks would not provide a signal and thus no information. A better approach is to take out only a 13×13 window of the array, for each event independently, with the station with the highest signal in its center. This size has been chosen, because only a negligible number of events have triggered stations outside this window. Additionally, the triangular grid of the SD tanks must be transformed into a squared Cartesian grid, in order to apply convolution techniques later. In figure 4.2, this transformation can be seen. Although there are also other alternatives to this transformation, it has been found that this has no significant impact during training.

Geometry of the FD. A similar procedure is followed for the hexagonally arranged FD pixels. Since all air showers are located in the Infill array, almost no event contains FD signals which do not originate from Coihueco or HEAT. Therefore, it is reasonable to ignore all other FD telescopes and just focus on HECO. The most inclined air showers trigger four out of six eyes of Coihueco and two out of three eyes of HEAT. Each eye consists of 22×20 pixels, so a good way to reduce the size of the FD data set is to take only four eyes of Coihueco (a 22×80 array) and 2 eyes of HEAT (a 22×40 array) into account. As in the case of the SD grid, these two arrays where transformed into a Cartesian tensor (see figure 4.3).

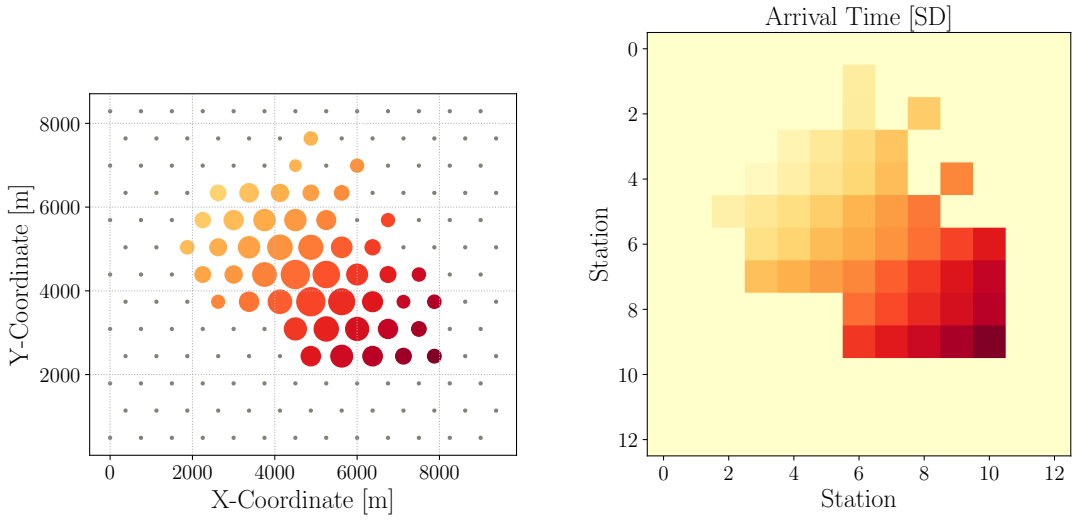


FIGURE 4.2: Example event with the triangular orientation of the SD tanks (left) and its transformation to Cartesian coordinates (right). The color indicates the arrival time; a darker color corresponds to a later signal.

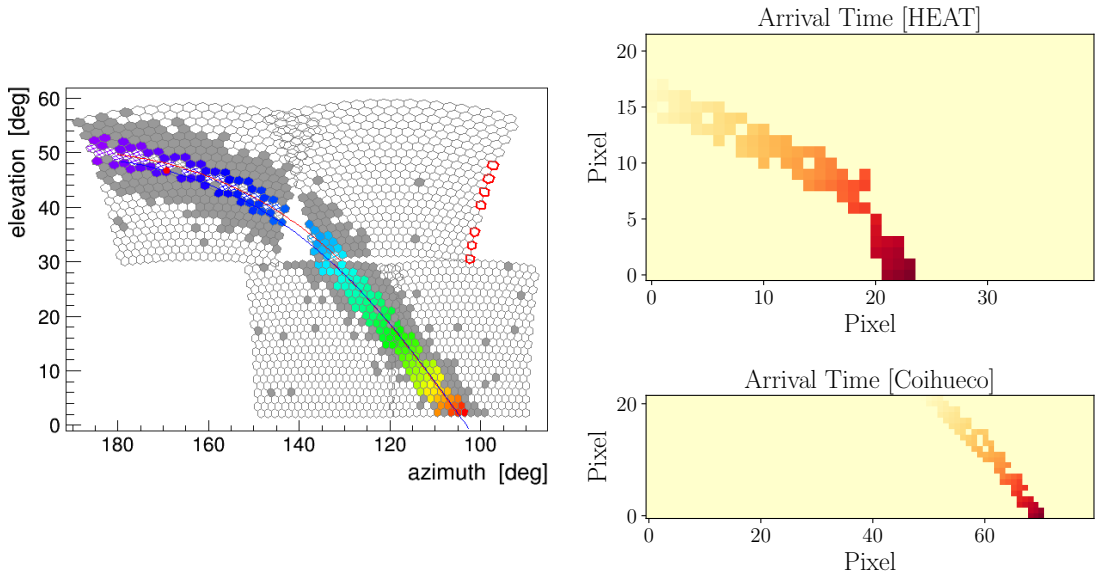


FIGURE 4.3: Example event with the hexagonal orientation of the FD pixels (left) and its transformation to Cartesian coordinates (right). Left: Representation of the event using the event browser with HEAT (above) and Coihueco (underneath). Right: After transformation, HEAT and Coihueco are stored in two separate arrays. The color indicates the arrival time; a darker color corresponds to a later signal.

Arrival time. The arrival times, both for every SD station and FD pixel, have been shifted in such a way that the first triggered SD tank and FD pixel respectively got the arbitrary value 1:

$$\tilde{T} = T - \min_{\text{event}}(T) + 1 \tag{4.1}$$

The arrival times of all tanks and pixels with no signal at all were set to zero. After that, all values were normalized by division by the global maximum arrival time of all events and all SD tanks or FD pixels:

$$\tilde{T}_{\text{X}} = \frac{T_{\text{X}}}{\max(T_{\text{X}})} \tag{4.2}$$

where $\max(T_{\text{X}})$ is $\max(T_{\text{SD}}) = 22\,301\,\text{ns}$, $\max(T_{\text{Coihueco}}) = 343.3\,\text{ns}$ and $\max(T_{\text{HEAT}}) = 130.4\,\text{ns}$.

Signal trace. The signal traces from Offline of SD contain 768 values per station, while FD has 1000 per pixel. Offline itself provides a method which determines the starting point of the signal in the trace. A time window of 100 time steps (corresponding to 2.5 ms) for SD and 50 time steps (5 ms) for FD was chosen, starting 5 time steps before the starting point calculated by Offline, to make sure that no signal is lost. The FD trace array is chosen to be of half the length of SD, because the FD signals are shorter and a larger array would be too computationally expensive. Signal traces of detectors without any received signals are set to zero, as well as signals below zero, since these values are only noise effects. The range of these signals covers several orders of magnitude which is, as already mentioned, suboptimal for DNNs. Therefore, the logarithmic transformation

$$\tilde{S} = \log_{10}(S + 1) \tag{4.3}$$

is applied, where an offset of 1 has been added to avoid negative values and keep no received signal at zero. The signal traces are then normalized with

$$\tilde{S}_{\text{X}} = \frac{S_{\text{X}}}{\log_{10}(\max(S_{\text{X}}) + 1)} \tag{4.4}$$

where, in analogy to the arrival time, $\max(S_{\text{X}})$ are the maximum values of all traces of all events ($\max(S_{\text{SD}}) = 164.5$, $\max(S_{\text{Coihueco}}) = 20\,164.8$, $\max(S_{\text{HEAT}}) = 18\,074.6$).

Total signal. We are also interested in the total signal per SD tank and per FD pixel, since it contains important information such as e.g. the primary energy. For this, each individual untransformed time trace is summed up and then the same transformation as for the signal traces

$$\tilde{S}_{\text{total,X}} = \frac{\log_{10}(S_{\text{total,X}} + 1)}{\log_{10}(\max(S_{\text{total,X}}) + 1)} \tag{4.5}$$

is applied where $\max(S_{\text{total,X}})$ is $\max(S_{\text{total,SD}}) = 3929.9$, $\max(S_{\text{total,Coihueco}}) = 121\,421.4$ and $\max(S_{\text{total,HEAT}}) = 102\,599.4$.

BDT variables. To compare the results of DNN methods with a previous BDT analysis [7], five additional variables have also been taken into account. These are: the FD reconstructed zenith angle of the shower axis θ, the calorimetric energy E_γ, the depth of the shower maximum $X_{\max}$, the number of triggered SD stations N_{stat} and the observable

$$S_b = \sum_i S_i \left(\frac{R_i}{1000\,\text{m}} \right)^b \tag{4.6}$$

where $b = 4$, S_i is the signal in VEM and R_i is the perpendicular distance to the shower axis of the i-th station. Since S_b and E_γ cover several orders of magnitude, they were logarithmized as well. All five variables were normalized with

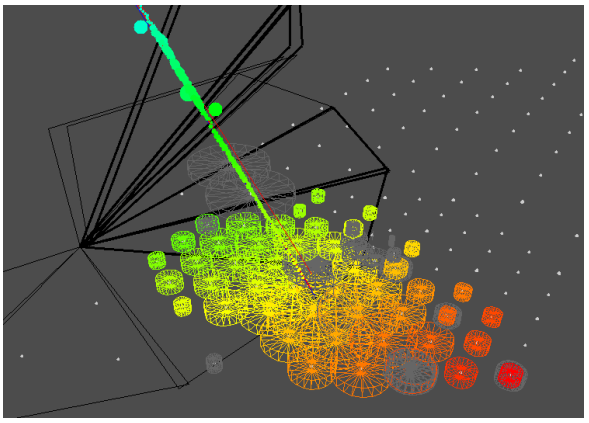
$$\tilde{X} = \frac{X - \min(X)}{\max(X) - \min(X)} \ . \tag{4.7}$$

The values of the minima and maxima are listed in table 4.3.

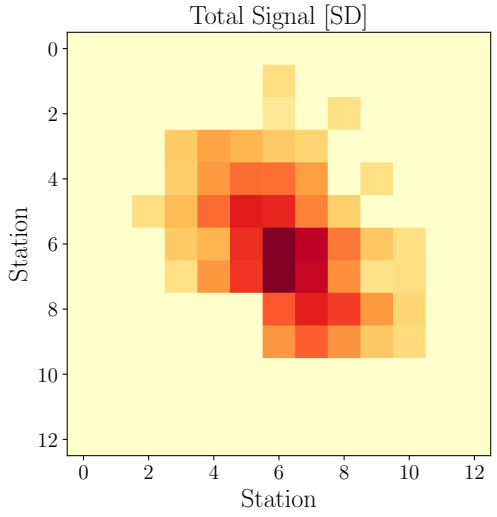
Variable	Minimum	Maximum
S_b	0.21	9909.76
$X_{\max}$	190.68	2342.24
N_{stat}	2	59
θ	0.0046	1.3549
E_γ	4.38×10^{16}	1.11×10^{20}

TABLE 4.3: Values of the minima and maxima that were used to normalize the five BDT variables.

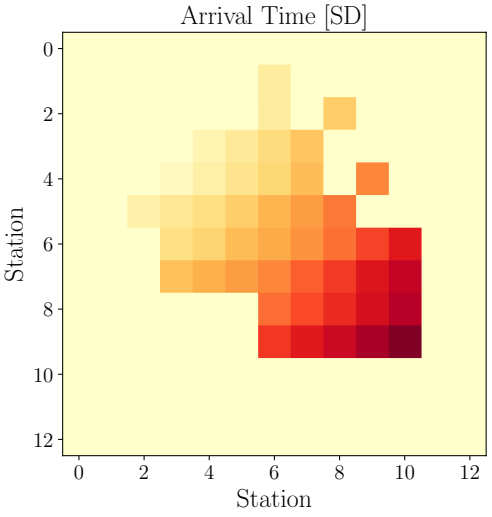
In summary, the input data consist of the arrival times of each SD tank (13×13 array), each pixel of Coihueco (22×80) and HEAT (22×40), the corresponding total signals of each tank and pixel with the same array shapes, the signal traces of each SD tank ($13 \times 13 \times 100$), each pixel of Coihueco ($22 \times 80 \times 50$) and HEAT ($22 \times 40 \times 50$) and five BDT variables (S_b, $X_{\max}$, N_{stat}, θ and E_γ). Figure 4.4 and 4.5 show one example event with all input data after the preprocessing procedure.



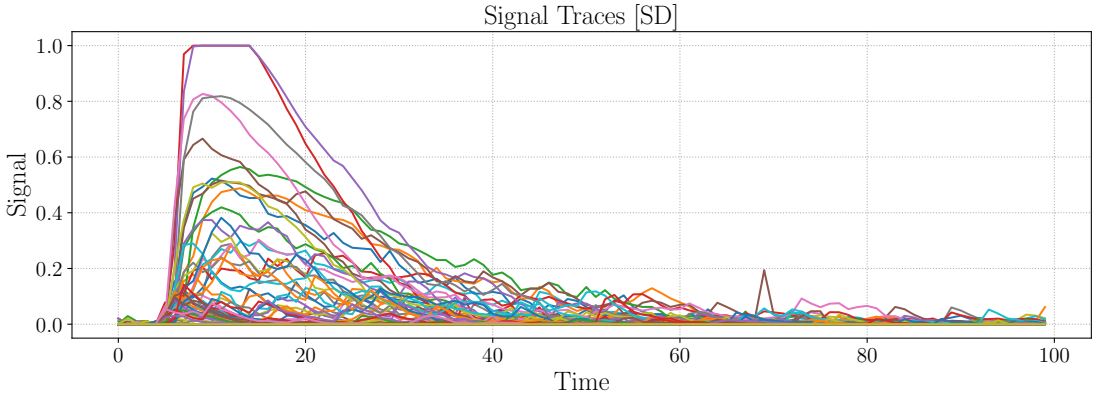
(a) Three dimensional representation of the example event. The shower axis, the triggered stations of the infill array and the field of views of the FD can be seen.



(b) Total signals of all SD stations. A darker color corresponds to a higher signal.

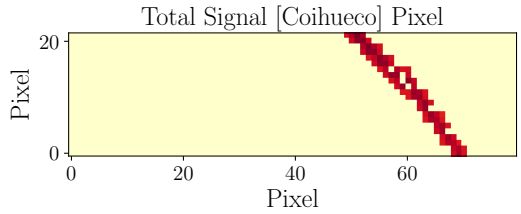


(c) Arrival times of all SD stations. A darker color corresponds to a later signal.

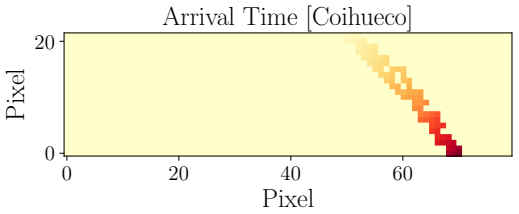


(d) Signal traces over time of all SD stations.

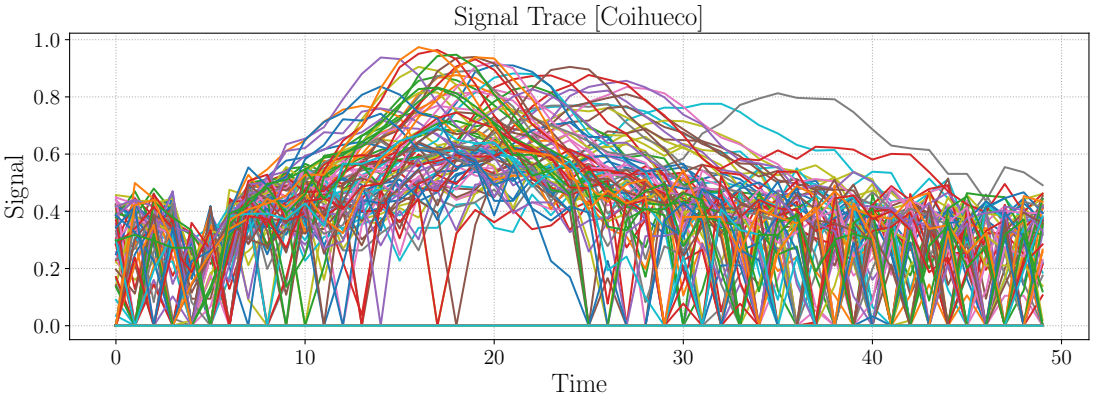
FIGURE 4.4: Example event: All SD input variables of a photon-induced air shower with a primary energy of $E = 2.88 \times 10^{19}\,\text{eV}$. The BDT input variables are $N_{\text{stat}} = 45$, $\theta = 55.96°$, $X_{\text{max}} = 973.92\,\text{g cm}^{-2}$, $S_b = 4499.25$ and $E_\gamma = 2.66 \times 10^{19}\,\text{eV}$. After preprocessing, all variables are normalized and in arbitrary units. All FD variables are shown in figure 4.5.



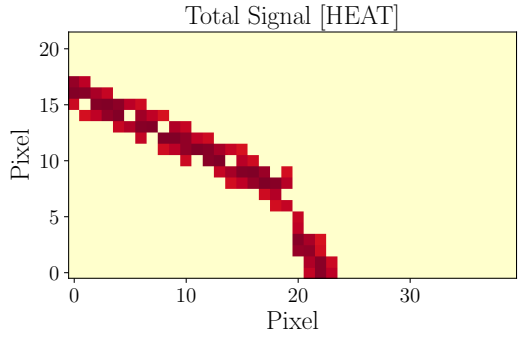
(a) Total signals of all pixels of Coihueco.



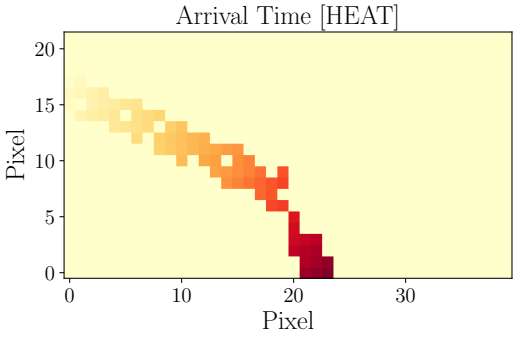
(b) Arrival times of all pixels of Coihueco.



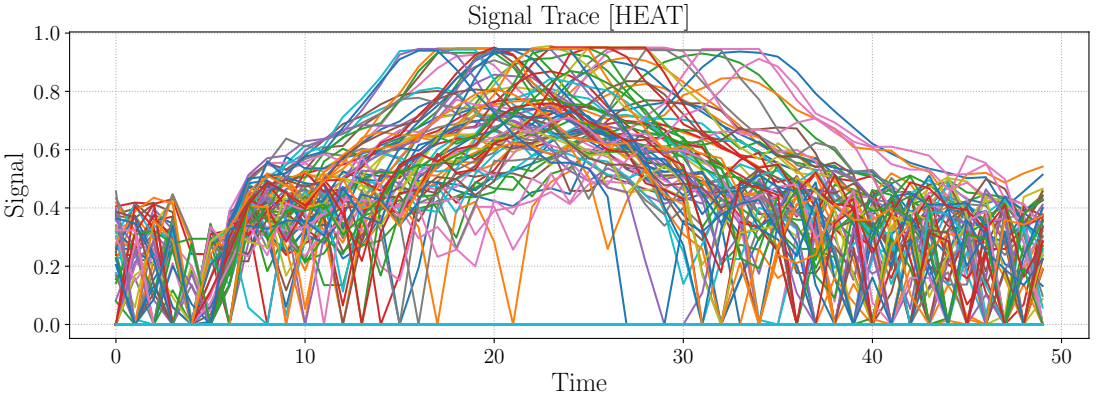
(c) Signal traces over time of all pixels of Coihueco.



(d) Total signals of all pixels of HEAT.



(e) Arrival times of all pixels of HEAT.



(f) Signal traces over time of all pixels of HEAT.

FIGURE 4.5: All FD input variables of the same event as in figure 4.4. Figure 4.3 on the left shows how this event looked like before preprocessing.

Chapter 5

Photon search with machine learning

Machine learning is a widespread field of algorithms with the capability of solving various problems without explicit instructions on how to solve them. Instead, the computer learns on its own. The key idea is learning by gaining experience. In order to train the algorithm, a dataset with all input variables is provided. In the case of the so-called supervised learning, the dataset also contains the output data, which the algorithm should predict. During the training, the algorithm predicts an output, based on the given input, and compares the result with the true output. Then, the parameters of the algorithm are adapted so that the prediction gets closer to the true value. By repeating this with a sufficient number of different data samples, the algorithm learns to make the optimal prediction for a dataset whose true output is unknown. Since this makes it possible to process and analyze huge datasets with multiple interconnected variables, where the development of a phenomenological analysis would be extremely complicated, machine learning algorithms are today used in almost all fields of data processing. Their growing popularity in the last years is based on the increasing computational power, the growing amount of stored data and more sophisticated methods.

Machine learning techniques have also become more common in physics applications. Often, it can be hard to analyze several measured interconnected variables. In the case of EASs, it is not possible to understand all relations of the numerous reconstructed shower properties and make a clear decision about the type of the particle. Instead, a BDT with a few observables as an input provides the best accuracy so far attainable for the decision between photon and proton-induced air showers. Hence, the methods presented in this analysis will be compared to the performance of a BDT, which was implemented as in previous analyses [6, 7].

Boosted decision trees are an ensemble of many separate decision trees, usually with only a few branches and sub-branches. The idea is that a set of weak classifiers creates a single strong classifier. This is done by combining the results of all decision trees to an output value. During the training process, all parameters of the trees have to be adjusted in such a way that the prediction gets as close as possible to the true output value. For that, a loss function is defined, which rates the deviation or quality of the prediction. When the loss function reaches a minimum and does not change anymore after further training iterations, the training is completed.

As already mentioned in section 4.4, five variables were used as an input for the BDT: the zenith angle of the shower axis θ, the calorimetric energy E_γ, the depth of the shower maximum $X_{\max}$, the number of triggered SD stations N_{stat} and the observable S_b. The distributions of these variables, separated for protons and photons, are shown in figure 5.1. While photon and proton-induced showers have different distributions in $X_{\max}$, N_{stat} and S_b, the distributions are the same for θ and E_γ, due to the simulation setup. Nevertheless, these two variables add important information for the BDT, as the other three are energy and zenith dependent.

The analysis was coded in Python [39], and scikit-learn [40] was used for the BDT algorithm. The initialization and training of the BDT was implemented as follows:

```
1 from sklearn.ensemble import GradientBoostingRegressor
2 BDT = GradientBoostingRegressor(n_estimators=1500)
3 BDT.fit(input_variables, type_true, 1/energy)
```

The least squares regression was used by default as the loss function, and 1500 boosting stages to perform were chosen in order to make sure that the training was long enough. The variable `input_variables` is the array of the five input variables for all events that were used during the training, and `type_true` is the corresponding array of the true type of the primary particle, i.e., the desired output of the BDT. The events were weighted with `1/energy` in order to get an E^{-2} spectrum, which is also assumed in the later analysis. The complete simulation sample from section 4.3 was divided into two parts: one larger sample with 100 574 events for the training process and a smaller sample with 20 043 events for testing. By testing the methods on events, which the respective method has not seen during training, it is ensured that the method has not just memorized the events, but is indeed able to generalize to unknown events. In order to ensure a meaningful comparison, this separation between test and training sample is fixed for all methods in this analysis. Both samples contain approximately the same fraction of protons and photons ($\sim 50\%/50\%$).

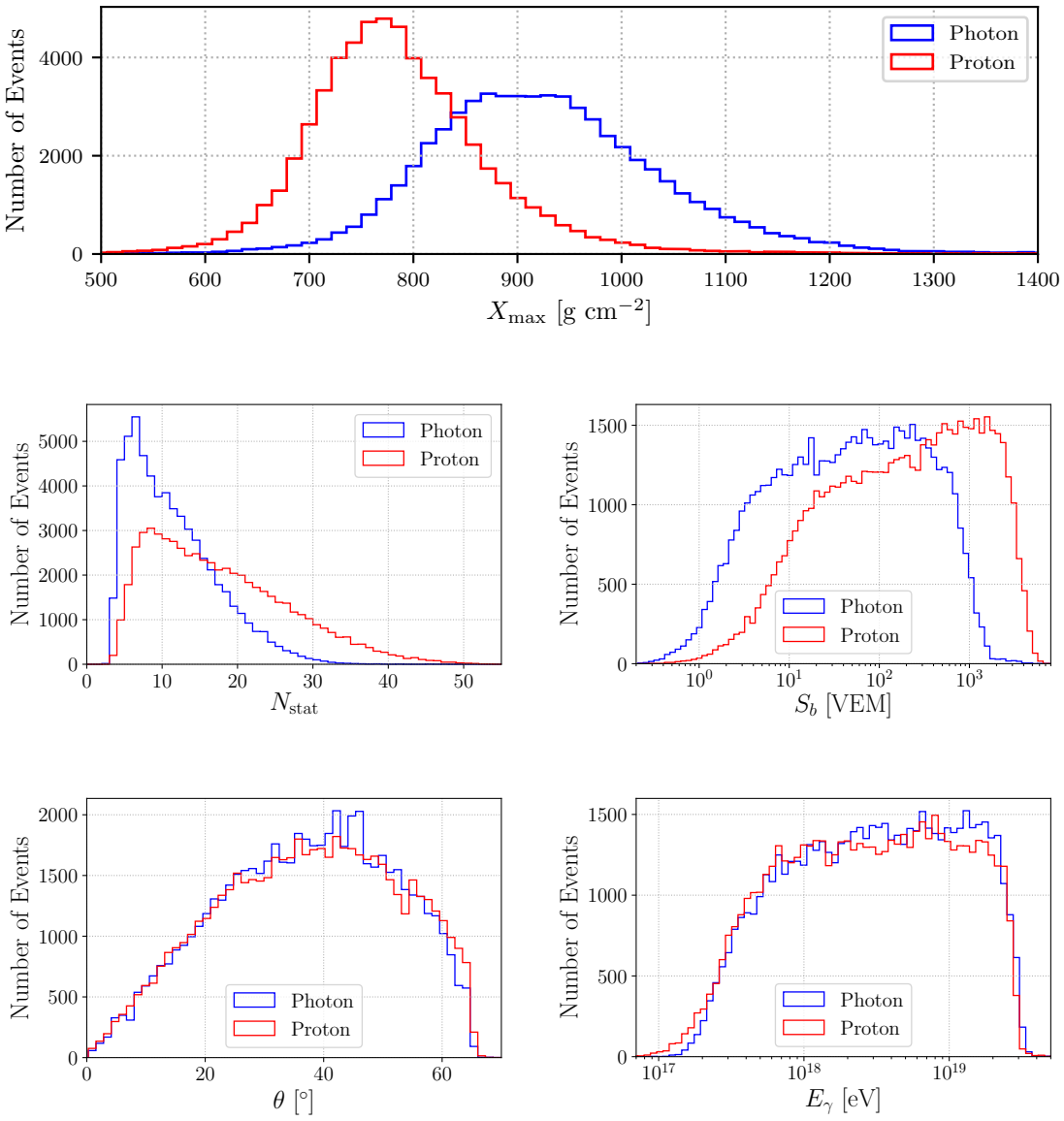


FIGURE 5.1: Histograms of the five BDT input variables. Proton-induced showers tend to have a lower $X_{\max}$ (top), more triggered SD stations N_{stat} and a higher S_b (middle). The θ and E_γ distribution (bottom) are very much the same for photons and protons, since these are specified by the simulation settings.

After the training has been performed, the BDT is applied to all events of the test set. In an ideal case, all protons would be set to zero and all photons to one. Since a perfect distinction is not possible and the output of the BDT is a float value, the result is a distribution between these two values, which is shown in figure 5.2.

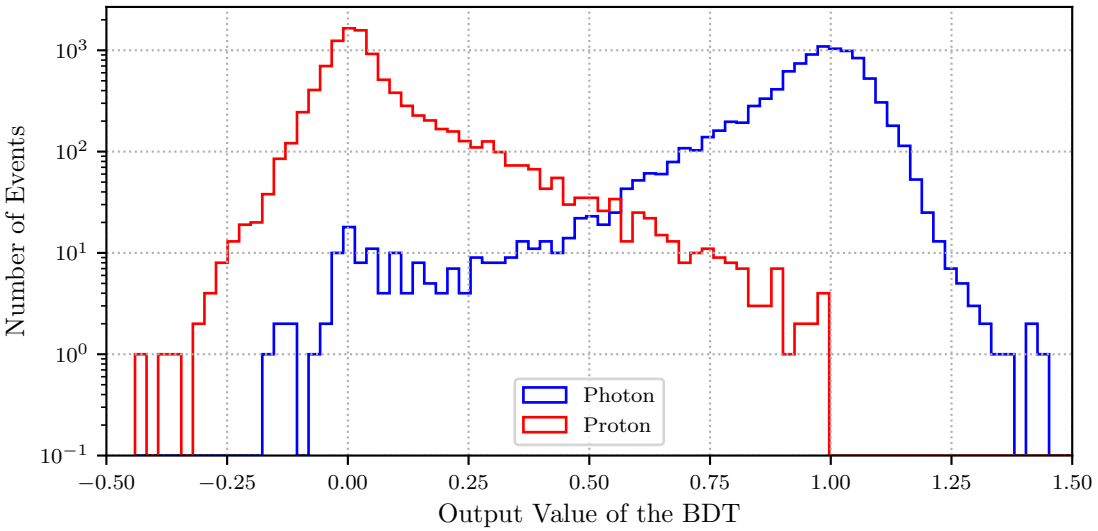


FIGURE 5.2: Histogram of the output values of the BDT for all events in the test sample. Clearly visible is the peak of the proton events at zero and of the photon events at one.

It can be seen that most events were classified correctly, though there are also a few events, where the BDT outputs a value close to the wrong type. A boundary value was chosen where every event with an output value greater than this will be classified as a photon. The higher this value is, the lower is the chance that a proton gets wrongly classified as a photon. But, on the other hand, also more photons will be classified as protons. These two factors must therefore be weighed up against each other. This can be illustrated by so-called receiver operating characteristic (ROC) curves. There, for all possible boundary values, the fraction of photons (signal) that are classified correctly (signal efficiency) and the fraction of protons (background) that are classified correctly (background rejection) is evaluated. In the ideal case of perfect distinction, the background rejection would be equal to one for all signal efficiencies. The energy distribution of the dataset follows an E^{-1} spectrum, so each event is weighted with E^{-1} in order to get an E^{-2} spectrum, which is a typical assumption in photon analyses and makes results comparable. This weighting is done for all upcoming methods. The ROC curve of the BDT is shown in figure 5.3. All methods presented in chapter 7 using DNNs will be compared with the BDT by comparing their ROC curves. At a signal efficiency of 50%, the background rejection reaches 99.91%. This is comparable with 99.85% from [7], though this was applied to a different dataset and different cuts were made.

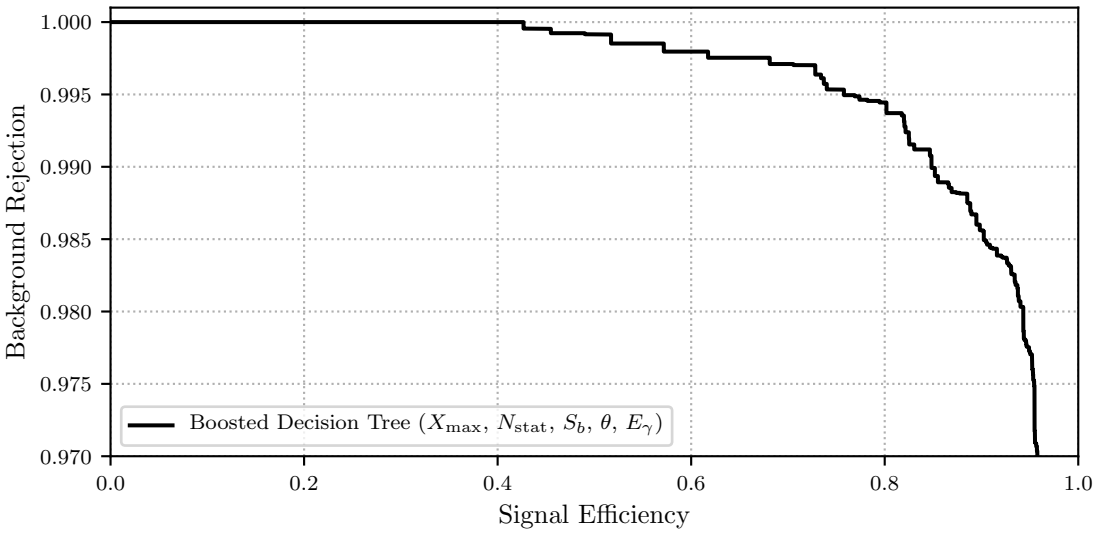


FIGURE 5.3: ROC curve of the BDT method.

Chapter 6

Deep neural networks

6.1 Basic principles

In the last decade, deep learning has become a fast growing field in machine learning. The basic concept of deep learning are artificial neural networks, which consist of nodes and connections that have variable weights. With multiple hidden layers between the input and output, deep neural networks (DNNs) are a special class of neural networks. An example of the structure of such a network is shown in figure 6.1.

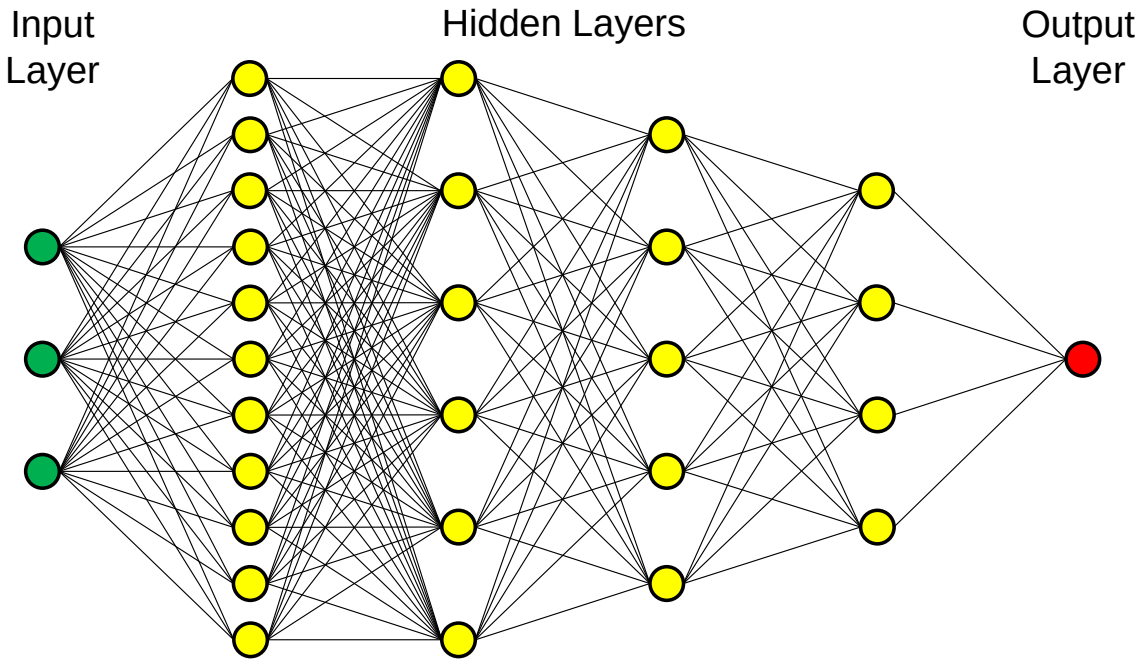


FIGURE 6.1: Illustration of a deep neural network with three input nodes (green), one output node (red) and four hidden layers (yellow) in between. All nodes are connected with independent weights to all nodes of the next layer.

As for almost all machine learning techniques, it is necessary to train the DNN first. Although there are also applications with unsupervised learning, where no explicit label is given during training, this analysis focuses on supervised learning, where each training sample consists of pairs of inputs and true outputs, which the DNN has to predict on the basis of the inputs. During the training process, the weights of the DNN are adapted so that the predicted output, dependent on the given input, fits the true value as well as possible. After the training, the DNN is then able to make predictions on unknown data. How exactly this works is explained in this chapter.

Mathematically, neural networks can be described as several affine transformations. The transformation from layer $\vec{l_i}$ with N nodes (so $\vec{l_i}$ is an N-dimensional vector) to the next layer $\vec{l_{i+1}}$ with M nodes (M-dimensional vector) can be written as

$$\vec{l_{i+1}} = \vec{\sigma}(\mathbf{W}_i \vec{l_i} + b_i) \tag{6.1}$$

where $\mathbf{W}_i$ is the $M \times N$ weight matrix, containing the weights of all connections between the two layers and b_i is the M-dimensional bias vector as an offset. Since multiple linear transformations can be described as a single one, a non-linear activation function $\vec{\sigma}$ is necessary in order to break the linearity. Popular examples of an activation function are the sigmoid function, the rectified linear unit (ReLU) $f(x) = \max(0, x)$ or the scaled exponential linear unit (SELU) [41], which is defined as

$$f(x) = \begin{cases} s \cdot x & x \geq 0 \\ s \cdot \alpha \cdot (\exp(x) - 1) & x < 0 \end{cases} \tag{6.2}$$

where α and s are predefined constants, which are chosen so that the mean and the variance of the inputs are preserved between the two layers. With that, it is possible to create a DNN with a nearly arbitrary number of layers. So for each event, the respective input variables can be given to the network and then, layer by layer, with all the weight parameters, the output can be calculated.

The typical approach for a binary classification problem is to have one output node with two possible values corresponding to the two classes. As mentioned in section 4.4, the value 0 for a proton and 1 for a photon has been chosen. In order to keep the output of the DNN within the interval [0,1] which then corresponds to the probability of a photon, the sigmoid function

$$\mathrm{sig}(x) = \frac{1}{1 + e^{-x}} \tag{6.3}$$

is applied to the last layer.

To evaluate the quality of the prediction of the network, a loss function is defined, which compares the prediction with the true value. For a classification problem with only two classes, the binary cross entropy

$$L(y_{\text{true}}, y_{\text{predict}}) = -y_{\text{true}}\log(y_{\text{predict}}) - (1 - y_{\text{true}})\log(1 - y_{\text{predict}}) \tag{6.4}$$

is the standard choice for the loss function. The objective is to adjust the parameters of the DNN so that $L(y_{\text{true}}, y_{\text{predict}})$ is minimized. The larger the difference between the prediction of the DNN y_{predict} and the true value y_{true} is, the larger is the output of the binary cross entropy. It especially penalizes predictions that are confident and wrong, for example when the DNN predicts a value close to one for a proton event, because the loss function increases rapidly for $|y_{\text{predict}} - y_{\text{true}}| \approx 1$. A perfect model would have $L(y_{\text{true}}, y_{\text{predict}}) = 0$ for every event.

To minimize the loss function, its partial derivative with respect to all parameters is calculated. The parameters are then adapted using backpropagation of the gradients. In other words, the goal is to find a local minimum in a hypersurface with the dimension of the number of parameters. The optimizer minimizes the loss function by using the derivative and changing the parameters in the direction of descent. The learning rate defines the step size of the optimizer in the hypersurface. So-called adaptive optimizers use changing learning rates, which makes them less sensitive to local fluctuations, leading to better performance and faster convergence.

During one so-called epoch, the training set is shuffled and split into many batches, which are gradually given to the DNN. The predictions of the DNN and the loss are calculated for one batch. Then the optimizer adapts the weights and the next batch is loaded. This is repeated until all events of the training set were used, which marks the end of one epoch. A whole training process can last several hundreds of epochs, until the DNN has reached the smallest loss. It is useful to control this process by crosschecking the performance of the DNN after each epoch with another separate dataset, the so-called validation set. Although this smaller data set is never used for training, it should achieve a similar performance as the training set. If it performs significantly worse, this is an indicator for inadvertent memorizing of the training events, which is called overtraining. In the following section, methods to prevent overtraining are explained.

6.2 Regularization

Overtraining is an important challenge to be mastered during training, because it can significantly worsen the final result. It becomes noticeable when the accuracy of the test and validation sample is worse than that of the training sample. Then the DNN has started to memorize the individual events and gets worse at generalizing. There are different reasons for the occurrence of this problem, but also many methods to prevent it; these are described below.

Number of events and parameters. Overtraining can be a sign that the training sample is too small. The number of events required for a good training without overtraining depends on the number of input variables as well as on the size of the DNN, i.e. the number of its parameters. A larger DNN does not necessarily lead to a better result. In fact, more parameters lead to more dimensions of the hypersurface, which makes it more difficult to find a minimum. Furthermore, overtraining is more likely if the network is larger or if there are fewer input variables or events in the training sample. So it is helpful to pay attention to the size of the DNN in order to prevent overtraining.

Dropout. One of the most common techniques for preventing overtraining is dropout. During the training, $\sim 10 - 50\%$ of the nodes are dropped. This fraction is randomly selected again after each batch. This means that there can no longer be certain weights that are excessively important for the DNN, and the input of the following layers is constantly changing. This ensures a more consistent weighting within the network and prevents memorizing of the training sample. After the training, for testing and applying the DNN, no dropout is used, so that the DNN can perform unthrottled.

L1 and L2 regularizers. Overtrained DNNs tend to have few connections with extremely large weights. In order to penalize this, L1 and L2 regularizers are used. The L1 regularizer adds the sum of the absolute values of all weights to the loss function, the L2 regularizer adds the sum of all squared weights. This means that large weights increase the loss function and the weights are pushed to smaller values, in order to minimize it. It is important to balance the prefactors of these extra terms. A factor that is too small has no effect, and a factor that is too large gives a significantly worse result due to underfitting.

Early stopping. At some point during the training, the validation accuracy of the DNN does not improve further. It might happen, that training beyond this point only improves the performance of the DNN on the training sample, but not on the validation sample. Thus the DNN might start to overtrain. In this case, it is necessary to use the early stopping function, which stops the training as soon as the result does not improve anymore or even deteriorates. It must be carefully adjusted after how many epochs, in which the loss function has not improved by a certain value, the training should be stopped.

Reducing learning rate. Closely related to the early stopping function is the reducing learning rate function. The learning rate can be reduced by a selectable factor if the training slows down. This ensures that the targeted local minimum can be approached more precisely, thus improving the final result. Again, the parameters must be fine-tuned to achieve optimal improvement.

6.3 Convolutional networks

Convolutional neural networks are a special form of neural networks. Here, not all input nodes are connected to all output nodes but only spatially adjacent ones are linked. This makes this type of network more efficient to detect local correlations in spatial or temporal arranged input data, which is why it is often used, for example, in image or video recognition. The input data are not processed as a vector, but as an $N + 1$ dimensional tensor. Convolutional neural networks can be interpreted as small filters that stride across the image as shown in figure 6.2. Each filter creates a so-called feature map with extracted features, that were found in the picture while striding over it. With many convolutional layers in succession, the filters stride over these feature maps and the receptive field expands. Therefore, this transition from locally correlated features towards more and more global ones forms a different learning hierarchy compared to standard fully connected networks. After several convolutional layers, the tensor of the last layer is usually flattened, followed by one or more fully connected layers. Each filter has a small set of weights, which is adjusted during the training. Due to weight sharing over the entire image, symmetries in the image can be efficiently exploited. Since much less parameters are needed for these filters than are required for connecting all nodes with one another, convolutional layers can be much more efficient than fully connected layers and drastically reduce the computational costs, especially for large input tensors. Advanced techniques exist that are used in this analysis; they are explained below.

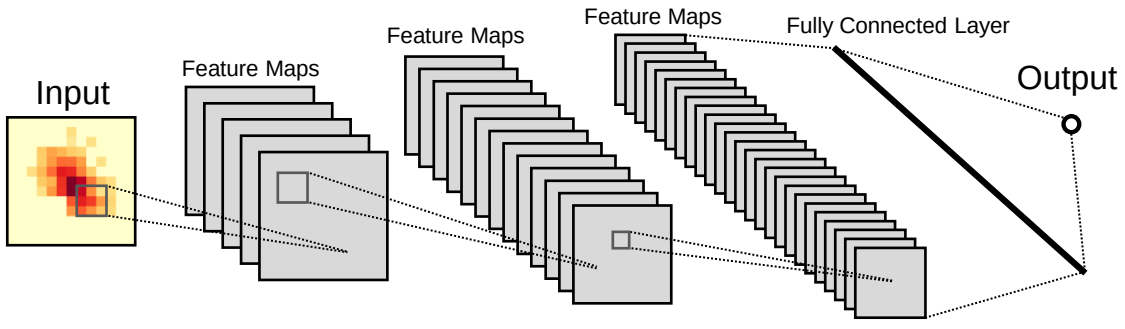


FIGURE 6.2: Illustration of a convolutional neural network. Filters stride over the image creating feature maps. In the end, a fully connected layer of the flattened tensor leads to the final output node.

Striding and padding. Normally, convolutional filters cannot be applied on the border of the image; as a consequence the size of the tensor is reduced. If the size is to remain the same, so-called padding is used, which adds zeros around the borders so that the filters can also be applied at the border of the tensor. But if the tensor is meant to get smaller, then strides can be used. This determines how big the striding steps of the filter are. If, for example, the filter does not strides with step size one, but with step size two, then only half as many positions of the filter are evaluated, which results in a feature map half the size of the first tensor. Both techniques are useful, for example to transform the tensor into the desired shape or to reduce the dimensions of the parameter space.

Pooling. The last convolutional layer is then usually flattened using pooling operations. Global maximum pooling takes the maximum value of each feature map. So a tensor with N feature maps becomes a vector with N values. This is an efficient method to reduce the dimensions of the tensor and the total number of nodes; this reduces the computational costs drastically, increases the learning speed and prevents overtraining.

Separable convolutions. A special form of convolutions are depth-wise separable convolutions [42]. Their spatial and cross-channel correlations are computed separately, which increases computational and parameter efficiency. Depth-wise separable convolutions perform better than normal convolutions, if correlations of different dimensions of the tensor are sufficiently decoupled.

Densely connected layers. In some cases, the performance of a network can be improved by increasing the connectivity with densely connected layers [43]. Here, all convolutional layers are connected to one another, which results in shortcuts through the network. These shortcuts are an essential property, which allows extracted features of different layers to be linked. The result is a better training, and an optimum can be found faster due to an easier backpropagation of the gradient.

Chapter 7

Photon search using deep neural networks

7.1 Network architecture

In this chapter, different approaches to separate proton and photon-induced air showers using deep learning techniques are discussed. Each approach has different input variables, therefore also the architecture of the DNN has to be optimized for the respective input. Each DNN has, therefore, been fine-tuned, for example the choice of the number of layers, their individual sizes, the values of all parameters of the regularizers and so on. Although all networks are different, they all share the same basic concepts for the architecture, which is inspired by the Air shower extraction Network (AixNet) [5], where a DNN was applied on SD measurements. This network architecture can be divided into two blocks.

The first block only processes the signal traces. The second block concatenates the processed output together with the arrival times and the total signals as an additional input. Since the SD and FD data can also be regarded as an image and local correlations between neighboring stations or pixels are very important, convolutional layers are used. As this concept has so far only been applied to the SD, this concept must now be transferred to hybrid measurements. Since both, SD and FD, cover comparable measurements, namely the signal traces and the arrival times, the idea is to set up a separate block for the FD, following the same concept as for the SD. Since the input tensors of the SD (13×13), Coihueco (22×80) and HEAT (22×40) all have different shapes (see section 4.4), it is not very convenient to put all inputs together into one block. Instead, they are processed individually by separated blocks. Then, every block outputs a layer of several nodes, containing the extracted features of the respective input. These layers are then stacked and processed with fully connected layers, leading to one single output node, which is then the

final prediction. The sigmoid function is used as activation for the output layer, so that the output is always a floating point number between zero and one. It is zero for a proton or one for a photon. For all other layers, SELU was used as an activation function.

Since the architecture of the networks for the analysis of the signal traces and arrival times is basically the same for the SD, Coihueco and HEAT, it is described below exemplary for the SD. The networks for the FD differ essentially only in different tensor sizes.

As already mentioned, the signal traces form the input of the first block of the convolutional network. So in the case of the SD, this corresponds to a $13 \times 13 \times 100$ tensor with 13×13 stations containing a signal trace with 100 time steps each. Four 3D convolution filters are used in succession, but each filter has a size of only 1×1 in the spatial dimensions. This means that each signal trace of the stations is analyzed on its own first, but with the same filters for all traces, and the input shape of 13×13 remains the same. This allows to stack the maps of the arrival times and the total signals to the feature maps. The goal of this first block is to extract important temporal features from the signal traces, while reducing the size of the tensor. In this case, it is an advantage of convolutional filters, that the weight-sharing of each filter across all detectors means, that each signal trace is analyzed in the same way. This is both physically meaningful and parameter-efficient. After these four convolutional layers, the network has extracted 30 feature maps.

In the second block, these 30 feature maps are stacked with the maps of the arrival times and the total signals, thus creating a $13 \times 13 \times 32$ tensor. This is followed by several layers of densely connected 2D separable convolutions. So every layer is connected to all previous ones. Each of the initial 32 feature maps contains different extracted features with different physical meanings. It is thus expected that the spatial and feature-wise correlations are decoupled. Therefore, separable convolutions can be applied. The size of the filters is 3×3, which also allows to extract spatial correlations and features across stations. All convolutional layers are initialized using the normal distribution proposed by K. He et al. [44]. In the end, feature maps are extracted which contain the most important observables of the signal traces, the arrival times and total signals. Finally, each feature map is compressed to one node using global maximum pooling. The resulting vector is fully connected to the output node or concatenated with vectors of other input blocks, depending on the respective approach.

In summary, the inputs of the different detectors (SD, Coihueco, HEAT) are first analyzed separately by the DNN using the AixNet architecture. Each of these blocks contributes a final vector containing the extracted features. They are concatenated and analyzed using fully connected layers, leading to one final output node.

For all networks which will be discussed below, the code of the implementation with details of the architectures can be found in appendix A. Keras [45] is used as the deep learning framework and the TensorFlow [46] software as a backend. For the training process, the adaptive optimizer ADAM [47] with an initial learning rate of 10^{-3} was used. 95 000 events were used for the training and 5574 events for validation. After the training process, the final performance of the DNN is checked with the test set of 20 043 events (i.e. 120 617 events in total). Each set consists of approximately 50% protons and 50% photons. It is impossible to store all input tensors of all events at the same time in the main memory. Hence, only 10 events were loaded at the same time. This rather small batch size is needed in order to reduce computational costs.

7.2 Network using only BDT input variables (BDT-Net)

The performance of all DNNs in the following analysis will be compared to a previous method, which uses a BDT and was described in chapter 5. Hence, before discussing the AixNet-based networks, it might be an interesting test to check the power of DNNs by directly comparing the BDT method to a network with the same input variables. So, a DNN was set up, which has five input nodes, corresponding to the five BDT input variables (S_b, X_{max}, N_{stat}, θ and E_γ) and seven fully connected layers in total. The distribution of the predictions on the test set of this network using only BDT input variables (BDT-Net) is shown in figure 7.1.

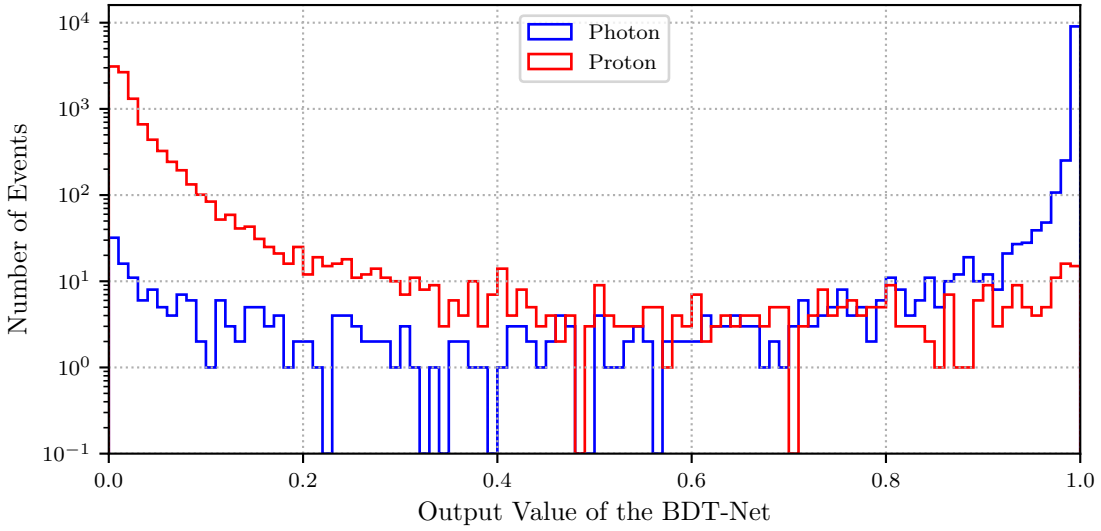


FIGURE 7.1: Histogram of the output values of the BDT-Net for all events in the test sample.

The distributions of all other DNNs that will be described below are very similar to this. A large peak of photon events at an output value close to one and also one of proton events close to zero is visible. This means that the DNN has successfully learned how to separate photons and protons. Only a small fraction of events gets a false prediction value. For each particle type there is also a smaller peak at the opposite end of the output value range. Especially the peak of photon events at zero is clearly visible. This has two reasons: first, wrong classification of protons as photons by the method would be worse than wrong classification of photons as protons. To prevent the former error, a weighting was applied to the events during training. The protons get half the weight of the photons, so that the DNN tends to zero as an output rather than to one. This leads to more wrongly classified photons, but also suppresses the number of protons with one as output. The second reason is the activation of the DNN in the last layer. As mentioned in the previous section, the sigmoid function was used as an activation in order to map the output of the DNN to the interval between zero and one. Considering the shape of this function, it is clear that the peaks at the edges are due to the fact that values with $|x| \gtrsim 3$ are mapped very close to zero or one. In figure 7.2, the output values of the same test set as before are plotted, however, before the sigmoid activation was applied. It can be seen that this distribution has two fading ends and a larger range $[-20, 13]$ of output values. All values with $|x| > 4.6$ are stacked in both outermost bins in figure 7.1.

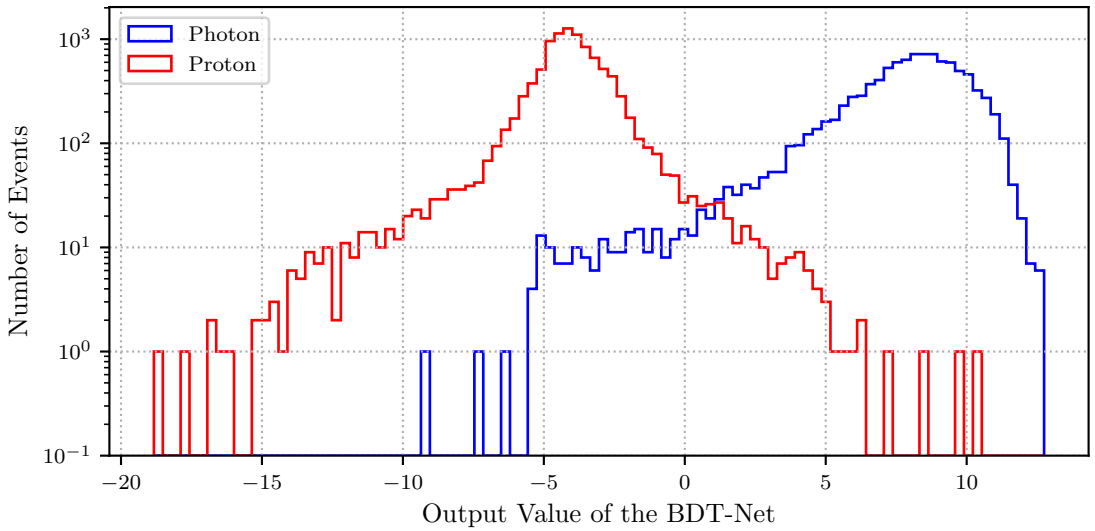


FIGURE 7.2: Histogram of the output values of the BDT-Net for all events in the test sample before the sigmoid activation was applied. Therefore, the output range is unlimited. The plotted range contains all events.

In order to compare the performance of the BDT-Net with the BDT method, the background rejection and signal efficiency for every possible decision value was calculated. The resulting ROC curve is shown in figure 7.3. The overall performance of the BDT-Net is better than the BDT. With 99.99% background rejection at 50% signal efficiency, the BDT-Net performs significantly better than the BDT method (99.91%). This is an interesting result, because the input is exactly the same for both methods. This means that for this purpose alone it is useful to apply DNNs. Nevertheless, it should be checked whether the differently chosen event cuts influence the result and how well the simulations fit. In the following sections, it will be explained how the strength of DNNs and especially convolutional networks can be used to further improve this result.

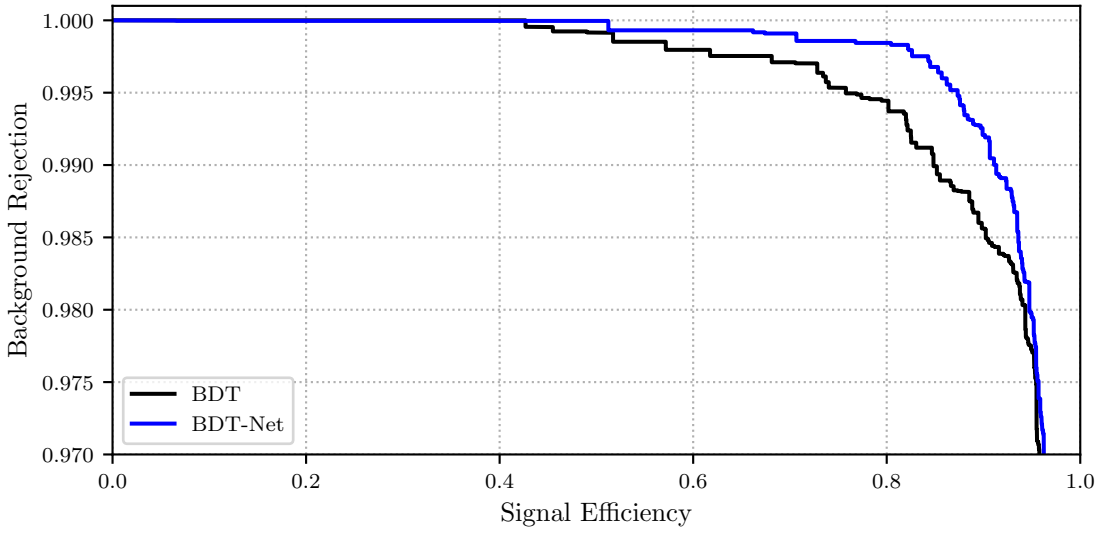


FIGURE 7.3: ROC curve of the BDT-Net, compared to the BDT method. The BDT-Net clearly performs better.

7.3 Surface detector network (SD-Net)

The BDT method and the DNN from the previous section use some variables that were calculated from FD or hybrid data. However, the FD has a duty cycle of only around 15% compared to almost 100% of the SD. A method that uses only SD data and still is sufficiently accurate would significantly increase the exposure for photon search. This could make the discovery of a UHEP more likely or could lower the upper limits on the possible photon flux at these energies. That is why a DNN was set up that uses only the measurements of the SD. A sketch of this SD-Net is shown in figure 7.4. Since this DNN only gets the signal traces, arrival times and total signals of the SD, the architecture of the SD-Net is very similar to the AixNet by which it was inspired, as described in section 7.1.

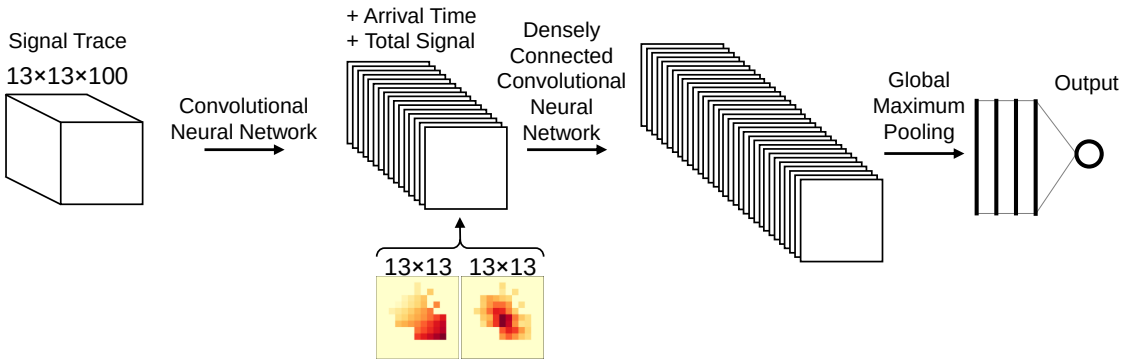


FIGURE 7.4: Sketch of the SD-Net. The signal traces are analyzed with four convolutional layers first. The resulting feature maps are concatenated with the arrival times and total signals and analyzed with four densely connected separable convolutional layers. The feature maps are compressed using global maximum pooling, followed by three fully connected layers and, finally, the output node.

The ROC curve of the trained SD-Net is shown in figure 7.5. It does not perform as well as the BDT at higher signal efficiencies, but below 70% signal efficiency, both methods perform almost equally well. This shows that the AixNet architecture can also be used to search for photons. With a background rejection of 99.91% at 50% signal efficiency, the SD-Net achieves the same value as the BDT; this is a good success considering that only SD data were used. This means that the use of the SD-Net for the UHEP search benefits from the higher duty cycle of the SD and thus a higher exposure, without losing accuracy.

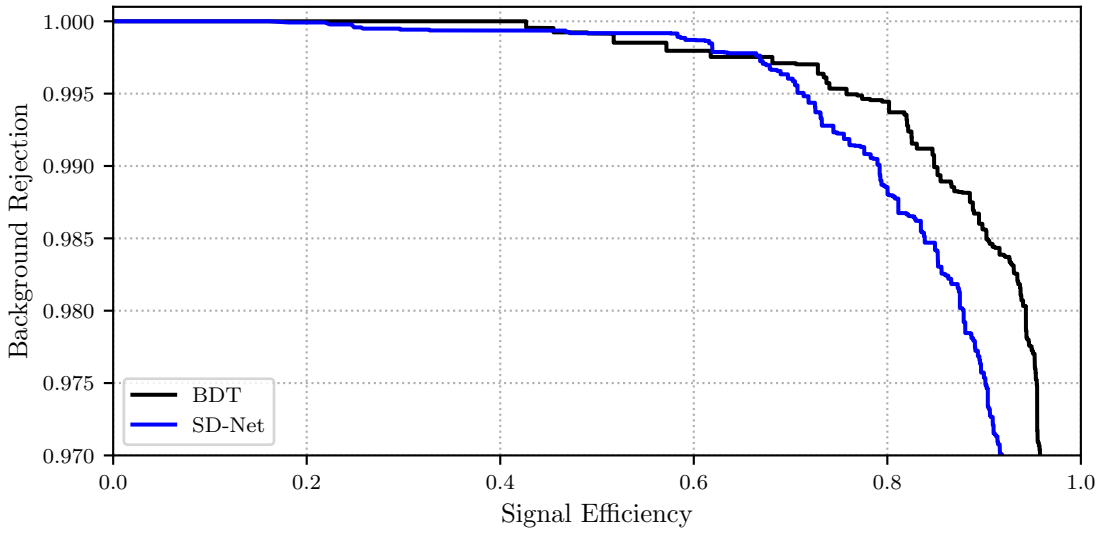


FIGURE 7.5: ROC curve of the SD-Net, compared to the BDT. Both methods perform almost equally well below 70% signal efficiency.

7.4 Fluorescence detector network (FD-Net)

In order to check the separation power of the FD, a DNN was set up, called FD-Net, with the signal traces, arrival times and total signals of the FD as input. A sketch of this FD-Net is shown in figure 7.6. The architecture is like that of the SD-Net, only the shapes of the tensors differ, as the SD has a $13 \times 13 \times 100$ grid whereas Coihueco has $22 \times 80 \times 50$ and HEAT has $22 \times 40 \times 50$ inputs. The AixNet principle was applied separately to Coihueco and HEAT. Then, after the pooling layer, each block provides a layer with 100 nodes, containing all extracted features. These two layers were concatenated and, after three fully connected layers, lead to the output node.

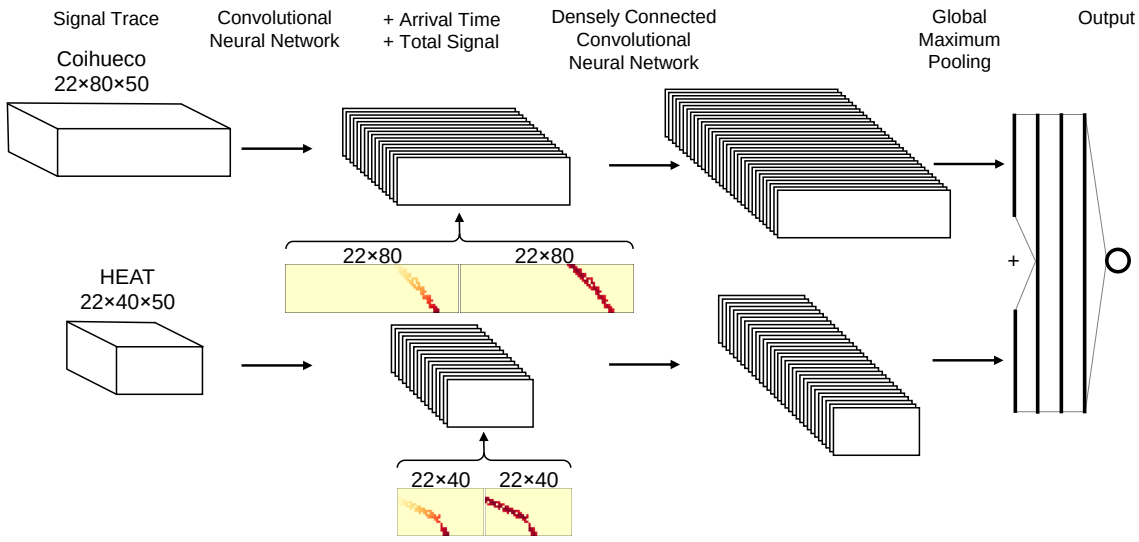


FIGURE 7.6: Sketch of the FD-Net. The input tensors of Coihueco and HEAT are analyzed separately in the same way as in the SD-Net. After applying global maximum pooling, the two resulting layers are concatenated, followed by three fully connected layers and the output node.

The ROC curve of the trained FD-Net is shown in figure 7.7. The performance of this DNN is very bad compared to the previously described methods. It seems that the DNN was not able to extract useful information from the FD traces input. A reason might be that the arrays are too large, considering in particular that most pixels do not carry any signal and, as a result, most traces contain zeros only. This is why another smaller network was tested excluding HEAT data. This network has only Coihueco data as an input and consequently only one convolutional block. As visible in figure 7.7, the ROC curve of this smaller DNN is almost identical with the one of its extended version. The DNN was thus not able to extract any important features from the HEAT data. In fact, only about half of all events have HEAT data at all. With a background rejection of around 75% at 50% signal efficiency, the performance of the FD-Net, both with and without HEAT, is far from being acceptable. It has not been unequivocally clarified whether these networks

perform worse because the input contains less useful information, the data are not provided properly or if it is simply a problem of optimizing the parameters during training. It is probably a combination of all these reasons, which is why further investigations would be necessary to fully understand what is the best way to feed FD data to a DNN and train it. So FD alone currently has no satisfactory separation power, but it might be useful as supporting information when combined with other inputs.

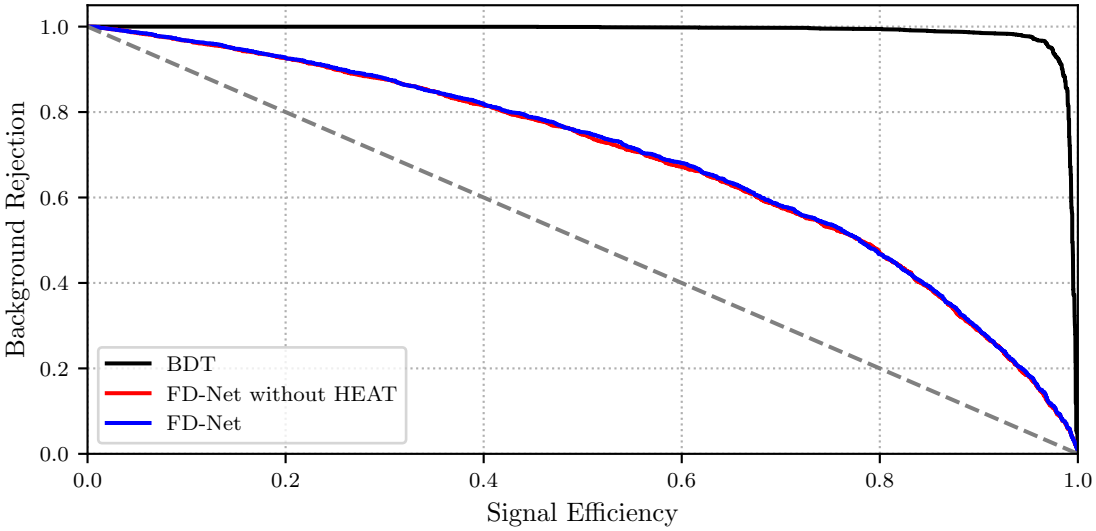


FIGURE 7.7: ROC curve of the FD-Net, compared to the BDT. The dotted line shows the diagonal that would result if only random guesses were made. Both FD-Nets perform far worse than the BDT.

7.5 Hybrid network (H-Net)

An important test is to check the performance of a DNN whose input comprises all possible variables and tensors. This Hybrid network (H-Net) gets the complete data of SD, Coihueco, HEAT and the five BDT variables. Therefore, it has four separate blocks that first analyze each input on its own; then the pooled layers with the extracted features are concatenated and analyzed by several fully connected layers. A sketch of this H-Net is shown in figure 7.8. In principle, this network is a combination of all DNNs that were discussed before. Naively, one would expect this network to perform best as it gets the most information as input. With around 560 000 parameters, it is also the largest network in this analysis. Due to the large input tensors and the huge amount of parameters, the training of this DNN is computationally very expensive and takes several days, compared to a few hours for the other networks.

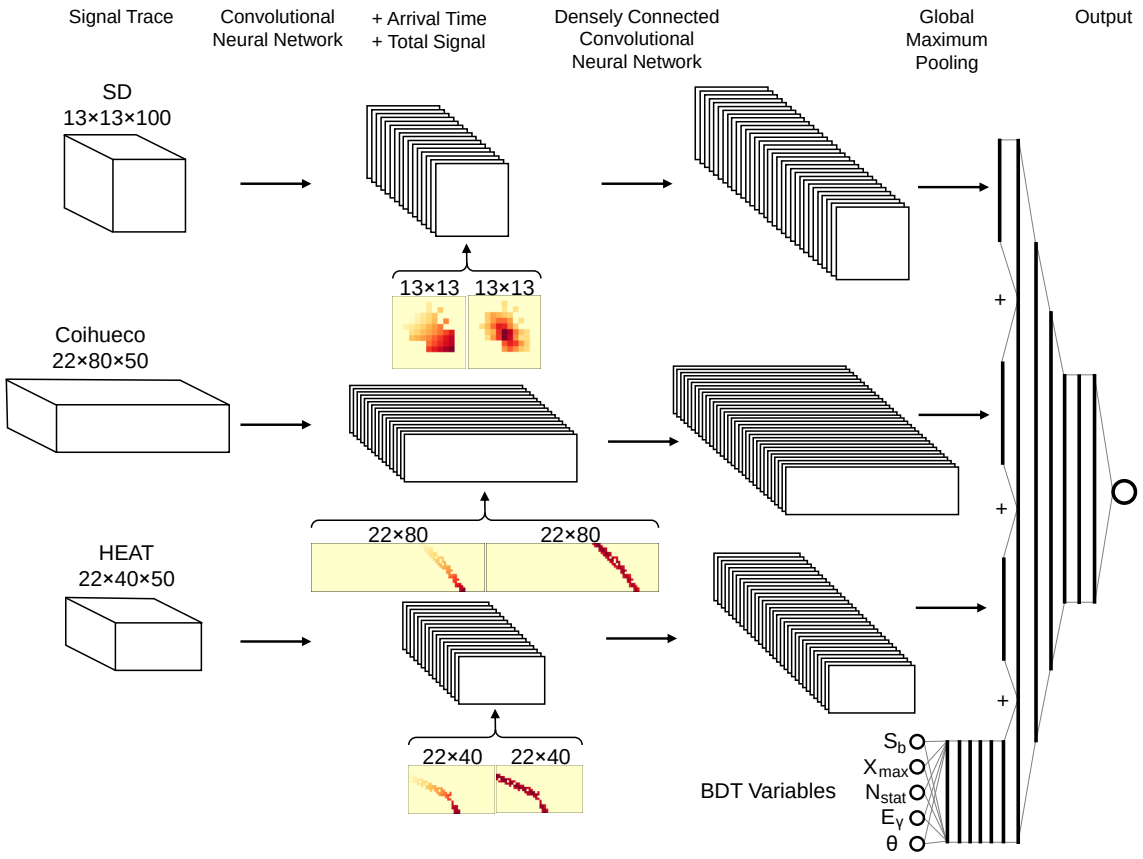


FIGURE 7.8: Sketch of the H-Net with SD data, FD data (Coihueco and HEAT) and the five BDT variables as input. The tensors of SD, Coihueco and HEAT are processed as in the SD-Net and FD-Net. After applying global maximum pooling, the three resulting layers are concatenated with the last layer of the BDT-Net. This is followed by six fully connected layers and the output node.

The results are shown in figure 7.9. It can be seen, that the performance of the H-Net is in general better than that of the BDT, although, contrary to expectations, the ROC curve of the H-Net is very close to that of the BDT-Net, which means that the additional input did not improve the accuracy. In view of the high computational costs of training, application of the BDT-Net is more convenient in direct comparison. Furthermore, networks without HEAT or FD data were trained. These smaller networks again performed almost equally well as the BDT-Net and H-Net.

This shows that neural networks do not necessarily work better when the amount of information is increased. More input variables can even worsen the result. It is important to develop a reasonable architecture that can extract the essential features. To build a DNN which achieves good accuracy using FD data turns out to be more difficult than for the SD. Using the AixNet architecture, the FD-Net performance seems to be too bad to provide improving information when it is included in the H-Net. Therefore, more research has to be done in order to make use of the additional input. However, the size of the input and of the DNN, and the resulting long training times make it harder to fine-tune the parameters and

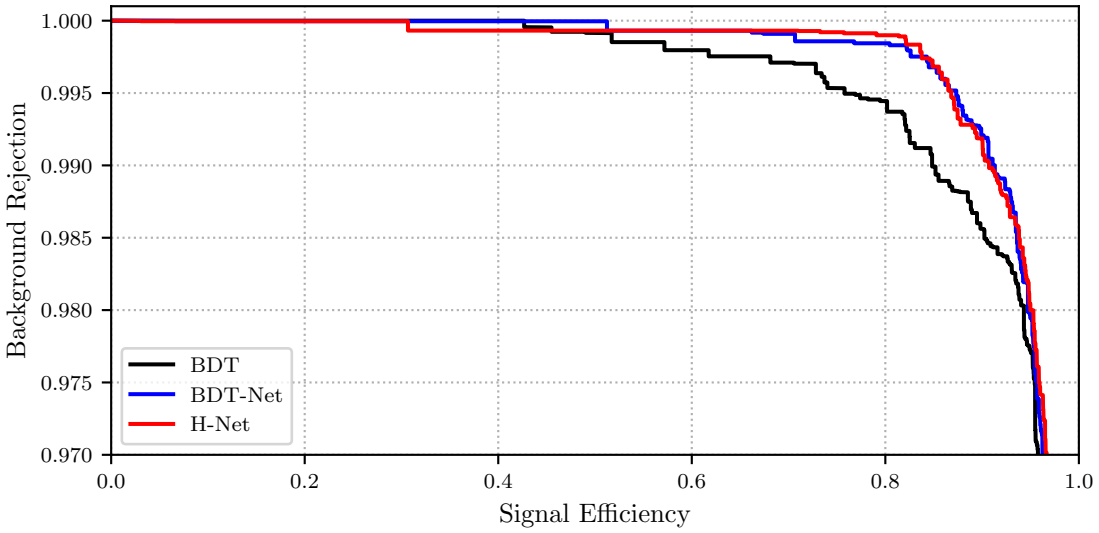


FIGURE 7.9: ROC curve of the H-Net, compared to the BDT and BDT-Net. The H-Net has approximately the same separation power as the BDT-Net.

the general structure of the network. This extends the required time for the optimization process. Another problem might be the huge discrepancy of the importance of different inputs. While only five BDT variables already achieve a good accuracy and thus have a high separation power, the FD tensors have 137 280 variables in total and carry less useful information like empty pixels. Therefore, it is not surprising that the network is likely to focus on the BDT variables and has problems to find important features in the large tensors to further improve the result. It should be noted that the BDT variables were optimized for years, whereas the detector signals are raw data. This might be the reason why the H-Net does not perform better than the BDT-Net. For a successful improvement of this result, more effort must be made to find the right implementation of these data into the network. For now, the best choice is the BDT-Net for performance reasons. However, the SD-Net might be useful as well, because of the higher duty cycle. Though, it is also possible that a limit has already been reached. The fact that the BDT-Net, the H-Net as well as the other tests with hybrid input all achieve approximately the same accuracy, might show that these results cannot be improved any further. It is possible that the last few events that were wrongly classified are indeed indistinguishable. This would mean that this result is close to the best possible, using the current detector configuration.

7.6 AugerPrime network (AP-Net)

The upgrade of the Pierre Auger Observatory, AugerPrime, promises improvements of many measurements and reconstructions of the shower properties. Since the SSDs on each WCD of the SD have a different response to the muonic and electromagnetic component of the shower than the WCD, it becomes possible to reconstruct the individual strength of the signal of these two components. A large improvement for UHEP search is expected, because photon and proton-induced EASs differ in the fraction of muonic and electromagnetic component. To give an outlook of the separation power of AugerPrime, a DNN was developed with the Monte Carlo numbers of electromagnetic and muonic particles. A sketch of the architecture of this AugerPrime-Net (AP-Net) is shown in figure 7.10. The input consists of three maps with 13×13 SD stations, containing the true number of muons ($\mu^{\pm}$), electromagnetic ($\gamma, e^{\pm}$) and total particles, that had traversed the station. These particle numbers were preprocessed by logarithmizing and normalizing them with the transformation

$$\tilde{N}_{\mathrm{X}} = \frac{\log_{10}(N_{\mathrm{X}} + 1)}{\log_{10}(\max(N_{\mathrm{X}}) + 1)} \tag{7.1}$$

where $\max(N_{\mathrm{X}})$ is the maximum number of particles of all stations and all events ($\max(N_{\mu^{\pm}}) = 9033$, $\max(N_{\gamma,e^{\pm}}) = 4\,652\,586$, $\max(N_{\mathrm{total}}) = 4\,654\,171$). The resulting $13 \times 13 \times 3$ tensor is then given to the AP-Net as an input. The network consists of four convolutional layers and, after a global maximum pooling layer, another four fully connected layers.

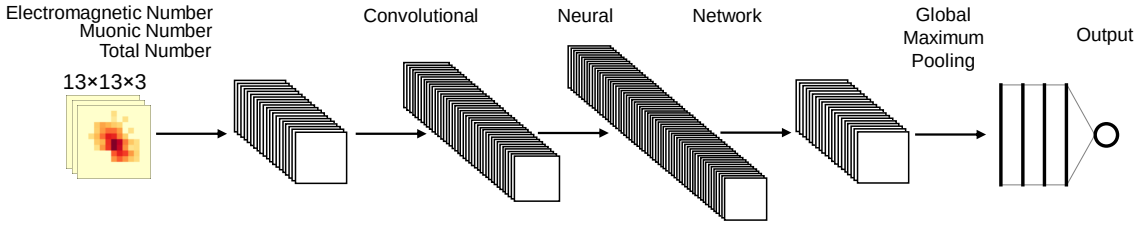


FIGURE 7.10: Sketch of the AP-Net with the stacked maps of the electromagnetic, muonic and total number of particles per SD station as input. These maps are analyzed by four convolutional layers. The feature maps are compressed using global maximum pooling, followed by three fully connected layers and the output node.

The ROC of this relatively small DNN is shown in figure 7.11. As expected, the performance of this network by far surpasses all other methods. Since it uses SD data only, it also benefits from its higher duty cycle and exposure. However, this shows only the ideal case of the separation power of AugerPrime, assuming that the particle counts of the two components can be perfectly reconstructed. Therefore, the real separation power of AugerPrime will be worse. But this result shows that this upgrade can significantly improve the search for UHEPs.

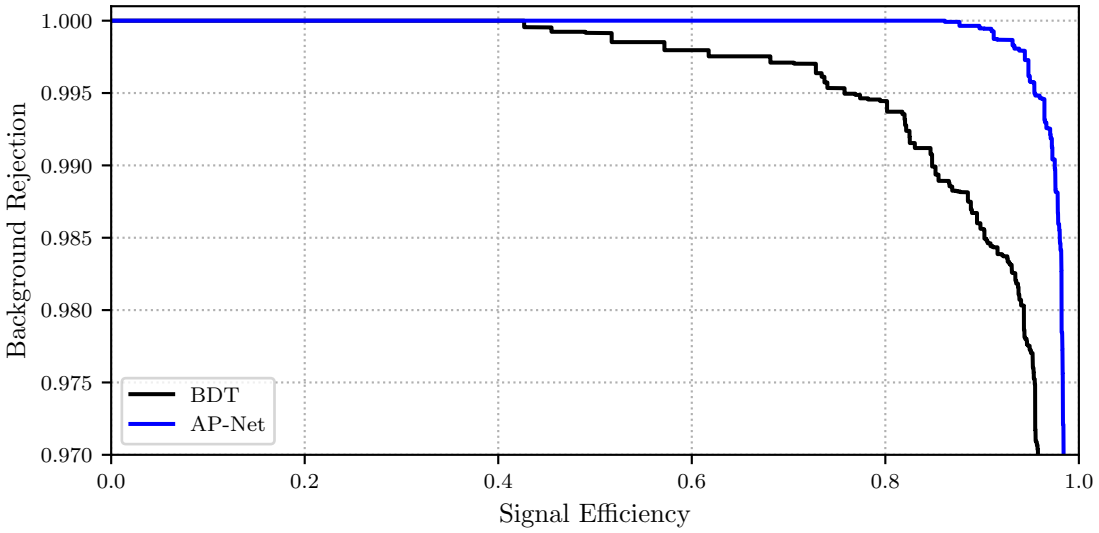


FIGURE 7.11: ROC curve of the AP-Net method, compared to the BDT. It can be seen that the AP-Net performs far better than the BDT.

7.7 Stability of performance

After having discussed several approaches with different architectures and inputs, it is important to check the stability of these results. Before these DNNs are applied to measured data, a lot of tests have still to be made. These include, for example, an analysis if the networks really train on properties of the shower or just on artifacts of the simulation. Therefore, more research regarding the differences between data and simulation, especially for the signal traces, has to be done.

A first test is to check the performance of the DNNs on different hadronic interaction models. As already mentioned in section 4.1, two additional test sets were simulated, using QGSJetII-04 and Sibyll 2.3c as hadronic interaction models. As an example, the ROC curves of all three models for the BDT-Net are shown in figure 7.12. Since only protons were simulated in the two additional data sets, the signal efficiency cannot be calculated. Instead, for each chosen decision line, the signal efficiency for the main test set (using EPOS LHC) was calculated, and the background rejection was calculated on the test set of the respective model, using the same decision line. It can be seen that, with some fluctuations, the performance on each interaction model is roughly the same. These fluctuations are normal and can also be seen for identical networks that are trained individually and applied on the same test set. This means that the DNN has not only trained on specific properties of the EPOS LHC simulation, which gives a certain confidence in this neural network method.

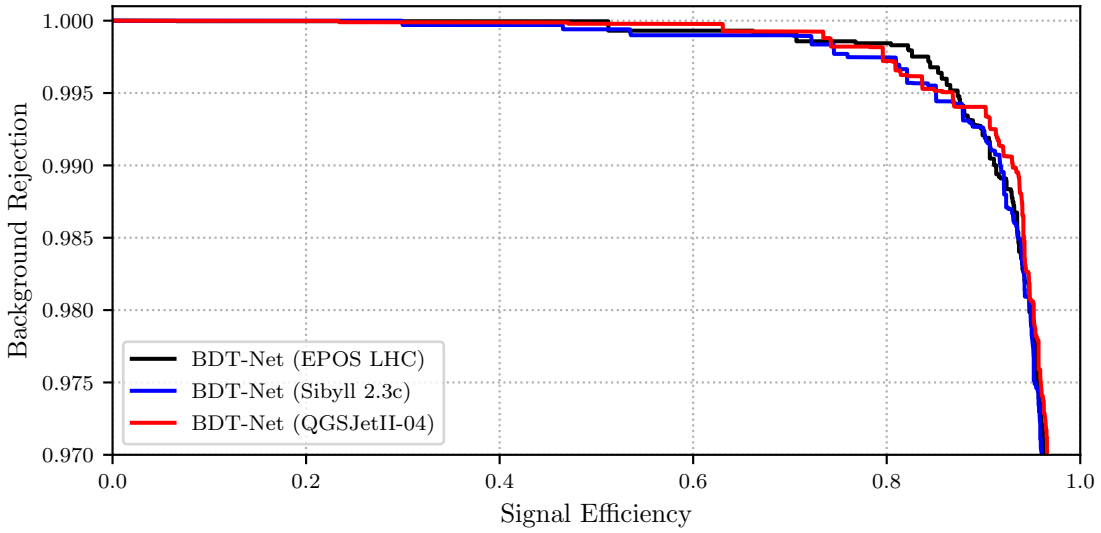


FIGURE 7.12: ROC curves of the BDT-Net for three test sets with different hadronic interaction models. The curves overlap with only small fluctuations.

Another test is to check the energy dependence of the methods. In order to do that, the decision line was set arbitrary to a DNN output value of 0.5, i.e. in the middle of the interval of possible outputs. All output values greater than 0.5 were classified as photons. The resulting histogram for the BDT-Net applied to the test set is shown in figure 7.13.

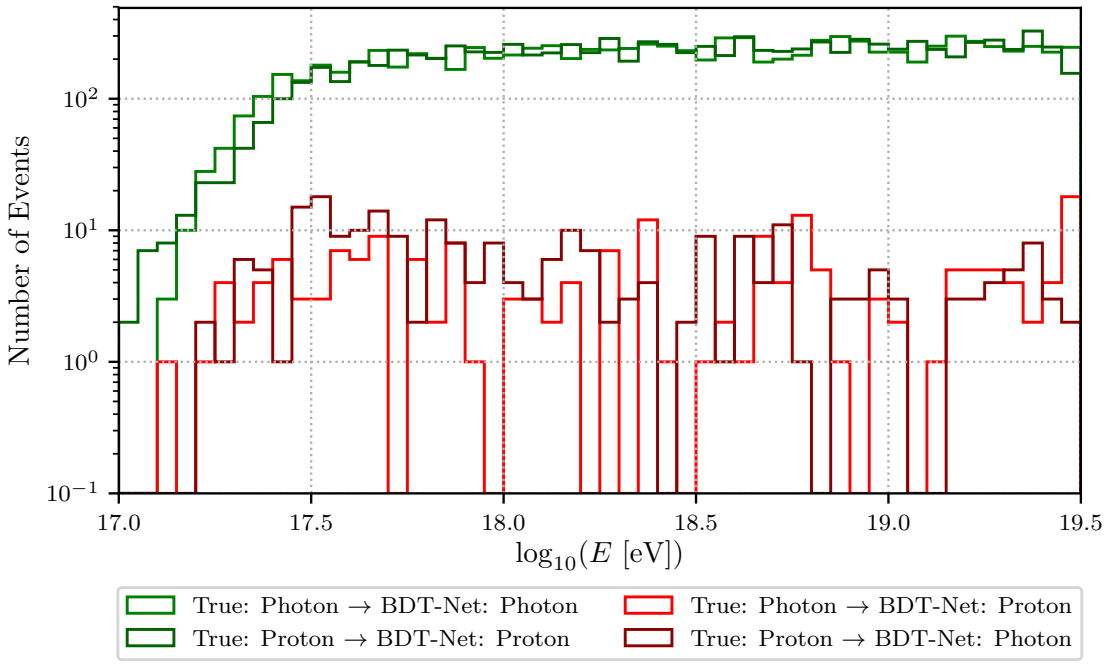


FIGURE 7.13: Histogram of correctly and wrongly classified events of the test set, using the BDT-Net. The cuts applied on the data set reduces the statistics below $E = 10^{17.5}$ eV. The fraction of wrongly classified events is approximately constant.

It can be seen that the accuracy of the DNN is almost energy independent. With lower primary energy of the cosmic ray, it is expected that it becomes harder to distinguish between photons and protons, since their $X_{\max}$ distributions get more similar. This dependence can be neglected, because its effect on the test set is very small. Nonetheless, an energy threshold could reduce possible energy dependencies. Although both tests verify the stability of methods using DNNs, a lot of tests have still to be made before applying them on measured data. For example, it is important to fully understand the differences between data and simulation, as well as what the features are that the DNN takes into account in its decision.

7.8 Discussion of results

Several approaches of DNNs using different input sets were discussed. The BDT-Net has a much better performance than the BDT even though both get the same five input variables. The SD-Net does not quite reach the accuracy of the BDT for high signal efficiencies, but it is comparable below 70%. This is a good result, considering the higher duty cycle of the SD. The integration of the FD data is problematic and the FD-Net is by far the worst method that was developed in this analysis. The H-Net could not use the additional input in order to outperform the BDT-Net, which leads to the hypothesis that a limit is reached when these deep learning techniques and detector measurements are used. Though, more research has to be done to find a better approach to analyze FD data with neural nets. The ongoing upgrade AugerPrime might increase the separation power of the SD enormously, as the AP-Net shows as a best case scenario. For better comparison, the ROC curves of all important DNNs developed are shown in figure 7.14. The methods show good stability over the tested energy ranges and were also tested on two additional hadronic interaction models. The performances are all in all stable, which means that the DNNs will probably provide good results on measured data. However, more tests have to be made to prevent false discoveries, because of the differences of simulation and data.

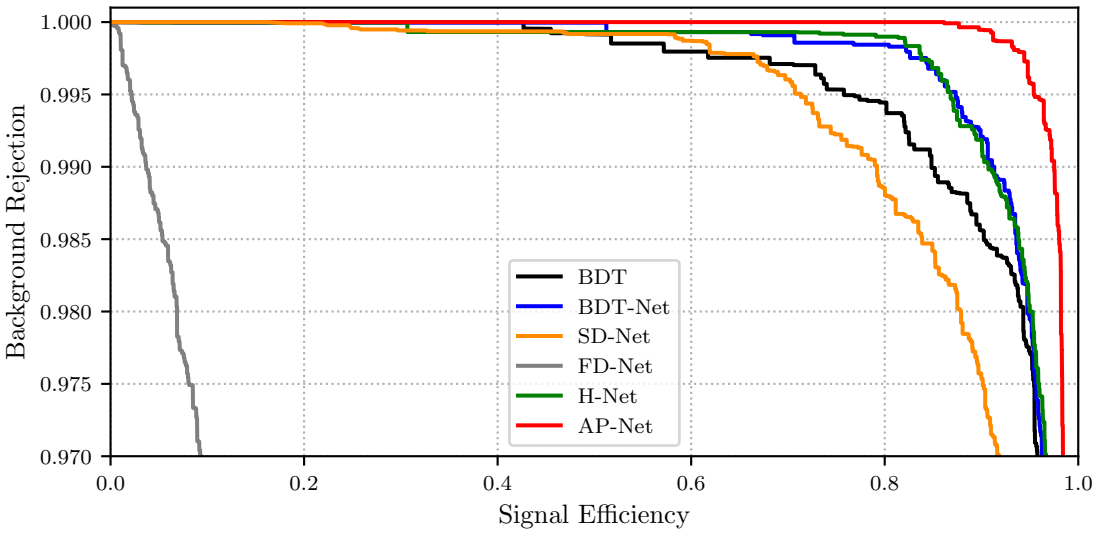


FIGURE 7.14: ROC curves of all methods in comparison.

Chapter 8

Estimation of the photon flux sensitivity

Since this analysis only presents new methods that can improve the separation between photons and protons, the actual search for photons in the data of the Pierre Auger Observatory has still to be done in the future. Before, the mismatch between simulation and data has to be investigated further. The impact on how the energy range of the simulated data influences the performance of the methods needs to be studied as well.

No UHEP has been found so far. Despite the fact that the developed deep learning approaches increase the current accuracy, it is unlikely to find a UHEP. The sensitivity on the photon flux is still orders of magnitudes higher than the predicted flux caused by the GZK effect (see section 2.1). So it is important to lower the upper limits of a possible photon flux drastically using improved methods to either prove or disprove the GZK effect. In order to get an idea of the possible improvement of these limits, an estimation was calculated for the different approaches. The SD-Net benefits from the higher duty cycle and, therefore, higher exposure. The performance of the hybrid approaches is better than that of the BDT. So both might lower the current limits.

The following calculation was redone from [48], where the upper limits on the photon flux were calculated using the BDT method. First, the effective exposure $\mathcal{E}_{\gamma,\text{eff}}$ above an energy threshold E_0 is calculated:

$$\mathcal{E}_{\gamma,\text{eff}}(E_\gamma > E_0 | E_\gamma^{-\Gamma}) = \frac{\int_{E_0}^{\infty} E_\gamma^{-\Gamma} \mathcal{E}_\gamma(E_\gamma) \mathrm{d}E_\gamma}{\int_{E_0}^{\infty} E_\gamma^{-\Gamma} \mathrm{d}E_\gamma} \tag{8.1}$$

where $\Gamma = 2$ is the assumed spectral index of the photon flux and $\mathcal{E}_\gamma(E_\gamma)$ the hybrid exposure. It is shown in figure 8.1.

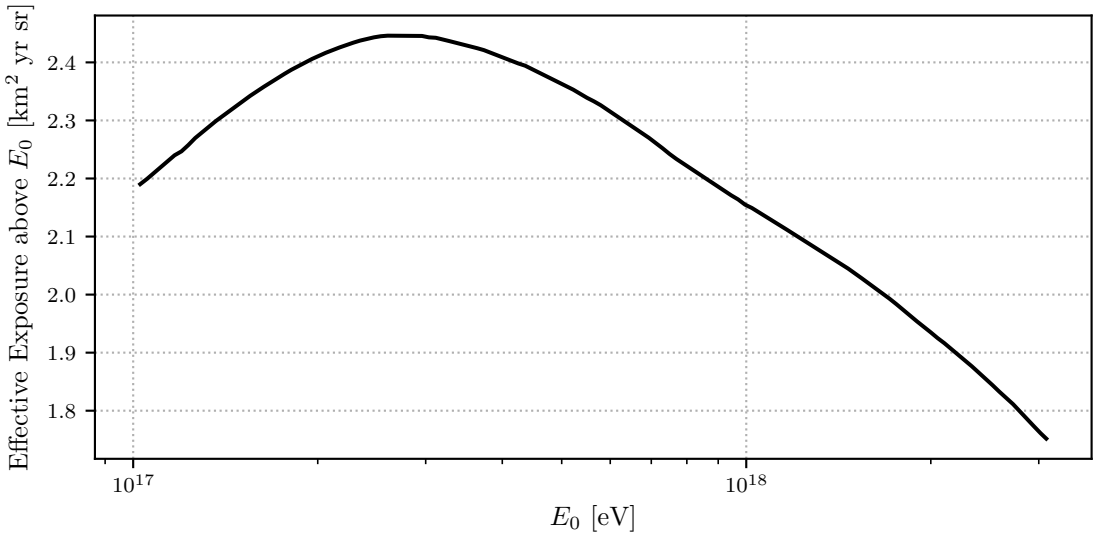


FIGURE 8.1: Effective exposure $\mathcal{E}_{\gamma,\mathrm{eff}}(E_\gamma > E_0|E_\gamma^{-\Gamma})$ above an energy threshold E_0. The FD reaches full efficiency at the maximum of the curve. Data taken from [48].

Then the upper limit on the integral photon flux at 95% confidence level $\Phi^{0.95}_{\mathrm{U.L.}}$ can be determined as

$$\Phi^{0.95}_{\mathrm{U.L.}}(E_\gamma > E_0) = \frac{N^{0.95}_{\gamma,\mathrm{cand}}(E_\gamma > E_0)}{\epsilon_{\mathrm{cand}} \times \mathcal{E}_{\gamma,\mathrm{eff}}(E_\gamma > E_0|E_\gamma^{-\Gamma})} \tag{8.2}$$

where ϵ_{cand} is the signal efficiency of the a-priori photon candidate cut and $N^{0.95}_{\gamma,\mathrm{cand}}(E_\gamma > E_0)$ is the upper limit on the number of photon candidate events, according to Feldman and Cousins [49] with no background subtraction. In order to calculate the upper limit, a Poisson distribution with background

$$P(n|\mu) = \frac{(\mu + b)^n}{n!}\mathrm{e}^{-(\mu+b)} \tag{8.3}$$

is assumed with the expected background events b, the number of candidate events found n and the expected number of candidate events μ. In the previous analysis [48], with 99.85% background rejection at 50% signal efficiency $b = 0$ background events were assumed. Since no reduction of zero background events is possible, the only way to improve the upper limits is to achieve a higher exposure or signal efficiency. Therefore, the signal efficiency ϵ_{cand} for a background rejection of 99.85% was calculated for each method. The result is shown in table 8.1.

Method	Back. rej. at 50% sig. eff.	Sig. eff. at 99.85% back. rej. (ϵ_{cand})
BDT	99.85%	50%
BDT-Net	99.99%	70.66%
SD-Net	99.91%	61.17%
H-Net	99.93%	82.07%
AP-Net	100%	92.64%

TABLE 8.1: Comparison of different methods at 50% signal efficiency or 99.85% background rejection. The right column was used as ϵ_{cand} in equation 8.2.

For a given n and b, the calculation of the Feldman-Cousins confidence intervals outputs the maximum possible number of expected candidate events within a certain confidence interval. In other words, if n photon events are found and b wrongly classified background events are expected, it is calculated how many true photon events at most can be in the sample with a certain probability. The assumption that there is no background ($b = 0$) and that no photon will be found in the data of the Pierre Auger Observatory ($n = 0$) leads to $N^{0.95}_{\gamma,\text{cand}}(E_\gamma > E_0) = 3.095$. The effective exposure for methods that only use SD data (SD-Net, AP-Net) was estimated to be larger by a factor $\frac{1}{0.13} \approx 7.69$, because the FD has 13% the duty cycle of the SD. This assumption might differ from the true SD exposure, because more complex relations like energy dependencies or different angle acceptances are not taken into account.

The final upper limits for the considered methods, compared to other experiments and model predictions, are shown in figure 8.2. The BDT-Net and H-Net benefit from a higher signal efficiency and lower the limits to 70% and 60% respectively of the BDT limit. The SD-Net has not only an increased signal efficiency but also an increased exposure, which reduces the limit by one order of magnitude compared to the BDT limit. More information inside the hybrid data thus cannot compensate the lower exposure, which is why methods that only use SD data set the lowest upper limits. Therefore, the AP-Net reduces the limits the most, down to 7% of the BDT limit, which is roughly $0.2\,\text{km}^{-2}\text{yr}^{-1}\text{sr}^{-1}$. However, even the AP-Net is not sufficiently sensitive to get close to the theoretical predictions. Since the signal efficiency can hardly be increased any further, the only way to achieve these predictions is to significantly increase the exposure.

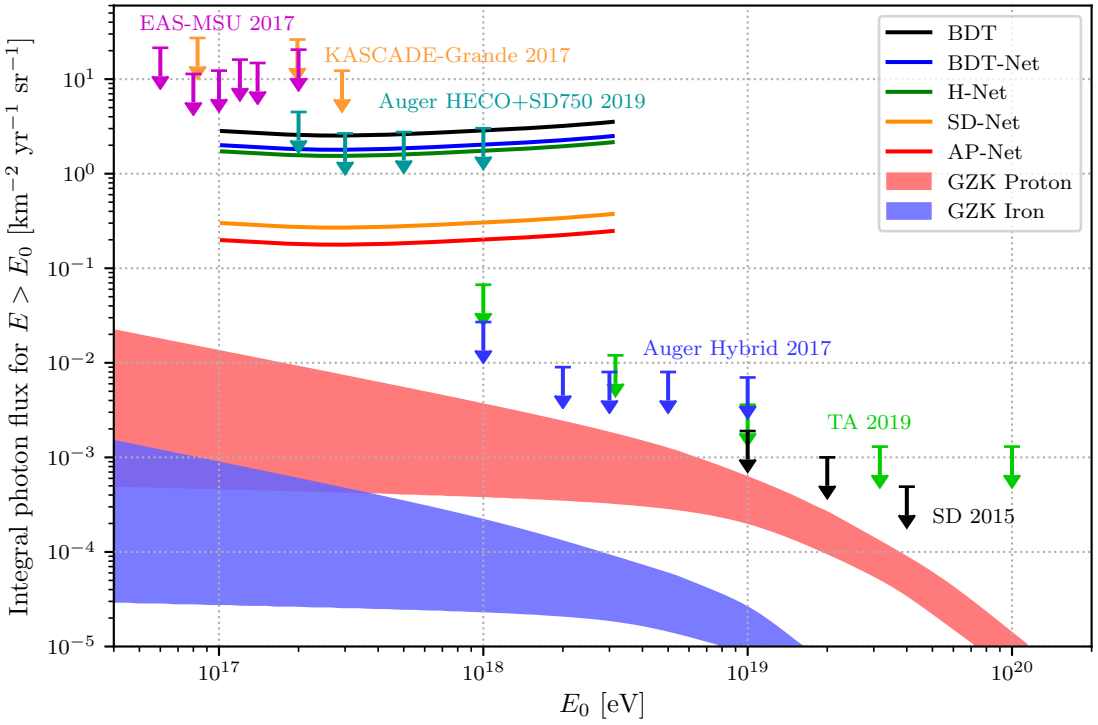


FIGURE 8.2: Upper limits on the integral photon flux $\Phi^{0.95}_{\text{U.L.}}(E_\gamma > E_0)$ with different methods and experiments [6, 50, 51, 52] at 95% confidence level (90% for EAS-MSU [53] and KASCADE-Grande [54]), compared to model predictions [55].

Chapter 9

Conclusion

In this analysis, the first attempt was made to apply deep learning techniques to SD and FD data of the Pierre Auger Observatory in order to find ultra-high energy photons. First, a BDT method based on a previous analysis was applied as a comparison.

Deep neural networks which are able to distinguish between proton and photon-induced extensive air showers were developed. By training the deep neural networks on simulated and labeled data, several approaches using different inputs were discussed. An essential feature of the network architectures, that were inspired by the AixNet, are convolutional layers that extract local correlations from the signal traces, arrival times and total signals of each SD station and FD pixel.

Given only the same five variables from the BDT analysis, the BDT-Net achieves a higher accuracy than the BDT. The H-Net, a very large network with all SD, FD (Coihueco and HEAT) and BDT variables as input, has approximately the same performance as the BDT-Net, despite the fact that it has much more information as input. Thus, it turned out that the improvement of the BDT-Net by use of more input data is challenging. The huge size of the input and the number of parameters of the H-Net extend the training duration to several days and complicate the development and fine-tuning of this network architecture. Furthermore, the FD-Net shows a bad performance, which might indicate that the DNNs have problems to extract important features from the FD traces. Further research should therefore be carried out on how this input can be used effectively.

On the other hand, the SD-Net shows very good results, considering that it gets only SD data. In addition, the developed AP-Net shows the potential of the ongoing AugerPrime upgrade. Using the number of particles of the muonic and electromagnetic component, this network increases the separation power between photons and protons significantly.

However, this is a best case scenario, because the AP-Net has Monte Carlo data as input. AugerPrime will nevertheless provide a major improvement in UHEP search.

Additionally, all methods were also tested on other hadronic interaction models and their energy dependence was analyzed. The DNNs show good stability against changes in the simulation and primary energy. Before applying the methods to measured data, further investigations are necessary regarding the mismatch between data and simulations.

However, a first estimation of the upper limits on the photon flux shows that the developed methods using DNNs are indeed able to lower the limits of current methods. The approaches that only use SD data (SD-Net and AP-Net) reach the lowest upper limits, because the gain in exposure, due to the higher duty cycle of the SD compared to the FD, provides a greater improvement than the higher signal efficiencies of the hybrid approaches. Therefore, currently the best approach in order to get the lowest upper limits on the photon flux is the SD-Net, an SD-only method using deep convolutional neural networks. The concept of the SD-Net could also be applied to the whole SD array and not only to the infill array. The benefit would be a much larger exposure, and this might lead to improvements of the upper limits, especially at higher energies. The AugerPrime upgrade with the resulting improvement of the discrimination between muonic and electromagnetic component will further lower the limit of the SD-Net. Nonetheless, a higher expose is still required, so that the photon upper limits, reached by the Pierre Auger Observatory, can clarify the predicted photon flux from the GZK effect.

Appendix A

Implementing deep neural networks

A.1 Boosted decision tree network (BDT-Net)

```
from tensorflow import keras as K
from tensorflow.keras import regularizers
from tensorflow.keras.layers import *

BDT_input = Input(shape=[5])
x = Dense(15, activation="selu")(BDT_input)
x = Dense(15, activation="selu")(x)
x = Dense(15, activation="selu")(x)
x = Dense(15, activation="selu")(x)
x = Dense(15, activation="selu")(x)
x = Dense(15, activation="selu")(x)
output = Dense(1, activation="sigmoid")(x)

model = K.models.Model([BDT_input],[output])

print(model.summary())

model.compile(loss="binary_crossentropy",
              optimizer=K.optimizers.Adam(lr=1E-3),
              metrics=["accuracy"])

batchsize = 10

model.fit_generator(generator(batchsize = batchsize),
                    steps_per_epoch=95000/batchsize,
                    epochs=500,
                    class_weight={0: 0.5, 1: 1},
```

```
28                      verbose=2,
29                      validation_data=val_data,
30                      max_queue_size=10,
31                      workers=1,
32                      use_multiprocessing=True,
33                      callbacks = [
34                      K.callbacks.CSVLogger("history.csv"),
35                      K.callbacks.ReduceLROnPlateau(monitor="val_acc",
36                                                    min_delta=0.001,
37                                                    patience= 30,
38                                                    verbose=1,
39                                                    mode="auto",
40                                                    factor=0.5,
41                                                    cooldown=5,
42                                                    min_lr=0),
43                      K.callbacks.EarlyStopping(    monitor="val_acc",
44                                                    min_delta=0.001,
45                                                    patience=500,
46                                                    verbose=1,
47                                                    mode="auto")])
48
49  model.save("model.h5")
```

A.2 Surface detector network (SD-Net)

```
from tensorflow import keras as K
from tensorflow.keras import regularizers
from tensorflow.keras.layers import *

SD_input_1 = Input(shape=[13,13,100,1])
SD_input_2 = Input(shape=[13,13,2])

l12 = 0.0003
kwargs1 = dict(activation="selu", kernel_initializer="he_normal")
kwargs2 = dict(activation="selu",
               kernel_regularizer = regularizers.l1_l2(l1=l12, l2=l12),
               kernel_size=(3,3),
               strides=(1, 1),
               padding="same",
               depth_multiplier=1,
               depthwise_initializer="he_normal",
               pointwise_initializer="he_normal")
kwargs3 = dict(activation="selu",
               kernel_regularizer = regularizers.l1_l2(l1=l12, l2=l12))

x = Conv3D(30, kernel_size=(1,1, 5), strides=(1,1,2),
           padding =  "same", **kwargs1)(SD_input_1)
x = Conv3D(50, kernel_size=(1,1, 5), strides=(1,1,2),
           padding =  "same", **kwargs1)(x)
x = Conv3D(70, kernel_size=(1,1, 5), strides=(1,1,2),
           padding =  "same", **kwargs1)(x)
x = Conv3D(30, kernel_size=(1,1,13), strides=(1,1,1),
           padding = "valid", **kwargs1)(x)
x = Reshape((13,13,30))(x)
x = concatenate([x,SD_input_2], axis=-1)
x = SeparableConv2D(32, **kwargs2)(x)
xl = [x]
for i in range(3):
    x = SeparableConv2D(2**(i+5), **kwargs2)(x)
    xl.append(x)
    x = concatenate(xl[:], axis=-1)
x = GlobalMaxPooling2D()(x)
x = Dropout(0.5)(x)
x = Dense(100, **kwargs3)(x)
x = Dropout(0.5)(x)
x = Dense(50, **kwargs3)(x)
x = Dropout(0.5)(x)
x = Dense(20, **kwargs3)(x)
x = Dropout(0.5)(x)

output = Dense(1, activation="sigmoid")(x)
```

```
47
48 model = K.models.Model([SD_input_1,SD_input_2],[output])
49
50 print(model.summary())
51
52 model.compile(loss="binary_crossentropy",
53               optimizer=K.optimizers.Adam(lr=1E-3),
54               metrics=["accuracy"])
55
56 batchsize = 10
57
58 model.fit_generator(generator(batchsize = batchsize),
59                     steps_per_epoch=95000/batchsize,
60                     epochs=300,
61                     class_weight={0: 0.5, 1: 1},
62                     verbose=2,
63                     validation_data=val_data,
64                     max_queue_size=10,
65                     workers=1,
66                     use_multiprocessing=True,
67                     callbacks = [
68                     K.callbacks.CSVLogger("history.csv"),
69                     K.callbacks.ReduceLROnPlateau(monitor="val_acc",
70                                                   min_delta=0.001,
71                                                   patience=30,
72                                                   verbose=1,
73                                                   mode="auto",
74                                                   factor=0.5,
75                                                   cooldown=5,
76                                                   min_lr=0),
77                     K.callbacks.EarlyStopping(    monitor="val_acc",
78                                                   min_delta=0.001,
79                                                   patience=80,
80                                                   verbose=1,
81                                                   mode="auto")])
82
83 model.save("model.h5")
```

A.3 Fluorescence detector network (FD-Net)

```
from tensorflow import keras as K
from tensorflow.keras import regularizers
from tensorflow.keras.layers import *

CO_input_1 = Input(shape=[22,80,50,1])
CO_input_2 = Input(shape=[22,80,2])

kwargs = dict(activation="relu", kernel_initializer="he_normal")

l12 = 0.00005
kwargs1 = dict(activation="selu", kernel_initializer="he_normal")
kwargs2 = dict(activation="selu",
               kernel_regularizer = regularizers.l1_l2(l1=l12, l2=l12),
               kernel_size=(3,3),
               strides=(1, 1),
               padding="same",
               depth_multiplier=1,
               depthwise_initializer="he_normal",
               pointwise_initializer="he_normal")
kwargs3 = dict(activation="selu",
               kernel_regularizer = regularizers.l1_l2(l1=l12, l2=l12))

y = Conv3D(30,  kernel_size=(1,1,5), strides=(1,1,2),
           padding =  "same", **kwargs1)(CO_input_1)
y = Conv3D(60,  kernel_size=(1,1,5), strides=(1,1,2),
           padding =  "same", **kwargs1)(y)
y = Conv3D(120, kernel_size=(1,1,5), strides=(1,1,2),
           padding =  "same", **kwargs1)(y)
y = Conv3D(30,  kernel_size=(1,1,7), strides=(1,1,1),
           padding = "valid", **kwargs1)(y)
y = Reshape((22,80,30))(y)
y = concatenate([y,CO_input_2], axis=-1)
y = SeparableConv2D(32, **kwargs2)(y)
yl = [y]
for i in range(4):
    y = SeparableConv2D(2**(i+5), **kwargs2)(y)
    yl.append(y)
    y = concatenate(yl[:], axis=-1)
y = GlobalMaxPooling2D()(y)
y = Dropout(0.4)(y)
y = Dense(100, **kwargs3)(y)

HE_input_1 = Input(shape=[22,40,50,1])
HE_input_2 = Input(shape=[22,40,2])

z = Conv3D(30,  kernel_size=(1,1,5), strides=(1,1,2),
```

```
47            padding =  "same", **kwargs1)(HE_input_1)
48 z = Conv3D(60,  kernel_size=(1,1,5), strides=(1,1,2),
49            padding =  "same", **kwargs1)(z)
50 z = Conv3D(120, kernel_size=(1,1,5), strides=(1,1,2),
51            padding =  "same", **kwargs1)(z)
52 z = Conv3D(30,  kernel_size=(1,1,7), strides=(1,1,1),
53            padding = "valid", **kwargs1)(z)
54 z = Reshape((22,40,30))(z)
55 z = concatenate([z,HE_input_2], axis=-1)
56 z = SeparableConv2D(32, **kwargs2)(z)
57 zl = [z]
58 for i in range(4):
59     z = SeparableConv2D(2**(i+5), **kwargs2)(z)
60     zl.append(z)
61     z = concatenate(zl[:], axis=-1)
62 z = GlobalMaxPooling2D()(z)
63 z = Dropout(0.4)(z)
64 z = Dense(100, **kwargs3)(z)
65
66 y = concatenate([y,z], axis=-1)
67 y = Dense(100, **kwargs3)(y)
68 y = Dropout(0.4)(y)
69 y = Dense(50, **kwargs3)(y)
70 y = Dropout(0.4)(y)
71 y = Dense(50, **kwargs3)(y)
72 y = Dropout(0.4)(y)
73 output = Dense(1, activation="sigmoid")(y)
74
75 model = K.models.Model([CO_input_1,CO_input_2,
76                         HE_input_1,HE_input_2],[output])
77
78 print(model.summary())
79
80 model.compile(loss="binary_crossentropy",
81               optimizer=K.optimizers.Adam(lr=1E-3),
82               metrics=["accuracy"])
83
84 batchsize = 10
85
86 model.fit_generator(generator(batchsize = batchsize),
87                     steps_per_epoch=95000/batchsize,
88                     epochs=300,
89                     class_weight={0: 0.5, 1: 1},
90                     verbose=2,
91                     validation_data=val_data,
92                     max_queue_size=10,
93                     workers=1,
94                     use_multiprocessing=True,
```

```
95                      callbacks = [
96                      K.callbacks.CSVLogger("history.csv"),
97                      K.callbacks.ReduceLROnPlateau(monitor="val_acc",
98                                                    min_delta=0.001,
99                                                    patience= 30,
100                                                   verbose=1,
101                                                   mode="auto",
102                                                   factor=0.5,
103                                                   cooldown=5,
104                                                   min_lr=0),
105                     K.callbacks.EarlyStopping(    monitor="val_acc",
106                                                   min_delta=0.001,
107                                                   patience=100,
108                                                   verbose=1,
109                                                   mode="auto")])
110
111 model.save("model.h5")
```

A.4 Hybrid network (H-Net)

```
from tensorflow import keras as K
from tensorflow.keras import regularizers
from tensorflow.keras.layers import *

SD_input_1 = Input(shape=[13,13,100,1])
SD_input_2 = Input(shape=[13,13,2])

l12 = 0.0003
kwargs1 = dict(activation="selu", kernel_initializer="he_normal")
kwargs2 = dict(activation="selu",
               kernel_regularizer = regularizers.l1_l2(l1=l12, l2=l12),
               kernel_size=(3,3),
               strides=(1, 1),
               padding="same",
               depth_multiplier=1,
               depthwise_initializer="he_normal",
               pointwise_initializer="he_normal")
kwargs3 = dict(activation="selu",
               kernel_regularizer = regularizers.l1_l2(l1=l12, l2=l12))

sd = Conv3D(30, kernel_size=(1,1, 5), strides=(1,1,2),
            padding =  "same", **kwargs1)(SD_input_1)
sd = Conv3D(50, kernel_size=(1,1, 5), strides=(1,1,2),
            padding =  "same", **kwargs1)(sd)
sd = Conv3D(70, kernel_size=(1,1, 5), strides=(1,1,2),
            padding =  "same", **kwargs1)(sd)
sd = Conv3D(30, kernel_size=(1,1,13), strides=(1,1,1),
            padding = "valid", **kwargs1)(sd)
sd = Reshape((13,13,30))(sd)
sd = concatenate([sd,SD_input_2], axis=-1)
sd = SeparableConv2D(32, **kwargs2)(sd)
sdl = [sd]
for i in range(3):
    sd = SeparableConv2D(2**(i+5), **kwargs2)(sd)
    sdl.append(sd)
    sd = concatenate(sdl[:], axis=-1)
sd = GlobalMaxPooling2D()(sd)
sd = Dense(100, **kwargs3)(sd)
sd = Dropout(0.3)(sd)
sd = Dense(50, **kwargs3)(sd)
sd = Dropout(0.3)(sd)
sd = Dense(20, **kwargs3)(sd)
sd = Dropout(0.3)(sd)
sd = Dense(10, **kwargs3)(sd)

CO_input_1 = Input(shape=[22,80,50,1])
```

```
47 CO_input_2 = Input(shape=[22,80,2])
48
49 co = Conv3D(30,  kernel_size=(1,1,5), strides=(1,1,2),
50             padding =  "same", **kwargs1)(CO_input_1)
51 co = Conv3D(60,  kernel_size=(1,1,5), strides=(1,1,2),
52             padding =  "same", **kwargs1)(co)
53 co = Conv3D(120, kernel_size=(1,1,5), strides=(1,1,2),
54             padding =  "same", **kwargs1)(co)
55 co = Conv3D(30,  kernel_size=(1,1,7), strides=(1,1,1),
56             padding = "valid", **kwargs1)(co)
57 co = Reshape((22,80,30))(co)
58 co = concatenate([co,CO_input_2], axis=-1)
59 co = SeparableConv2D(32, **kwargs2)(co)
60 col = [co]
61 for i in range(4):
62     co = SeparableConv2D(2**(i+5), **kwargs2)(co)
63     col.append(co)
64     co = concatenate(col[:], axis=-1)
65 co = GlobalMaxPooling2D()(co)
66 co = Dense(100, **kwargs3)(co)
67 co = Dropout(0.3)(co)
68 co = Dense(50, **kwargs3)(co)
69 co = Dropout(0.3)(co)
70 co = Dense(20, **kwargs3)(co)
71 co = Dropout(0.3)(co)
72 co = Dense(10, **kwargs3)(co)
73
74 HE_input_1 = Input(shape=[22,40,50,1])
75 HE_input_2 = Input(shape=[22,40,2])
76
77 he = Conv3D(30,  kernel_size=(1,1,5), strides=(1,1,2),
78             padding =  "same", **kwargs1)(HE_input_1)
79 he = Conv3D(60,  kernel_size=(1,1,5), strides=(1,1,2),
80             padding =  "same", **kwargs1)(he)
81 he = Conv3D(120, kernel_size=(1,1,5), strides=(1,1,2),
82             padding =  "same", **kwargs1)(he)
83 he = Conv3D(30,  kernel_size=(1,1,7), strides=(1,1,1),
84             padding = "valid", **kwargs1)(he)
85 he = Reshape((22,40,30))(he)
86 he = concatenate([he,HE_input_2], axis=-1)
87 he = SeparableConv2D(32, **kwargs2)(he)
88 hel = [he]
89 for i in range(4):
90     he = SeparableConv2D(2**(i+5), **kwargs2)(he)
91     hel.append(he)
92     he = concatenate(hel[:], axis=-1)
93 he = GlobalMaxPooling2D()(he)
94 he = Dense(100, **kwargs3)(he)
```

```
95  he = Dropout(0.3)(he)
96  he = Dense(50, **kwargs3)(he)
97  he = Dropout(0.3)(he)
98  he = Dense(20, **kwargs3)(he)
99  he = Dropout(0.3)(he)
100 he = Dense(10, **kwargs3)(he)
101
102 BDT_input = Input(shape=[5])
103
104 bdt = Dense(15, activation="selu")(BDT_input)
105 bdt = Dense(15, activation="selu")(bdt)
106 bdt = Dense(15, activation="selu")(bdt)
107 bdt = Dense(15, activation="selu")(bdt)
108 bdt = Dense(15, activation="selu")(bdt)
109 bdt = Dense(15, activation="selu")(bdt)
110
111 x = concatenate([sd,co,he,bdt], axis=-1)
112
113 x = Dense(40, **kwargs3)(x)
114 x = Dense(30, **kwargs3)(x)
115 x = Dense(20, **kwargs3)(x)
116 x = Dense(15, **kwargs3)(x)
117 x = Dense(15, **kwargs3)(x)
118 x = Dense(15, **kwargs3)(x)
119
120 output = Dense(1, activation="sigmoid")(x)
121
122 model = K.models.Model([SD_input_1,SD_input_2,
123                         CO_input_1,CO_input_2,
124                         HE_input_1,HE_input_2,
125                         BDT_input],[output])
126
127 print(model.summary())
128
129 model.compile(loss="binary_crossentropy",
130               optimizer=K.optimizers.Adam(lr=1E-3),
131               metrics=["accuracy"])
132
133 batchsize = 10
134
135 model.fit_generator(generator(batchsize = batchsize),
136                     steps_per_epoch=95000//batchsize,
137                     epochs=200,
138                     class_weight={0: 0.5, 1: 1},
139                     verbose=2,
140                     validation_data=val_data,
141                     max_queue_size=10,
142                     workers=1,
```

```
143                     use_multiprocessing=True,
144                     callbacks = [
145                     K.callbacks.CSVLogger("history.csv"),
146                     K.callbacks.ReduceLROnPlateau(monitor="val_acc",
147                                                   min_delta=0.001,
148                                                   patience= 30,
149                                                   verbose=1,
150                                                   mode="auto",
151                                                   factor=0.5,
152                                                   cooldown=5,
153                                                   min_lr=0),
154                     K.callbacks.EarlyStopping(    monitor="val_acc",
155                                                   min_delta=0.001,
156                                                   patience=100,
157                                                   verbose=1,
158                                                   mode="auto")])
159
160 model.save("model.h5")
```

A.5 AugerPrime network (AP-Net)

```
from tensorflow import keras as K
from tensorflow.keras import regularizers
from tensorflow.keras.layers import *

input = Input(shape=[13,13,3,1])

kwargs = dict(activation="selu", strides=(1,1,1), padding = "valid",
    kernel_initializer="he_normal")

x = Conv3D(20, kernel_size=(3,3,1), **kwargs)(input)
x = Conv3D(40, kernel_size=(3,3,1), **kwargs)(x)
x = Conv3D(60, kernel_size=(3,3,1), **kwargs)(x)
x = Conv3D(20, kernel_size=(3,3,3), **kwargs)(x)
x = GlobalMaxPooling3D()(x)
x = Dense(20, activation="selu")(x)
x = Dropout(0.3)(x)
x = Dense(20, activation="selu")(x)
x = Dropout(0.3)(x)
x = Dense(10, activation="selu")(x)

output = Dense(1, activation="sigmoid")(x)

model = K.models.Model([input],[output])

print(model.summary())

model.compile(loss="binary_crossentropy",
              optimizer=K.optimizers.Adam(lr=1E-3),
              metrics=["accuracy"])

batchsize = 10

model.fit_generator(generator(batchsize = batchsize),
                    steps_per_epoch=95000//batchsize,
                    epochs=300,
                    class_weight={0: 0.5, 1: 1},
                    verbose=2,
                    validation_data=val_data,
                    max_queue_size=10,
                    workers=1,
                    use_multiprocessing=True,
                    callbacks = [
                    K.callbacks.CSVLogger("history.csv"),
                    K.callbacks.ReduceLROnPlateau(monitor="val_acc",
                                                  min_delta=0.001,
                                                  patience=30,
```

```
46                                                  verbose=1,
47                                                  mode="auto",
48                                                  factor=0.5,
49                                                  cooldown=5,
50                                                  min_lr=0),
51                      K.callbacks.EarlyStopping(  monitor="val_acc",
52                                                  min_delta=0.001,
53                                                  patience=80,
54                                                  verbose=1,
55                                                  mode="auto")])
56
57  model.save("model.h5")
```

Bibliography

[1] D. J. Bird et al., *Detection of a Cosmic Ray with Measured Energy Well Beyond the Expected Spectral Cutoff due to Cosmic Microwave Radiation* (1994), arXiv:astro-ph/9410067

[2] The Tibet ASγ Collaboration, *First Detection of Photons with Energy Beyond* 100 TeV *from an Astrophysical Source* (2019), arXiv:1906.05521v1

[3] The Pierre Auger Collaboration, *The Pierre Auger Cosmic Ray Observatory* (2015), arXiv:1502.01323v5

[4] O. Russakovsky et al., *ImageNet Large Scale Visual Recognition Challenge IJCV* (2015), arXiv:1409.0575

[5] M. Erdmann, J. Glombitza, D. Walz, *A Deep Learning-based Reconstruction of Cosmic Ray-induced Air Showers* (2017), arXiv:1708.00647v2

[6] J. Rautenberg for the Pierre Auger Collaboration, *Limits on ultra-high energy photons with the Pierre Auger Observatory* (2019), PoS(ICRC2019)398

[7] S. Eickhoff, M. Niechciol, M. Risse, *Extending the search for primary photons with the hybrid detector to energies below* 1 EeV (2019), GAP 2019-010 (internal note)

[8] J. R. Hörandel et al., *Dissecting the knee - Air shower measurements with KASCADE* (2003), arXiv:astro-ph/0311478

[9] The Pierre Auger Collaboration, *Measurement of the energy spectrum of cosmic rays above* 10^{18} eV *using the Pierre Auger Observatory* (2010), arXiv:1002.1975

[10] K. Greisen, *End to the Cosmic-Ray Spectrum?* (1966), Physical Review Letters, Vol. 16, Issue 17, pp. 748-750, DOI: 10.1103/PhysRevLett.16.748

[11] G. T. Zatsepin, V. A. Kuzmin, *Upper Limit of the Spectrum of Cosmic Rays* (1966), Journal of Experimental and Theoretical Physics Letters, Vol. 4, p. 78, bibcode: 1966JETPL...4...78Z

[12] E. Fermi, *On the Origin of the Cosmic Radiation* (1949), Physical Review 75, 1169

[13] W. Hanlon, *Updated cosmic ray spectrum*, `http://www.physics.utah.edu/~whanlon/spectrum.html` (November 12th 2019)

[14] G. B. Gelmini, *GZK Photons as Ultra High Energy Cosmic Rays* (2007), arXiv:astro-ph/0506128v3

[15] O. C. Allkofer, *Introduction to cosmic radiation* (1975), Verlag Karl Thiemig München

[16] M. Tanabashi et al. (Particle Data Group), *Review of Particle Physics* (2018), Physical Review D, Vol. 98, Issue 3, ID 030001, DOI: 10.1103/PhysRevD.98.030001

[17] W. Heitler, *The Quantum Theory of Radiation, Dover Books on Physics and Chemistry* (1954), Dover Publications

[18] T. K. Gaisser, A. M. Hillas, *Reliability of the method of constant intensity cuts for reconstructing the average development of vertical showers* (1977), Proceedings of the 15th ICRC, 8:353-357

[19] T. Pierog et al., *Dependence of the longitudinal shower profile on the characteristics of hadronic multiparticle production* (2005), Proceedings of the 29th ICRC, 7:103-106

[20] L. Landau, I. Pomeranchuk, *Limits of applicability of the theory of bremsstrahlung electrons and pair production at high-energies* (1953), Dokl. Akad. Nauk Ser. Fiz., Vol. 92, pp. 535-536

[21] A. Migdal, *Bremsstrahlung and Pair Production in Condensed Media at High Energies* (1956), Physical Review, Vol. 103, Issue 6, pp. 1811-1820, DOI: 10.1103/PhysRev.103.1811

[22] A. N. Cillis et al., *Influence of the LPM effect and dielectric suppression on particle air showers* (1999), arXiv:astro-ph/9809334

[23] X. Bertou et al., *LPM effect and pair production in the geomagnetic field: a signature of ultra-high energy photons in the Pierre Auger Observatory* (2000), Astroparticle Physics, Vol. 14, Issue 2, p. 121-130

[24] F. Schmidt, J. Knapp, *CORSIKA Shower Images*, `https://www-zeuthen.desy.de/~jknapp/fs/showerimages.html` (November 25th 2019)

[25] F. G. Schröder, *Radio detection of Cosmic-Ray Air Showers and High-Energy Neutrinos* (2017), arXiv:1607.08781

[26] I. Allekotte et al., *The Surface Detector System of the Pierre Auger Observatory* (2007), arXiv:0712.2832v1

[27] D. Gora for the Pierre Auger Collaboration, *The Pierre Auger Observatory: review of latest results and perspectives* (2018), arXiv:1811.00343v2

[28] B. G. Keilhauer, *Investigation of Atmospheric Effects on the Development of Extensive Air Showers and their Detection with Pierre Auger Observatory* (2004), Forschungszentrum Karlsruhe Report FZKA 6958

[29] The Pierre Auger Collaboration, *The Fluorescence Detector of the Pierre Auger Observatory* (2009), arXiv:0907.4282v1

[30] The Pierre Auger Collaboration, *The Pierre Auger Observatory Upgrade "AugerPrime" Preliminary Design Report* (2016), arXiv:1604.03637v1

[31] A. Castellina for the Pierre Auger Collaboration, *AugerPrime: the Pierre Auger Observatory Upgrade* (2019), arXiv:1905.04472v1

[32] Miguel Mostafá for the Pierre Auger Collaboration, *The Hybrid Activities of the Pierre Auger Observatory* (2006), arXiv:astro-ph/0608670

[33] D. Heck et al., *CORSIKA: A Monte Carlo code to simulate extensive air showers* (1998), Forschungszentrum Karlsruhe Report FZKA 6019

[34] A. Haungs, *Air Shower Measurements in Karlsruhe* (2007), arXiv:0705.0202v1

[35] S. Argirò et al., *The Offline Software Framework of the Pierre Auger Observatory* (2007), arXiv:0707.1652v1

[36] R. R. Prado, *Tests of hadronic interactions with measurements by Pierre Auger Observatory* (2018), arXiv:1810.00586

[37] A. Ferrari, P. R. Sala, A. Fassò, J. Ranft, *FLUKA: a multi-particle transport code*, CERN-2005-10

[38] E. dos Santos, A. Yushkov, *Extending the Naples CORSIKA shower library for Auger studies at* $16.5 \leq \log_{10}[E/\mathrm{eV}] \leq 18.0$ (2018), GAP 2018-043 (internal note)

[39] `https://www.python.org/`

[40] F. Pedregosa et al., *Scikit-learn: Machine Learning in Python* (2018), arXiv:1201.0490v4

[41] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, *Self-Normalizing Neural Networks* (2017), arXiv:1706.02515v5

[42] F. Chollet, *Xception: Deep Learning with Depthwise Separable Convolutions* (2017), arXiv:1610.02357v3

[43] G. Huang, Z. Liu, K. Q. Weinberger, L. v. d. Maaten, *Densely Connected Convolutional Networks* (2018), arXiv:1608.06993v5

[44] K. He, X. Zhang, S. Ren, J. Sun, *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification* (2015), arXiv:1502.01852

[45] F. Chollet et al., *Keras* (2015), `https://keras.io/`

[46] M. Abadi et al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (2015), `https://www.tensorflow.org/`

[47] D. P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization* (2017), arXiv:1412.6980v9

[48] M. Niechciol, M. Risse, *Preliminary results of the hybrid search for photons below* 1 EeV (2019), Pierre Auger Collaboration Analysis Meeting 2019 (Nijmegen)

[49] G. Feldman, R. Cousins, *A Unified Approach to the Classical Statistical Analysis of Small Signals* (1999), arXiv:physics/9711021v2

[50] The Pierre Auger Collaboration, *Search for photons with energies above* 10^{18} eV *using the hybrid detector of the Pierre Auger Observatory* (2017), arXiv:1612.01517v2

[51] C. Bleve for the Pierre Auger Collaboration, *Update of the neutrino and photon limits from the Pierre Auger Observatory* (2015), PoS(ICRC2015)1103

[52] R. U. Abbasi et al., *Constraints on the diffuse photon flux with energies above* 10^{18} eV *using the surface detector of the Telescope Array experiment* (2019), arXiv:1811.03920v2

[53] Y. A. Fomin et al., *Constraints on the flux of* $\sim (10^{16} - 10^{17.5})$ eV *cosmic photons from the EAS-MSU muon data* (2017), arXiv:1702.08024v1

[54] KASCADE-Grande Collaboration, *KASCADE-Grande limits on the isotropic diffuse gamma-ray flux between* 100 TeV *and* 1 EeV (2017), arXiv:1710.02889v1

[55] K. Kampert et al., *Ultra-High Energy Photon and Neutrino Fluxes in Realistic Astrophysical Scenarios* (2011), DOI: 10.7529/ICRC2011/V02/1087

Acknowledgements

First of all I would like to thank Prof. Dr. Bretz and Prof. Dr. Hebbeker for having made it possible for me to work on this interesting topic and to learn about current methods of deep learning and machine learning in general. I am grateful for the opportunity granted to me by them to attend the Frühjahrstagung 2019 of the Deutsche Physikalische Gesellschaft in Aachen and the Auger Analysis Meeting 2019 in Nijmegen.

I would like to express my special thanks to Paulo Ferreira, Julian Kemp and Adrianna García for their competent support. They have been a great help to me and have always answered my questions patiently. I would also like to thank them for their repeated proof reading.

I also would like to especially thank Jonas Glombitza for all his support concerning my many questions about deep learning and working on the VISPA cluster.

My thanks also go to the entire Auger working group, as well as Jan Audehm, Fabian Theißen, Marc Klinger, Marvin Beck and Christine Peters who also gave important impulses for my work in the weekly meetings and ensured a generally pleasant working atmosphere.

Last but not least, I would like to thank my whole family for their moral support. I especially thank my parents and my grandmother who have also agreed to proof read this thesis.