Ultra-Fast Generation of Imaging Atmospheric Cherenkov Telescope Camera Images using Generative Adversarial Networks

Master's Thesis in Physics

Presented by

Christian Elflein

16.10.2023

Erlangen Centre for Astroparticle Physics

Friedrich-Alexander-Universität Erlangen-Nürnberg

Supervisor: Prof. Dr. Stefan Funk

Disclaimer

Parts of this thesis are prepared for submission to JINST and are currently under H.E.S.S. Courtesy review. The relevant passages in this work are marked accordingly with [1].

Abstract

Cosmic gamma rays are highly energetic photons produced close to astrophysical sources in the universe. When such particles reach the Earth, they induce particle cascades, which can be detected using ground-based instruments such as Imaging Atmospheric Cherenkov Telescopes (IACTs). By measuring the Cherenkov light emitted during the particle cascade, IACTs yield images that contain information about the primary particle and its kinematic properties. To fully understand the physics and the instruments used to detect gamma rays, large simulation libraries are required. However, Monte Carlo (MC) simulations that model the air shower development as well as the instrument response are computationally expensive. New approaches based on deep learning offer encouraging prospects for accelerating the simulation of IACT events.

In this thesis, the capabilities and efficiency of deep learning models for the augmentation of MC simulations are investigated. IACT camera images are generated by training a generative learning algorithm using simulated data of one of the current generation IACT arrays – the High Energy Stereoscopic System (H.E.S.S.). The framework for accelerating IACT simulations developed within this work, based on conditional Wasserstein Generative Adversarial Networks, is presented and explained in detail. In particular, modifications and learning strategies of the algorithm are discussed to enable a successful application to generate IACT images. Finally, we comprehensively examine the quality of the generated events and find promising prospects to accelerate IACT simulations by order of 10^5 without significant loss of the simulation quality.

Kurzfassung

Kosmische Gammastrahlen sind hochenergetische Photonen, die in der Nähe von astrophysikalischen Quellen im Universum erzeugt werden. Wenn solche Teilchen die Erde erreichen, lösen sie Teilchenkaskaden aus, die mit bodengestützten Instrumenten wie den Imaging Atmospheric Cherenkov Telescopes (IACTs) nachgewiesen werden können. Durch die Messung des Cherenkov-Lichts, das während der Teilchenkaskade ausgesendet wird, liefern IACTs Bilder, die Informationen über das Primärteilchen und seine kinematischen Eigenschaften enthalten. Um die Physik und die zum Nachweis von Gammastrahlen verwendeten Instrumente vollständig zu verstehen, sind umfangreiche Simulationsbibliotheken erforderlich. Monte-Carlo-Simulationen (MC), die sowohl die Entwicklung des Luftschauers als auch die Reaktion des Instruments modellieren, sind jedoch sehr rechenintensiv. Neue Ansätze, die auf Deep Learning basieren, bieten vielversprechende Aussichten, die Simulation von IACT-Ereignissen zu beschleunigen.

In dieser Arbeit werden die Möglichkeiten und die Effizienz von Deep-Learning-Modellen für die Erweiterung von MC-Simulationen untersucht. IACT-Kamerabilder werden durch Training eines generativen Lernalgorithmus mit simulierten Daten eines IACT-Arrays der aktuellen Generation – dem High Energy Stereoscopic System (H.E.S.S.) – erzeugt. Das in dieser Arbeit entwickelte Framework zur Beschleunigung von IACT-Simulationen, das auf konditionalen Wasserstein Generative Adversarial Networks basiert, wird im Detail vorgestellt und erläutert. Insbesondere werden Modifikationen und Lernstrategien des Algorithmus diskutiert, um eine erfolgreiche Anwendung zur Erzeugung von IACT-Bildern zu ermöglichen. Abschließend wird die Qualität der generierten Ereignisse umfassend untersucht und es werden vielversprechende Möglichkeiten aufgezeigt, IACT-Simulationen um eine Größenordnung von 10^5 zu beschleunigen, ohne dass die Simulationsqualität signifikant beeinträchtigt wird.

Contents

1 Introduction

Every day, countless amounts of different particles are produced throughout the whole universe. One type of these particles is cosmic gamma rays [2], that are high-energy photons produced in various astrophysical sources. After their production, they propagate through the universe, and some of the gamma rays arrive at Earth. Due to their lack of charge, they do not get deflected by magnetic fields during their travels. Thus, by detecting gamma rays, astronomical observations can be conducted. Since the atmosphere is opaque to them, they cannot be detected directly. However, when gamma rays penetrate the atmosphere, they induce a cascade of secondary particles, which move down to the Earth's surface. These secondary particles emit Cherenkov light, which the Imaging Atmospheric Cherenkov Telescope (IACT) can detect. An example of a current generation IACT array is the High Energy Stereoscopic System (H.E.S.S.) [3]. The instrument, located in Namibia, is an array of five IACTs detecting gamma rays with energies in the GeV to TeV regime.

For a better understanding of the physics and the instrumentation, simulations are an essential tool. In the astroparticle physics community, two different simulation programs are commonly used: CORSIKA [4] for the simulation of the air shower, including the Cherenkov light, and `sim_telarray` [5] for the simulation of the instrument response. In the end, like in real measurements, IACT images are obtained, and examples for such are shown in Figure 1[1]. A challenge with Monte Carlo (MC) simulations is that they are computationally expensive and they can take a long time. A machine learning-based approach for the simulation of IACT events opens promising prospects to overcome this challenge and comes with several additional advantages. Over the last decade, generative machine learning models like Generative Adversarial Networks (GANs) [8] or diffusion models [9] have shown a lot of success [10, 11, 12, 13] in accelerating physics simulations.

In this thesis, a novel technique for the generation of IACT camera images using deep generative models is presented. To fully understand the physics part of this work, the theoretical basics of gamma-ray astronomy are explained in Section 2. This includes a brief overview of gamma rays, their sources and their indirect detection using ground-based instruments. Additionally, two different IACT arrays, H.E.S.S. and the Cherenkov Telescope Array (CTA), are discussed in detail, and the basics of event reconstruction are explained. An introduction to deep learning is given at the beginning of Section 3 to be able to understand the approach in this work fully. Furthermore, the concept of GANs used for the IACT image generation is presented. The theoretical aspects of MC simulations are briefly discussed in Section 4 followed by a look at the simulated data used for training the GAN and the preparation of the data for it to be suitable for machine learning training. Next, the framework for the IACT image generation

[1] All similar camera images in this work are created with the open-source software package ctapipe [6] (v0.19.0 [7]).

in this work is discussed Section 5. This includes an overview of the different networks and their designs but also the setup of the training. Besides that, the training strategy to ensure the generation of realistic-looking IACT images is discussed. The IACT images generated with the framework are then presented in Section 6 and compared in detail to simulated images. Subsequently, the image quality is examined on a visual and physical level, with various camera image parameters being investigated. Some of these image parameters are the Hillas parameters, which are usually examined in the standard analysis method of IACT images. Lastly, the work is summarised in Section 7, and the future of IACT event simulations is shortly discussed.

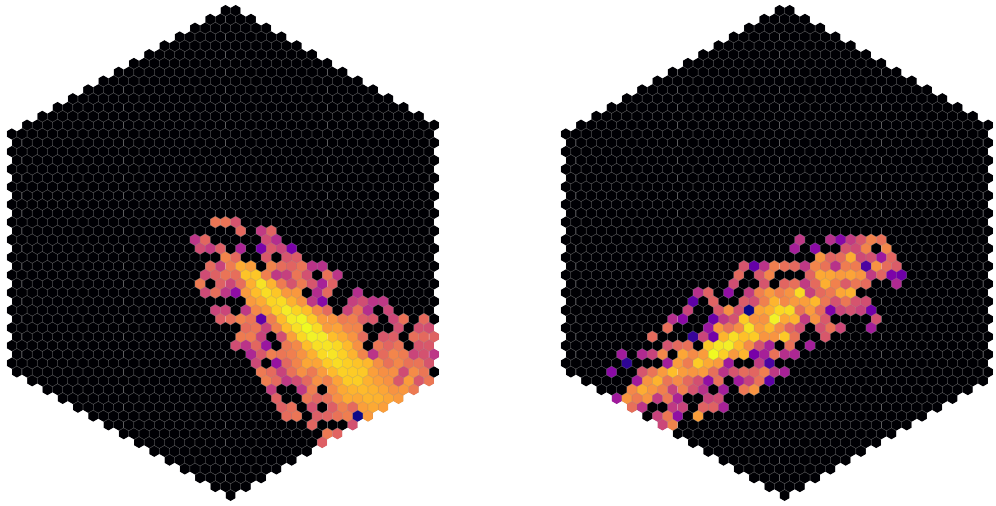


Figure 1: Two examples of IACT images from H.E.S.S. simulations. A brighter signal is equivalent to a higher detection of Cherenkov light.

2 Gamma-ray astronomy

Before the novel way to generate IACT images with machine learning is introduced, the related physical concepts are discussed. This includes a brief overview of gamma rays, their sources and production mechanism in Section 2.1. Extensive air showers in the Earth's atmosphere are discussed in Section 2.2. Within these showers, secondary particles emit Cherenkov light, which ground-based instruments like Imaging Atmospheric Cherenkov Telescopes (IACTs) can detect. This type of telescope and its detection principle, the imaging technique, is explained in Section 2.3. Lastly, an overview of even reconstruction is given in Section 2.4.

2.1 Gamma rays

Cosmic gamma rays are highly energetic photons from outer space which cover the majority of the electromagnetic spectrum [15]. Their energies start in the MeV regime and can exceed $10^{15}\,\mathrm{eV}$ [16] and their energy distribution closely follows in most cases a simple power law [2] of the form

$$\frac{\mathrm{d}N}{\mathrm{d}E} \propto E^{-\alpha} \tag{1}$$

with N and α as the particle number and the spectral index, which is usually between 2 and 3. There are various astrophysical sources in the universe that

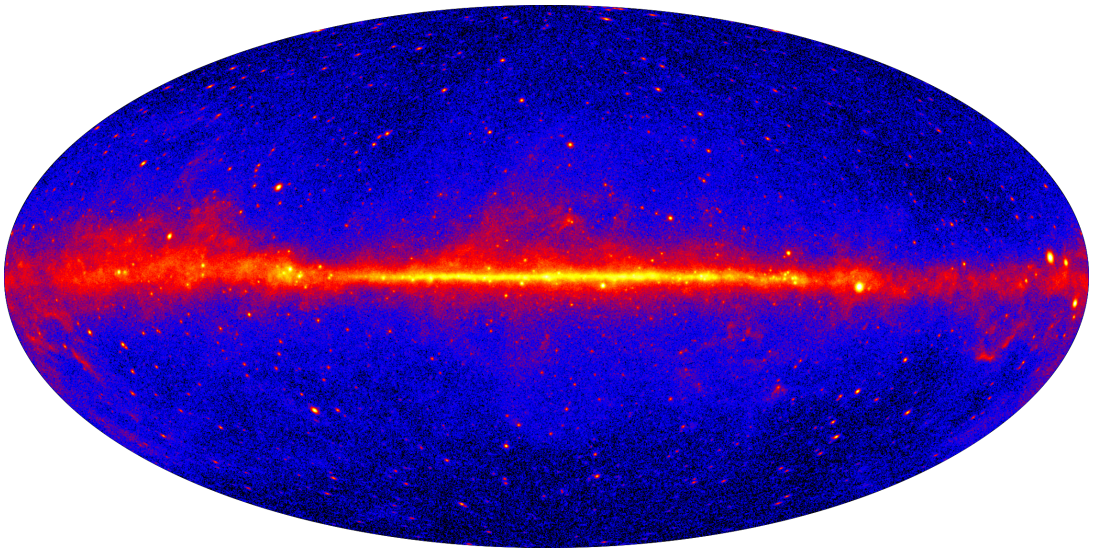


Figure 2: Image of the gamma-ray sources in the sky in the galactic coordinate system with source energies greater than 1 GeV. The data is based on five years of measurements with the Large Area Telescope (LAT) instrument of NASA's Fermi Gamma-Ray Space Telescope. The colour map illustrates the perceived emission strength of the different sources. Adapted from [14].

can emit gamma rays, with examples being supernovae, neutron stars or active galactic nuclei [17]. To visualize the vast number of gamma-ray sources in the sky, an image of detected sources with gamma-ray energies exceeding 1 GeV is shown in Figure 2. At the different sources, charged particles like protons or electrons are accelerated to very high energies, resulting in cosmic rays (CR) [18] that propagate through the universe. Further, secondary interactions of CRs result in the emission of gamma rays. Commonly, the production mechanism of gamma-rays is split into a leptonic and a hadronic case [19]. In the leptonic case, processes such as inverse Compton scattering, synchrotron radiation, and bremsstrahlung are considered. During the inverse Compton scattering

$$e^-_{\text{high energy}} + \gamma_{\text{low energy}} \longrightarrow e^-_{\text{low energy}} + \gamma_{\text{high energy}} \tag{2}$$

highly relativistic electrons scatter with low-energy photons and transfer a part of their kinetic energy to the photon. In a second scenario high-energy electrons that gyrate in a magnetic field emit synchrotron radiation due to the acceleration of the charged particle. Another way that gamma rays may be emitted is by bremsstrahlung:

$$e^{-/+}_{\text{before}} + Z \longrightarrow e^{-/+}_{\text{after}} + \gamma + Z \tag{3}$$

Here, a charged particle (usually an electron) gets accelerated in the electrostatic field of another nucleus.

Additionally, the hadronic case includes one process involving neutral pions, which are produced during high-energy particle interactions. These neutral pions

$$\pi^0 \longrightarrow \gamma + \gamma \tag{4}$$

promptly decays into two photons. After the production of the gamma rays in all these processes, they traverse the universe and can arrive at Earth. Since these particles are not charged like CRs, they do not get deflected by magnetic fields while traversing the universe. That makes it possible to reconstruct the path back to their source and learn about the highest energetic processes in the universe.

Due to the steeply falling power law distribution, increasingly large detection areas are required to maintain acceptable detection rates. Above tens of GeV, the needed detection areas become infeasible for satellites. Hence, particles with such energies have to be detected with ground-based instruments. However, since the atmosphere is opaque to gamma rays, the particles cannot be detected directly. Instead, they are detected indirectly through particle cascades induced by the primary photon.

2.2 Extensive air showers

Incident cosmic gamma rays interact with atoms and molecules in the earth's atmosphere and produce a cascade of secondary particles. On their way down, these particles interact again, resulting in even more secondary particles. Eventually, a cascade of particles called Extensive Air Shower (EAS) [2] is present in the

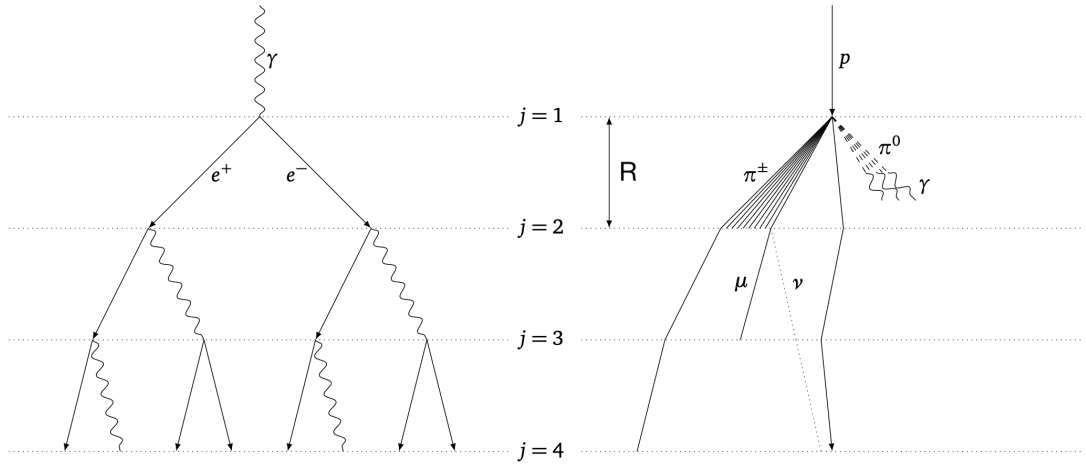


Figure 3: Simplified illustration of extensive air showers based on the Heitler model [20] and Heitler-Matthews model [21]. Left: Electromagnetic air shower induced by a primary photon. The photons decay into electrons and positrons through pair production. The electrons and positrons emit photons through bremsstrahlung. Right: Hadronic air shower induced by a primary proton. The proton decays into several hadrons, including neutral and charged pions. Adapted from [22].

atmosphere. The particles of this cascade can emit Cherenkov radiation that can be detected by ground-based instruments. There are two types of EAS – electromagnetic and hadronic – which are determined by the primary particle inducing the air shower. In Section 2.2.1 and Section 2.2.2, the electromagnetic and hadronic air shower is explained in detail and in Section 2.2.3, the physics of Cherenkov light.

2.2.1 Electromagnetic air shower

A simplified version of an electromagnetic air shower – the Heitler model [20] – caused by a primary gamma ray is shown on the left of Figure 3. The particle with energy E_0 interacts with the coulomb field of a nucleus Z, which changes to Z' due to energy and momentum transfer, and pair production

$$\gamma + Z \longrightarrow e^- + e^+ + Z' \tag{5}$$

takes place, resulting in an electron and positron, with both assumed to get roughly half of the energy of the primary photon. This process requires a photon energy of more than $2m_e c^2$. After a length $R = X_0 \cdot \ln 2$ with the radiation length X_0 in air being about $36.5\,\mathrm{g/cm^{-2}}$, both particles emit a photon through bremsstrahlung by interacting with an electrostatic field of a nucleus in the air. In this process, they lose half of their energy to the photons, resulting in four particles with the energy $\sim E_0/4$.

The new photons again undergo pair production while the electrons and positrons can emit bremsstrahlung. This happens after another length R because the radiation length of both processes is approximately the same. This production of new particles continues, resulting in the electromagnetic air shower. At some point in the shower development, the average particle energy drops to the critical energy E_c, and no new particles can be produced. This is when the maximum of the shower is reached with the highest number of particles $\sim E/E_c$ being present at the same time. The depth of the shower maximum is then defined by

$$X_{\max} = X_0 \cdot \frac{\ln(E_0/E_c)}{\ln 2}. \tag{6}$$

Afterwards, no new particles are produced due to the lack of energy and ionisation losses. Due to the structure of this shower type, roughly 2/3 of the particles are electrons and positrons, and 1/3 are photons [18].

2.2.2 Hadronic air shower

A hadronic air shower induced by a cosmic ray, represented by a proton, is illustrated as a simplified version on the right side of Figure 3. This type of EAS can be separated further into different types of sub-showers, namely hadronic, muonic and electromagnetic sub-showers. The primary proton interacts, and different types of hadrons are produced, and some of these can again produce new hadrons, resulting in a hadronic sub-shower. Within the Heitler-Matthews model [21], 1/3 of the particle energy is distributed at each generation on neutral pions and 2/3 on charged pions. Neutral pions generated during the hadronic interaction decay very quickly into two photons, with both particles able to start an electromagnetic sub-shower. This sub-shower results in the production of new electrons, positrons and photons, as explained above. Produced charged pions may decay following:

$$\pi^+ \longrightarrow \mu^+ + \nu_\mu \longrightarrow e^+ + \nu_e + \bar{\nu}_\mu + \nu_\mu \tag{7}$$

$$\pi^- \longrightarrow \mu^- + \bar{\nu}_\mu \longrightarrow e^- + \bar{\nu}_e + \nu_\mu + \bar{\nu}_\mu \tag{8}$$

If the pions decay high enough in the atmosphere, the muons with a mean lifetime of $\tau = 2.2 \cdot 10^{-6}\,\mathrm{s}$ still have a lot of energy left, interact and can travel to the surface due to their low interaction and ionisation loss and effects of the special relativity. However, if the muon's energy is low, it decays before it can be detected. Electrons or positrons are produced in the decay, which can start electromagnetic sub-showers [18].

2.2.3 Cherenkov light

The electrons, positrons and muons produced in an EAS have very high energies and can move with a velocity v greater than the local speed of light in the atmosphere. Unlike in a vacuum, a charged particle can radiate in a medium

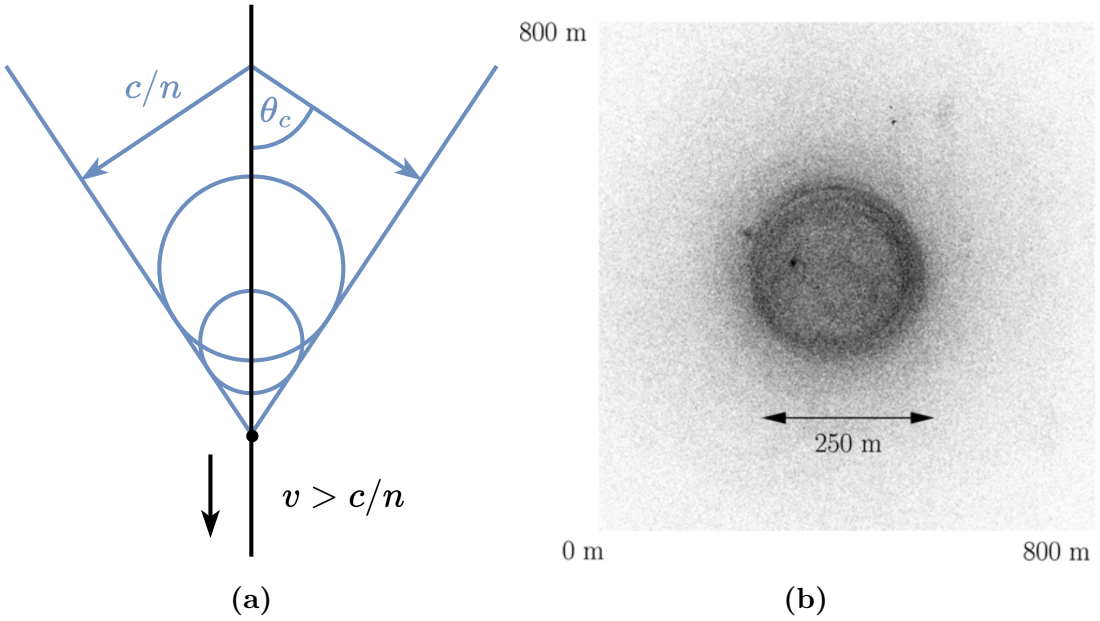


Figure 4: (a) Illustration of Cherenkov radiation, which is emitted under the angle θ_c from a charged particle travelling through the atmosphere with a velocity higher than the speed of light in this medium. Reproduced from [18]. (b) Illustration of a Cherenkov light pool on the ground from a simulated, gamma ray-induced air shower with a primary energy of 300 GeV. Adapted from [15].

without getting accelerated. When such fast-moving particles travel through the atmosphere, they cause the radiation of Cherenkov light by the particles in the atmosphere [18]. As illustrated in Figure 4a, the produced wavefront moves away from the particle trajectory under the Cherenkov angle

$$\theta_C = \arccos\left(\frac{c_0}{n(z) \cdot v}\right) \tag{9}$$

with c_0 and $n(z)$ as the speed of light in vacuum and the refractive index of the atmosphere depending on the height z above sea level [15]. Following the assumption of an exponential model [15]

$$n(z) \approx 1 + \eta_0 \cdot \exp(-z/h_0) \tag{10}$$

with parameters $\eta_0 = 2.9 \cdot 10^{-4}$ and $h_0 = 7250\,\text{m}$ for the height dependency of the refractive index, typical Cherenkov angles can be estimated. According to [23] in the higher parts of the atmosphere, the refractive index is 1.00003, resulting in maximum Cherenkov angles of 0.44°. Due to the height dependence of the refractive index, the Cherenkov angle increases while the particle is moving down. At sea level at which the refractive index has a value of 1.00029, the maximum Cherenkov angle is 1.38°. When the Cherenkov light reaches the surface, it is mostly concentrated on an elliptical to circular area for gamma ray-induced air shower, as shown in Figure 4b. The Cherenkov light produced by a hadronic air

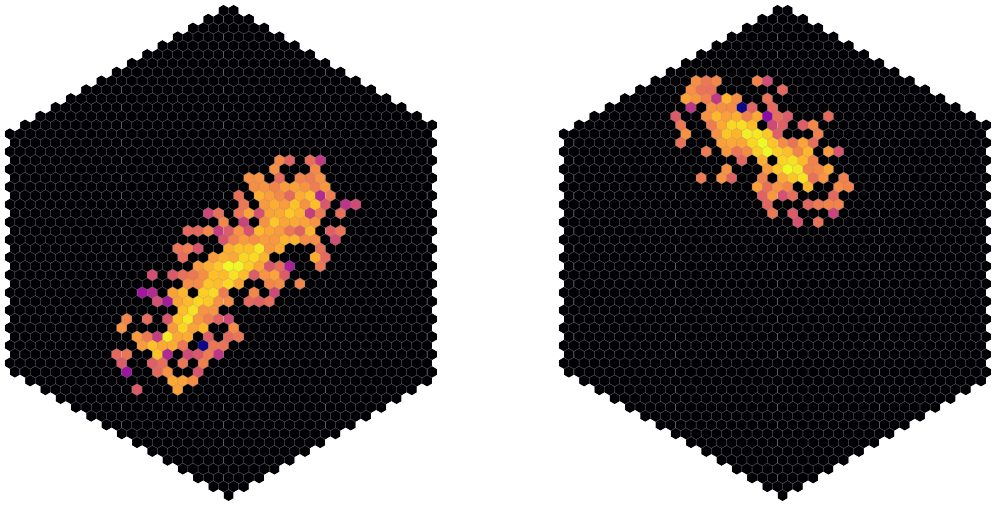


Figure 5: Two IACT images from H.E.S.S. simulations containing elliptical signals. A brighter signal is equivalent to a higher detection of Cherenkov light.

shower would be more spread over a larger area when it reaches the surface due to the shower's increased lateral distribution.

2.3 Imaging Atmospheric Cherenkov Telescopes

The following section is about Imaging Atmospheric Cherenkov Telescopes (IACTs). Their basic principle is explained in Section 2.3.1. Furthermore, two arrays of such IACTs are introduced in Section 2.3.2 and Section 2.3.3. The first one is the High Energy Stereoscopic System (H.E.S.S.), which is an IACT array of the current generation. The second one is the Cherenkov Telescope Array (CTA), which will be the next-generation IACT array. The former is especially important since the data in this work comes from H.E.S.S. simulations.

2.3.1 Basic principle

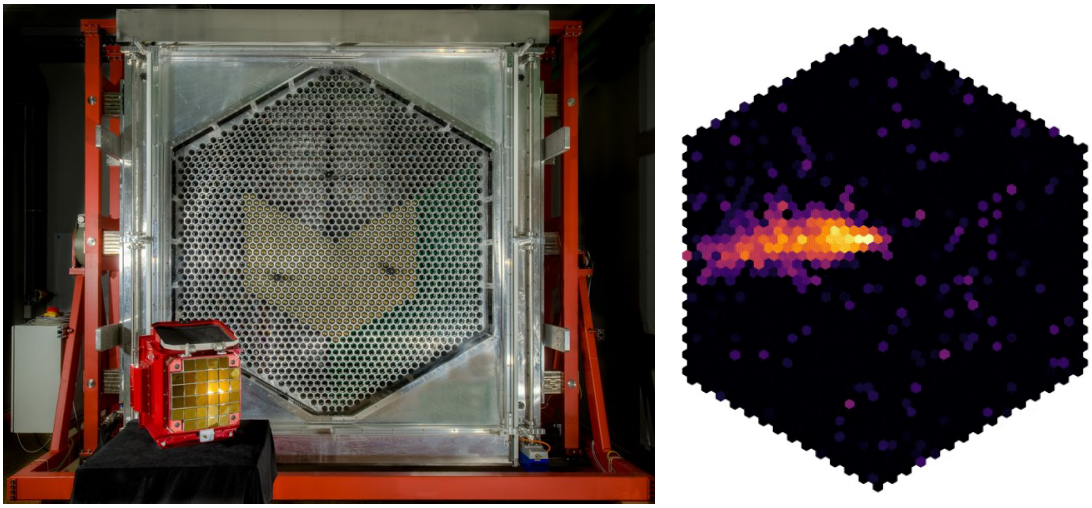
An IACT [2] is a telescope used to detect Cherenkov light emitted during the gamma-ray-induced air showers in the atmosphere. By using the imaging technique, an image of the electromagnetic air shower can be taken. The most important parts of these telescopes are the mirrors and the camera. Depending on the telescope size, the total mirror area, called the reflector, can have a diameter of many tens of metres while the camera is usually only a few hundred centimetres wide [3]. The single mirrors, whose shape, size and arrangement differ from telescope to telescope, are used to reflect and focus Cherenkov light onto the small area of the camera. The camera usually consists of several Photo Multiplier Tubes (PMTs) referred to as pixels. Modern Cherenkov telescope cameras possess more than 1000 pixels. Even though IACTs try only to detect Cherenkov light from gamma-ray-induced air showers, background light is also present during

Figure 6: Photo of the H.E.S.S. site in Namibia showing its five IACTs. CT5 is positioned in the centre, and CT1-4 are located around it in a square. Taken from [26].

IACT measurements. Any light not coming from the air showers, e.g., moonlight or light of cities, is referred to as Night Sky Background (NSB) [24]. IACT measurements should be done during dark, moonless nights with as little artificial light as possible to minimise this background. Another type of background is hadronic air showers. Their muons and electromagnetic sub-showers can also result in Cherenkov light, which the IACTs detect. The separation between gamma-ray and hadron-induced air showers is done by analysing the Cherenkov signal on the taken images. Examples of gamma images, whose signal can be approximated by an elliptical shape [25], are shown in Figure 5. Due to the inhomogeneous development of hadronic showers (compare Figure 4b) a good separation if possible. In the end, the images containing Cherenkov light and NSB are obtained with the signals in the pixels being in units of photoelectron (p.e.). By analysing the IACT images, information about the primary particle, like its energy or its direction of arrival, can be determined.

2.3.2 The High Energy Stereoscopic System

H.E.S.S. [3] is an array of five IACTs located in the Khomas Highlands in Namibia at an elevation of roughly 1.8 km above sea level and presented in Figure 6. The four small IACTs (CT1-4) are arranged in a square with a side length of 120 m and they can detect gamma rays with energies above 100 GeV. The fifth larger



(a) **(b)**

Figure 7: (a) Photo of a camera with the FlashCam design. Taken from [30]. (b) Recorded CT5 FlashCam image of a gamma-ray event, including night sky background light. Adapted from [3].

telescope (CT5) is positioned in the centre of the other telescopes and can detect gamma rays with energies of tens of GeV [27]. The small telescopes have a mirror area of about $100\,\mathrm{m}^2$ while the mirror area of CT5 is about $600\,\mathrm{m}^2$.

Up until now, H.E.S.S. has had several hardware upgrades. Initially, in phase I, the array consisted of the four small telescopes, which started operating together in December 2003 [3]. Just a few years later, the construction of CT5 started [28]. This telescope, which has been operational since July 2013, started the second phase of the array called phase II. The addition of this telescope was the first big upgrade of H.E.S.S., and it improved the overall performance of the array. In 2015/2016, the cameras of CT1-4 were replaced to improve the array performance, but especially to reduce the dead time and the system failure rate [29].

The last major hardware update of the H.E.S.S. array, which is also most important for this work, was the upgrade of the CT5 camera in October 2019. The old camera was replaced with a camera based on the "FlashCam" design [31], and nominal operation started only one month later. The new camera contains 1758 active pixels[2] with Winston cones in front of them to improve the photon collection efficiency. An example of such a camera is shown on Figure 7a. An image of a gamma-ray event, including NSB, taken with the H.E.S.S. FlashCam can be seen in Figure 7b.

H.E.S.S. has different detection modes depending on the telescopes participating in the observations. If only CT5 is used to detect Cherenkov light, the observation mode [32] is called *mono*. When two or more telescopes take part in

[2]There are six inactive pixels.

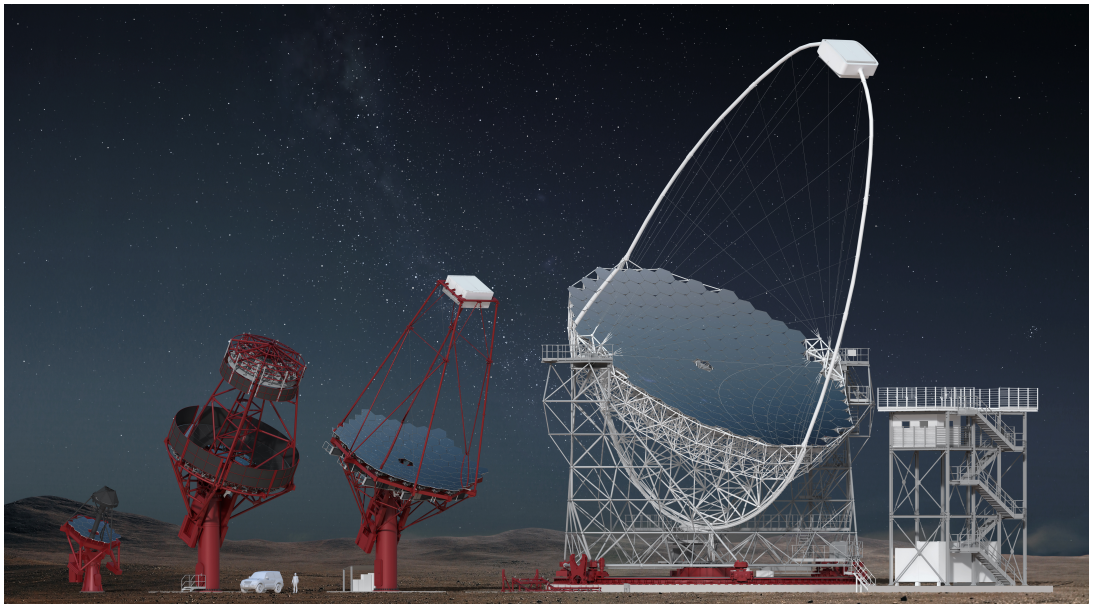


Figure 8: Image of the four different telescope types proposed for the Cherenkov Telescope Array. From left to right: Small-Sized Telescope (SST), Schwarzschild-Couder Telescope (SCT), single-mirror Middle-Sized-Telescope (MST), Large-Sized Telescope (LST). Taken from [34].

the observation, it is a *stereo* or *hybrid* observation. Detecting the Cherenkov light with more than one telescope makes it possible to get a stereoscopic view of the air shower, significantly improving the data quality.

Another important part of the telescopes is their trigger system [33], that is responsible for the read-out of detected signals. The trigger system consists generally of two parts: a camera trigger and a multiplicity trigger. The trigger in the camera triggers if a minimum amount of pixels within a sector, which is a neighbouring group of pixels, exceeds a minimum signal of typically a few photoelectrons. During stereo and hybrid observations, the whole system triggers only if the cameras of at least two telescopes trigger within a short time, typically nanoseconds.

2.3.3 The Cherenkov Telescope Array

The Cherenkov Telescope Array [35] is a future array of Imaging Atmospheric Cherenkov Telescopes aimed to be the gamma-ray observatory of the next generation. The current plan foresees one location in La Palma in the northern hemisphere and one in Chile in the southern hemisphere to cover the whole sky. The energies of the primary particles to be detected will range from about 20 GeV to 300 TeV. To increase the sensitivity over a wide energy range, CTA uses different types of telescopes, shown in Figure 8.

The Large-Sized Telescope (LST) is used for the lowest energy gamma rays with full available sensitivity between 20 GeV and 150 GeV. Its parabolic

mirror has a diameter of 23 m, and the camera contains almost 2000 pixels. For the Middle-Sized Telescopes (MST), which are sensitive in an energy range of 150 GeV to 5 TeV, there are two different designs: the single-mirror MST and the Schwarzschild-Couder Telescope (SCT). The reflector of the single-mirror MST follows the modified Davis-Cotton design, and the cameras of the MSTs can follow two different designs. The first one is the NectarCam, and the second one is the FlashCam, which is the same camera that is currently in use in the H.E.S.S. CT5 telescope. The SCT reflector consists of two separate mirrors, and the camera contains almost 12000 pixels. Small-Sized Telescopes (SSTs) are sensitive in the energy range of 5 TeV to 300 TeV and their cameras consist of 2048 pixels. Currently, it is planned to use the so-called alpha configuration for the construction of the array.

In this configuration, the northern array, CTA-North, will consist of 4 LSTs and 9 MSTs, while the southern array, CTA-South, makes use of 14 MSTs and 37 SSTs. Furthermore, CTA-south will have foundations for LSTs so that this type of telescope can be added later on.

2.4 Event reconstruction

The main goal of analysing IACT images is to obtain information about the primary particle which induced the air shower. The most important parameters to determine are the particle energy and the direction of the air shower. There are several different ways of analysing IACT images, and one of them is the Hillas parameterisation, which is explained in Section 2.4.2 after a short overview of IACT image cleaning in Section 2.4.1.

2.4.1 Image cleaning

The purpose of image cleaning is to remove unwanted signals on the image coming from the night sky background. In the default tail-cut cleaning [36], there are two threshold values for keeping pixels: a lower and a higher threshold. All pixel values smaller than the lower threshold are set to zero. Higher pixel values are kept either if they exceed the lower threshold with a neighbouring pixel exceeding the higher threshold or if they exceed the higher threshold with a neighbouring pixel exceeding the lower threshold. In the case of extended cleaning, a certain number of pixel rows around the cleaned signal is kept. This extension can be done to keep more of the detected Cherenkov light. However, this also means less of the background light is removed from the image.

Two examples of tail-cut cleaning are visualised in Figure 9 using the same IACT image. On the left, an image with extended 4/7 cleaning is shown and on the right, one with 9/16 cleaning, which is typically applied when analysing CT5 images [37]. The image with the lower cleaning thresholds contains more NSB around the edges of the signal, while the other image consists mostly of detected Cherenkov light.

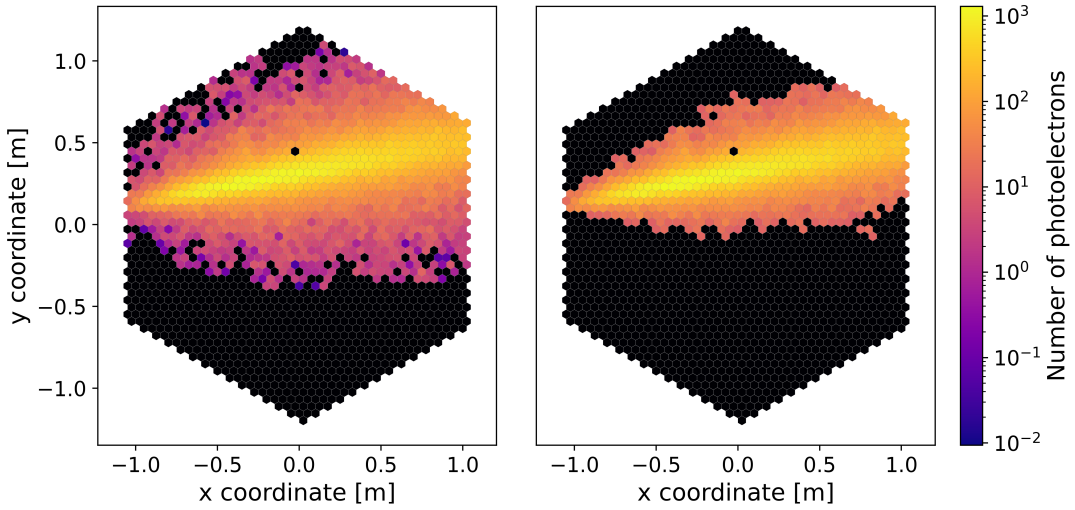


Figure 9: Illustration of tail-cut cleaning for background removal of NSB for an example H.E.S.S. CT5 image. Left: camera image with applied 4/7 extended cleaning. Here, four rows around the signal in the image with applied 4/7 cleaning are kept as extensions. Right: camera image with 9/16 cleaning applied.

2.4.2 Hillas parameterisation

Hillas parameterisation is a way of describing Cherenkov images as an ellipsis through the so-called Hillas parameters. The parameterisation is named after Michael Hillas, who first introduced it in [25]. An updated, more modern version of it can be found in [36]. Using these parameters, further analysis like event reconstruction or gamma/hadron separation can be carried out. The basic idea is to consider the signal pixels within the IACT image and approximate it with an elliptical shape. To remove pixels containing no significant Cherenkov signal, the image is cleaned. Every parameter is calculated using the position of the pixels and their detected signal. This fixed way of determining the parameters allows even the Hillas parameterisation of circular-shaped signals. The different Hillas parameters are defined as follows:

- Intensity/size (integrated signal) of the cleaned image
- Length L (spread along major axis) and width W (spread along minor axis) of the ellipsis
- Radial coordinate r and polar coordinate ϕ of the ellipsis centre
- Rotation angle ψ between the $x-$axis of the camera and the major axis of the ellipsis
- X and Y coordinate of the ellipsis centre
- Skewness (asymmetry) and kurtosis (tailedness) of the Hillas ellipsis

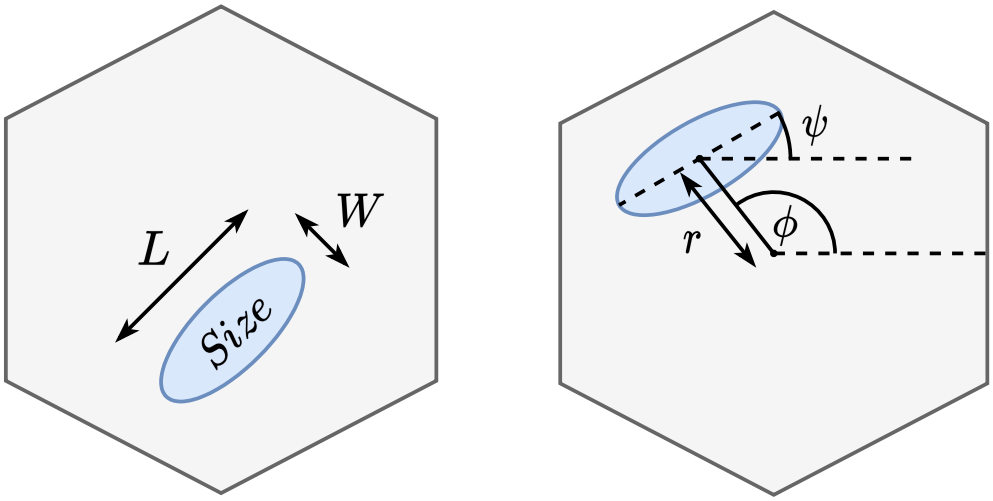


Figure 10: Illustration of some parameters of the Hillas parameterisation using an elliptical signal on the CT5 camera. Left: Hillas intensity, length and width. Right: Radial and polar coordinate and rotation angle. Taken from [1].

The Hillas size, length and width, which are illustrated on the left of Figure 10, give a description of the approximated ellipsis itself. Since the intensity of the image depends on the amount of detected Cherenkov light, the primary particle energy can be determined through the analysis of this parameter. On the right side of the same figure, the radial and polar coordinates and the rotation angle are illustrated. These three parameters give information about the location and orientation of the ellipsis on the telescope camera and thus make the directional reconstruction of the air shower possible. The X and Y coordinates contain the same information as the radial and polar coordinates, so it is sufficient to investigate one or the other. The skewness and kurtosis are generally less important Hillas parameters, but they are still interesting to examine [1].

3 Deep learning and generative models

The concept of machine learning (ML) [38, 39], as a sub-field of computer science, revolves around the idea of letting the computer "learn" a chosen task by itself after some initial setup of a human user. This can be achieved by programming a machine learning model, which inputs some data x, processes it and outputs the desired values. Machine learning itself can be divided into supervised and unsupervised learning as shown in Figure 11. In supervised learning tasks like regression or classification, the ML model – a discriminative model – learns to predict a value from the input values. The predictions are compared to labels y of the input provided by the user so that the model knows what the output should be. For example, as a regression task, a model takes IACT images as input and learns to predict the energy of the primary particle, which induced the air shower. In unsupervised learning tasks like synthesis, the ML model – a generative model – learns to produce new samples which are similar to the input samples. The generation of IACT images, like in this work, is an example of an unsupervised learning task. Deep learning is a sub-field of machine learning, and their difference is explained later on.

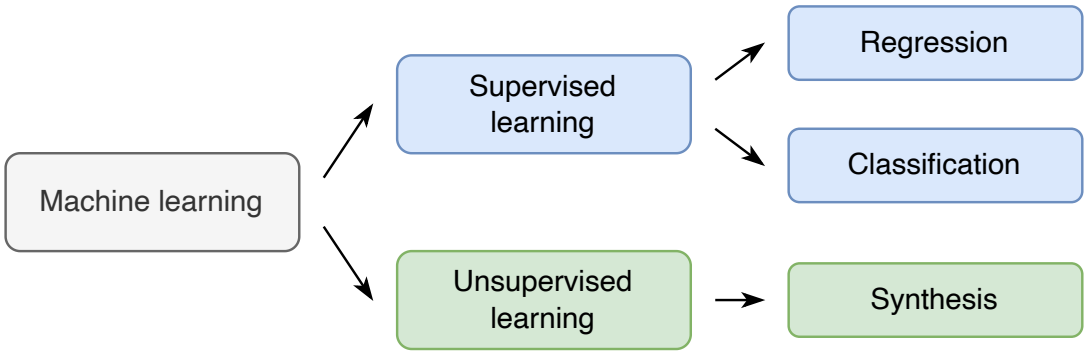


Figure 11: Illustration of different types of machine learning training and tasks.

In this section, deep learning and generative models are explained to be able to understand everything around the framework for IACT image generation. In Section 3.1, the fundamental architecture used in machine learning, the neural network, is discussed. A detailed overview of neural network training is given in Section 3.2. Furthermore, convolutional neural networks, which are a special type of neural network, are explained in Section 3.3. The theoretical aspects of the machine learning approach for image generation – generative models – are discussed in Section 3.4. The information for this section is mostly taken from [38].

3.1 Neural network structure

The architecture that machine learning revolutionised is the so-called neural network (NN). To make the explanation of it as easy as possible, the concept

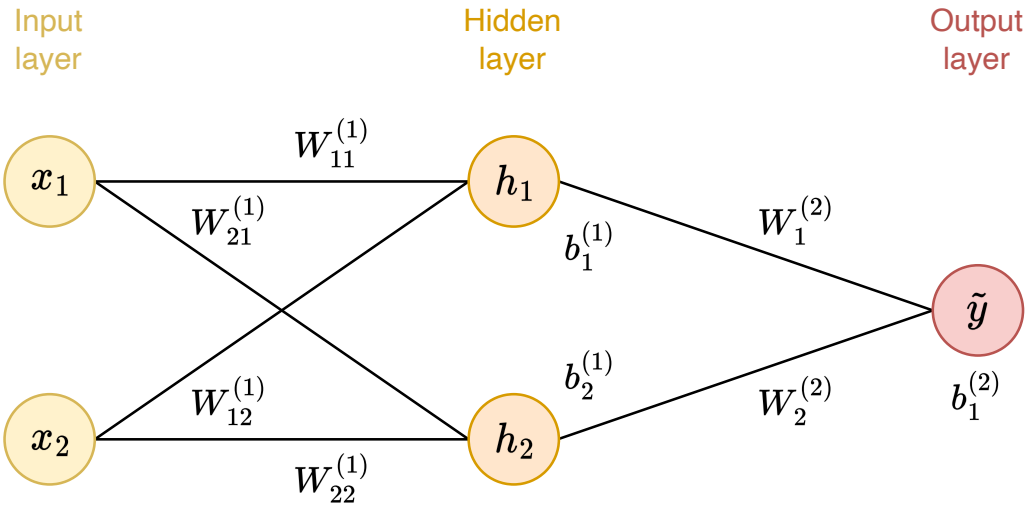


Figure 12: Simple neural network with two input nodes x_i, one hidden layer with two nodes h_i, and one output node y. Additionally, the weights W and biases b, which are the adaptive network parameters, are shown.

is explained using a simplistic example, shown in Figure 12. A neural network consists of several layers, which can be separated further into an input layer, hidden layers and an output layer. Here, the difference between machine and deep learning can already be explained. A neural network is called deep if it contains more than one hidden layer.

Each layer consists of several nodes, which are a representation of numbers. If all nodes of every layer are connected with all nodes of the previous and following layer, the network is called fully connected. The layers of a fully connected network are called *Dense* [40] layers. As the name of the input layer may imply, it processes the input data first that is given into the neural network. In this example, there would be two input features called x_1 and x_2 representing the data input into the network. In the hidden layer with nodes h_i, the actual learning of the neural network happens by processing the input data. In this case, there is only one output node called $\tilde{y}$. The nodes of the different layers are connected using so-called weights W and biases b, which are the adaptive parameters of the neural network. These connections are modelled as linear transformations. The value of the node h_1 in the hidden layer is calculated as follows:

$$h_1 = W_{11}^{(1)} \cdot x_1 + W_{12}^{(1)} \cdot x_2 + b_1^{(1)} \tag{11}$$

The adaptive weights $W_{11}^{(1)}$ and $W_{12}^{(1)}$ are multiplied with the input nodes, respectively, and an additional bias $b_1^{(1)}$ is added onto the resulting value. If we stack another layer on top of the hidden layer, putting the calculations of all nodes together would result in the final function for the output node $\tilde{y}$, the neural

network $\mathcal{Y}$:

$$\begin{aligned}\mathcal{Y}(x, W, b) &= \tilde{y} \\ &= \left(W_{11}^{(2)} W_{12}^{(2)}\right) \times \left[\begin{pmatrix} W_{11}^{(1)} W_{12}^{(1)} \\ W_{21}^{(1)} W_{22}^{(1)} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \end{pmatrix}\right] + b_1^{(2)} \\ &= W^{(2)} \left(W^{(1)} x + b^{(1)}\right) + b^{(2)}\end{aligned} \tag{12}$$

The parameters $W^{(1)}$, $W^{(2)}$ are matrices and $b^{(1)}$, $b^{(2)}$ are vectors. So, as shown in this example, neural networks are basically just a function f involving matrix multiplication, which takes some input x and returns some outputs $\tilde{y}$.

However, there is one problem with neural networks that follow the above structure. All of these NNs only describe a linear model, no matter how many hidden layers are used. To prevent this and add non-linear mapping to neural networks, so-called activation functions, which are non-linear, are used. These functions are applied to the node value calculation $Wx + b$. One of the most common activation functions is the ReLU activation

$$\sigma_{\text{ReLU}} = \max(0, x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

This function maps all negative values to zero and keeps all other values as they are. Another activation function used in this work is LeakyReLU [41]:

$$\sigma_{\text{LeakyReLU}} = \begin{cases} x, & \text{if } x > 0 \\ a \cdot x, & \text{otherwise} \end{cases} \tag{14}$$

Just as for the ReLU activation, all positive values are kept the same. However, negative values are multiplied by a parameter a, which is between 0 and 1. After applying activation functions to the hidden layer of the simple neural network, as described in Equation (12), its function changes and looks as follows:

$$\mathcal{Y}(x, W, b) = W^{(2)} \sigma\left(W^{(1)} x + b_1\right) + b_2 \tag{15}$$

This is a typical example of a neural network with only one hidden layer. However, this function can be expanded to describe networks with multiple hidden layers. A neural network with k hidden layers with $k \geq 1$ and adaptive parameters W and b is then defined by:

$$\mathcal{Y}(x, W, b) = \sigma(W^{(k+1)} \sigma\left(W^{(k)} ... + b_k\right) + b_{k+1}) \tag{16}$$

In this case, activation functions are applied after every hidden layer and at the network's output. This function $\mathcal{Y}$ now defines a deep neural network, the central building block of deep learning.

3.2 Training of neural networks

After introducing the main architecture of machine learning, the training of neural networks is explained in the following part. Before going into the training procedure itself in Section 3.2.3, loss functions and optimisers have to be discussed in Section 3.2.1 and Section 3.2.2 since they are essential parts of neural network training. Followed by that is a short overview of residual learning blocks used to improve NN training in Section 3.2.4 and generalisation and regularisation in Section 3.2.5.

3.2.1 Objective function

The objective function $\mathcal{L}$, also called loss or cost function, is a function comparing the output $\tilde{y}$ of the neural network to the actual label y of the data sample. The function is a measure of how well the network reconstructs the label with the given data. An example of such a loss function is the mean squared error (MSE), here shown for one sample x:

$$\mathcal{L}(\theta) = (\mathcal{Y}(x, \theta) - y)^2 \tag{17}$$

In this case, an objective function of zero would mean the neural network outputs would match exactly the actual label. Since the goal of the network training is precise reconstruction, this function is minimised during the training. Usually, the network is not trained sample by sample since this is computationally inefficient, but it uses batches. So, the equation is expanded to batches of N samples, which are evaluated simultaneously, resulting in the following objective function:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\mathcal{Y}(x_i, \theta) - y_i)^2. \tag{18}$$

3.2.2 Optimiser

The adaptive parameters θ of the network can be updated using the loss function and a gradient-based optimiser. The optimiser is responsible for the update of the parameters using the information that the output of the loss function provides. A simple optimiser is the gradient descent algorithm

$$\tilde{\theta} \to \theta - \alpha \cdot \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} \tag{19}$$

with $\tilde{\theta}$ being the updated and more optimised parameters. The parameter α is the learning rate, which defines how much the parameters are changed in one step. It is typically set to an initial small value ($\sim 10^{-3} - 10^{-4}$). In every optimisation step, the respective gradients of the loss function are determined and used for the calculation of the updated parameter. Afterwards, the parameters are more optimised for the task of the neural network, which results in a smaller difference

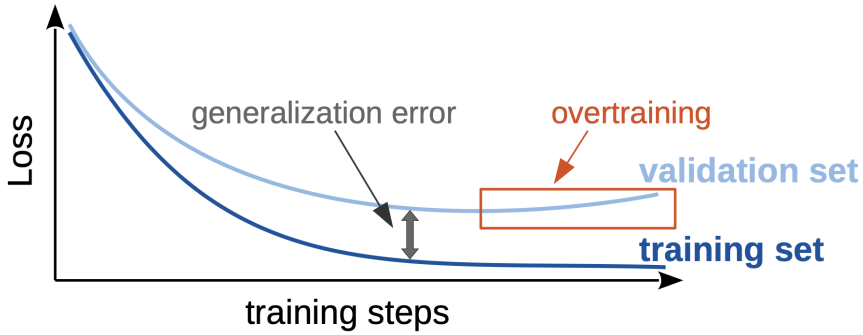


Figure 13: Illustration of the typical trend of loss curves during neural network training. The loss curve of the training set continuously decreases while the loss curve of the validation set increases at some point due to overtraining of the network. The difference between the training and validation loss is called generalisation error. Taken from [38].

between NN output $\tilde{y}$ and the actual label y. If the network is trained using batches, then the optimiser is called stochastic gradient descent.

Gradient descent is not the best optimiser for parameter updates. In this work, the Adam optimiser [42] is used every parameter optimisation. The update of the parameters is defined as follows:

$$\tilde{\theta} \rightarrow \theta - \alpha \cdot \frac{\widehat{m}(\beta_1, \mathrm{d}\mathcal{L}/\mathrm{d}\theta)}{\sqrt{\widehat{v}(\beta_2, \mathrm{d}\mathcal{L}/\mathrm{d}\theta)} + \epsilon} \tag{20}$$

Generally, this looks very similar to the optimisation using gradient descent. However, the term containing the gradient looks vastly different and more complex. The parameters $\widehat{m}$ and $\widehat{v}$ are the so-called bias-corrected first-moment estimate and bias-corrected second raw moment estimate, and they are calculated using the gradient of the loss function, β_1 and β_2, where ϵ is used for numerical stability to prevent division by 0. The parameters ϵ, β_1 and β_2 are settable hyper parameters. The Adam optimiser is chosen since it is computationally efficient and a good choice for NN training with large amounts of data, like in our case.

Another important option for optimiser is the learning rate decay. The learning rate initially set usually decreases by a bit after every step to make the learning more efficient towards the end of the network training. However, it has to be so chosen that the learning rate does not get too low at some point since the update of the parameters would then be negligible. Decaying the learning rate by one order of magnitude, e.g. from $\sim 10^{-3}$ to 10^{-4}, during one training is a good choice.

3.2.3 Training and testing

A neural network is an algorithm processing some data using parameters $\theta = (W, b)$ that are adaptive. However, the optimal network parameters are unknown and

must be determined. This is done by training the network to "learn" to find the optimal value for each parameter. The general concept of neural network training is quite simple and is based on gradient optimisation.

Initially, the adaptive parameters are randomly sampled. In the first training step, one sample (or a batch of samples) of the training data is given into the network, which processes it through the different layers. Through the use of the loss function and the optimiser, the parameters of every layer are updated during backpropagation using the chain rule to calculate the different gradients. For better understanding, the update of the parameter $W_{11}^{(1)}$ of the simple network in Figure 12 is carried out in detail using Equation (17) as loss function and gradient descent as optimiser:

$$
\begin{aligned}
\widetilde{W}_{11}^{(1)} &= W_{11}^{(1)} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial W_{11}^{(1)}} = W_{11}^{(1)} - \alpha \cdot \frac{\partial (y(x,\theta) - y)^2}{\partial W_{11}^{(1)}} \\
&= W_{11}^{(1)} - \alpha \cdot \frac{\partial \left(\left[W^{(2)} \sigma \left(W^{(1)} x + b_1 \right) + b_2 \right] - y \right)^2}{\partial W_{11}^{(1)}} \\
&= W_{11}^{(1)} - \alpha \cdot 2(y(x,\theta) - y) \cdot \frac{\partial \left(W^{(2)} \sigma \left(W^{(1)} x + b_1 \right) + b_2 \right)}{\partial W_{11}^{(1)}} \\
&= W_{11}^{(1)} - \alpha \cdot 2(y(x,\theta) - y) \cdot W^{(2)} \cdot \frac{\partial \sigma \left(W^{(1)} x + b_1 \right)}{\partial W_{11}^{(1)}} \\
&= W_{11}^{(1)} - \alpha \cdot 2(y(x,\theta) - y) \cdot W^{(2)} \cdot \frac{\partial \sigma \left(W^{(1)} x + b_1 \right)}{\partial \left(W^{(1)} x + b_1 \right)} \cdot \frac{\partial \left(W^{(1)} x + b_1 \right)}{\partial W_{11}^{(1)}} \\
&= W_{11}^{(1)} - \alpha \cdot 2(y(x,\theta) - y) \cdot \frac{\partial \sigma \left(W^{(1)} x + b_1 \right)}{\partial \left(W^{(1)} x + b_1 \right)} \cdot \left(W_1^{(2)} W_2^{(2)} \right) \times \begin{pmatrix} x_1 \\ 0 \end{pmatrix} \\
&= W_{11}^{(1)} - \alpha \cdot 2(y(x,\theta) - y) \cdot \frac{\partial \sigma \left(W^{(1)} x + b_1 \right)}{\partial \left(W^{(1)} x + b_1 \right)} \cdot W_1^{(2)} x_1
\end{aligned}
$$

After calculating the updated adaptive parameters, one training step is finished, and the network has improved regarding its task and data evaluation. Then, the next training sample is given into the network to improve its performance further. The same steps of evaluating the input data, calculating the objective function and updating adaptive parameters are carried out.

After the network has evaluated the whole training data set once, an epoch is finished, and it starts going through the whole data set again. While training the network, the loss function is monitored and saved for every epoch or batch as shown in Figure 13. In an ideal scenario of optimal training, the loss curve of the training would continuously decrease since the network learns how to map correctly from input to output and is improving from batch to batch. The training is continued until the loss curve converges, i.e. a saturated NN performance is reached, which means the network cannot improve anymore. At this point, the network is either sufficiently trained to do the task accurately or evaluates poorly and its architecture has to be improved.

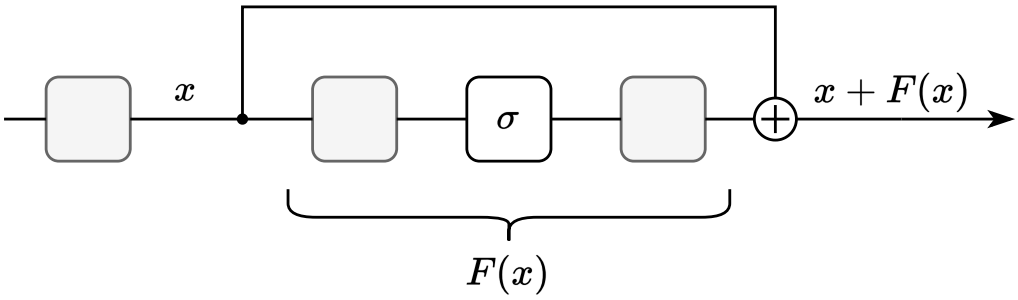


Figure 14: Illustration of a residual learning block.

In the end, after training the network over several epochs, the network is trained and ready to be used. As a final test to verify that the neural network can do its task properly, a test data set is given to the network to evaluate some new data.

3.2.4 Residual learning blocks

Most of the networks in this work make use of so-called residual learning (ResNet) blocks [43]. They make the optimisation and training of very deep neural networks easier. The principle of these ResNet blocks is illustrated in Figure 14. In the beginning, there is the data x, which is the output from previous layers. This data is processed through two more layers with activation in between. The output $F(x)$ of the last layer is then added with the identity x in an *Add* [40] layer. The result is then forwarded to the next layers in the network. The use of several ResNet blocks in one network can significantly increase its performance.

3.2.5 Generalisation and regularisation

A challenge for NNs is their memorisation of data, which means they learn the training data too well and evaluate it well but do not do the same for new data. This phenomenon is called overtraining, and it can be observed and prevented using a validation set, which is part of the initial data set. After every epoch, the validation data set is given to the trained network, evaluated and followed by the calculation of the loss. It has to be emphasised that this loss is not used for parameter updates, and its sole reason is to look for possible overtraining. If the training and validation loss decrease, the network learns the task properly and can be applied to new data. However, suppose the validation loss starts increasing again at some point, so its generalisation worsens, as shown in Figure 13. In that case, the network parameters are optimised for the training data but not for new data, i.e., the validation set.

The challenge of overtraining can be solved using regularisation. There are different types, but the one used in this work is dropout [44]. The basic idea is to deactivate some nodes during the network training to let the network not memorise too much about the training data. In every layer of the network, a

chosen percentage of nodes, usually between 0.2 and 0.5, is deactivated. The nodes which are affected by the drop out are changed in every training iteration. During the use of the validation set and the later use of the network for new data, the deactivated nodes are activated again.

3.3 Convolutional neural networks

A convolutional network [45] is a special type of neural network especially efficient for the extraction of features from natural images. The main difference to the fully connected network introduced above is that not all nodes of one layer are connected to all nodes of the next layer. Instead, a convolutional operation, explained in Section 3.3.1, is used between layers. An overview of several other advanced operations for convolutional neural networks is given in Section 3.3.2.

3.3.1 Convolutional operation

Instead of multiplying the incoming nodes with the weights, convolutions with a kernel are utilised in convolutional layers, as shown in Figure 15. These Kernels slide over the image, and two operations are carried out in every step. An element-wise multiplication of the Kernel and the specific part of the image is performed, followed by summing up all the resulting values. The output of this Kernel application represents the value of the new image. The two convolutions highlighted in this simple example are, thus, calculated as follows:

$$\tilde{I}_{11} = I_{11} \cdot W_{11} + I_{12} \cdot W_{12} + I_{21} \cdot W_{21} + I_{22} \cdot W_{22}$$

$$\tilde{I}_{23} = I_{23} \cdot W_{11} + I_{24} \cdot W_{12} + I_{33} \cdot W_{21} + I_{34} \cdot W_{22}$$

A particular property is the sharing of the weights since this supports the translational symmetry in images.

As seen in this example, convolutions reduce the dimension of the image. Usually, several kernels are applied to the image during the same convolution, which results in the increase of the third dimension of the images. They are similar to the number of nodes and are to be chosen by the user. So, for example, applying 16 kernels to the example image of dimension $3 \times 4 \times 1$ would result in an outcoming image of dimension $2 \times 3 \times 16$. Using multiple filters ensures the extraction of as many features as possible from the input images. Activation functions are also applied to convolutional layers and change the output accordingly.

After many convolutional layers, the output is usually given into a fully connected neural network to process the extracted features more efficiently. This is done by using a *Flatten* [40] layer, which changes the three-dimensional image to a one-dimensional vector. The length of the vector is defined as image height $\times$ image width $\times$ number of filters. The other way around, using a *Reshape* [40] layer, makes it possible to connect a fully connected layer to a convolutional layer.

Since this work is about IACT images, all of our networks consist mostly of *Conv2D* [40] layers to learn the image features properly.

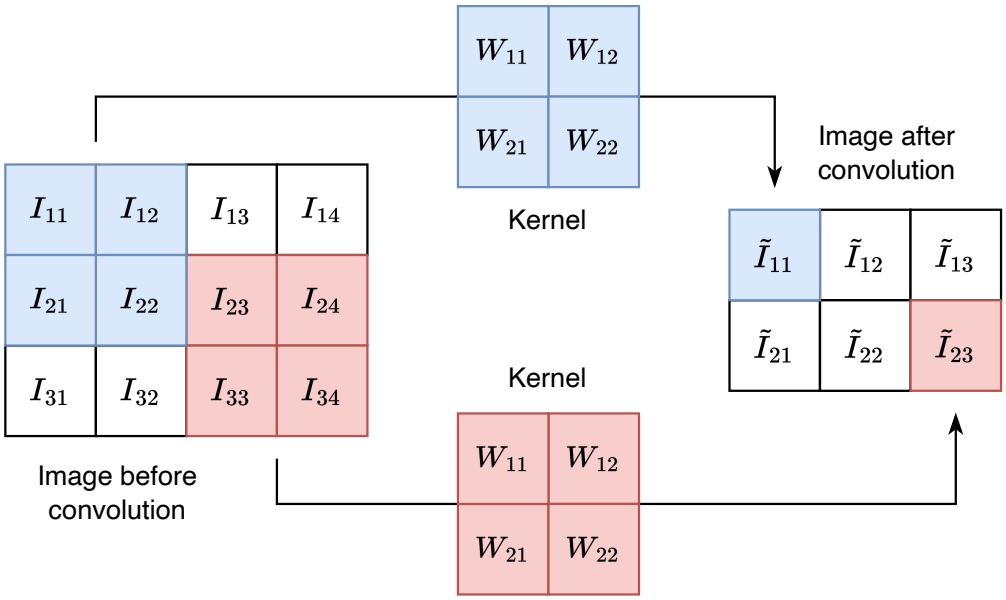


Figure 15: Illustration of convolutions in a convolutional neural network using two example calculations. The kernel is moved over the image and multiplied piece-wise with certain parts of the image. The resulting values are summed up, and the new parts of the images are obtained.

3.3.2 Advanced operations

There are several special operations that can be specifically applied to convolutional layers. Three of these operations are important for our work and thus will be explained in detail.

Pooling Pooling [46] is a way to reduce the dimension of the input image beside the default convolution. As seen in the upper left part of Figure 16, pooling can be divided further into different types, e.g. AveragePooling and MaxPooling. The chosen pool size directs the reduction of the image dimension. For this example image of dimension 4×4, pool sizes of 2×2 are chosen. In AveragePooling, the mean of all numbers in a pool is used as the new value of the reduced image at the position of the pool. In comparison, in MaxPooling, the highest number of the pool is used as the new value. In this work, *AveragePooling2D* [40] layers are used in the various networks to reduce the IACT image dimension.

Zero padding Both convolutional operations and pooling layers end up reducing the dimension of the input image. To prevent the reduction in the former case, zero padding [38], shown in the lower left of Figure 16, can be used to keep the dimension of the image. Applying a 3×3 kernel to a zero-padded image of dimension 2×3 results in an output image of the same dimension. Thus, with zero padding, many convolutional layers can be used in a row to extract as many features as possible without reducing the dimension.

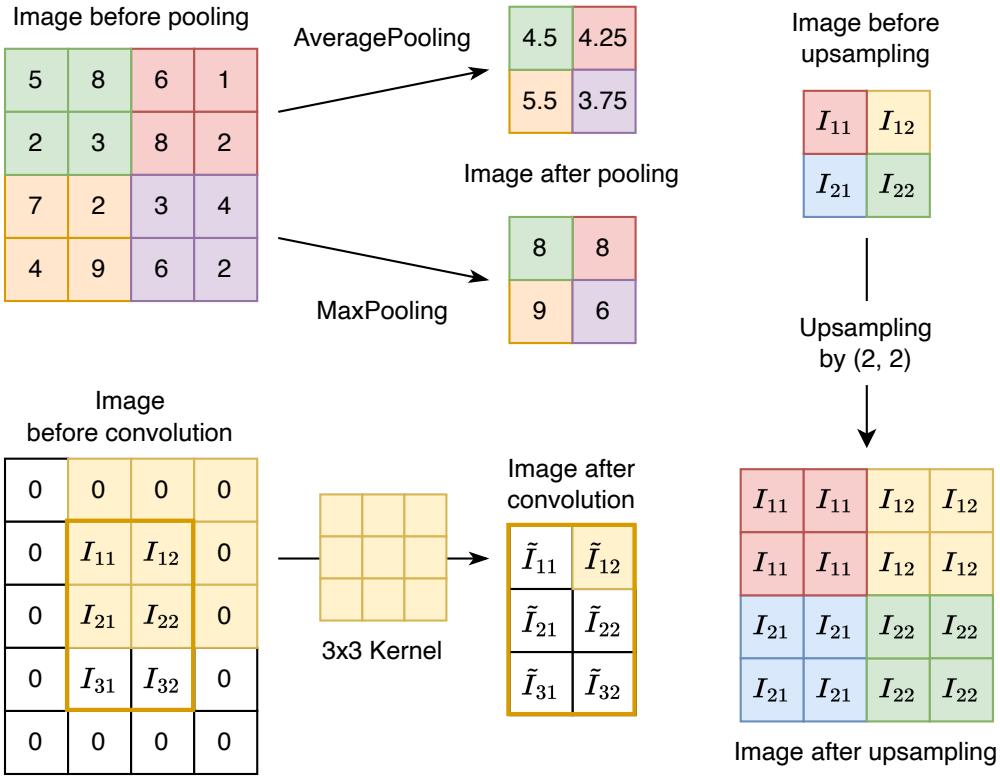


Figure 16: Illustration of pooling (upper left), zero padding (lower left) and upsampling (right).

Upsampling While pooling is used for the reduction of image dimension, upsampling [38], shown on the right of Figure 16, is an operation which can do the opposite and increase the image dimension. In this example an image of shape 2×2 is upsampled by 2 for both height and width, resulting in an image of shape 4×4. This operation is especially important for our work since we want to generate IACT images. Thus, we make use of *Upsampling2D* [40] layers.

3.4 Generative models

After explaining the basics of deep learning, the concept of generative models used for image generation in this work is discussed. Currently, several generative types of generative models exist [47]. In this work, Generative Adversarial Networks are used due to their low computational complexity, which offers ultra-fast simulation speeds. In the rest of this section, the concept of GANs and their improved version, Wasserstein GANs, is explained. Furthermore, the concept of conditioning is discussed to enforce the implementation of certain properties during the generation process.

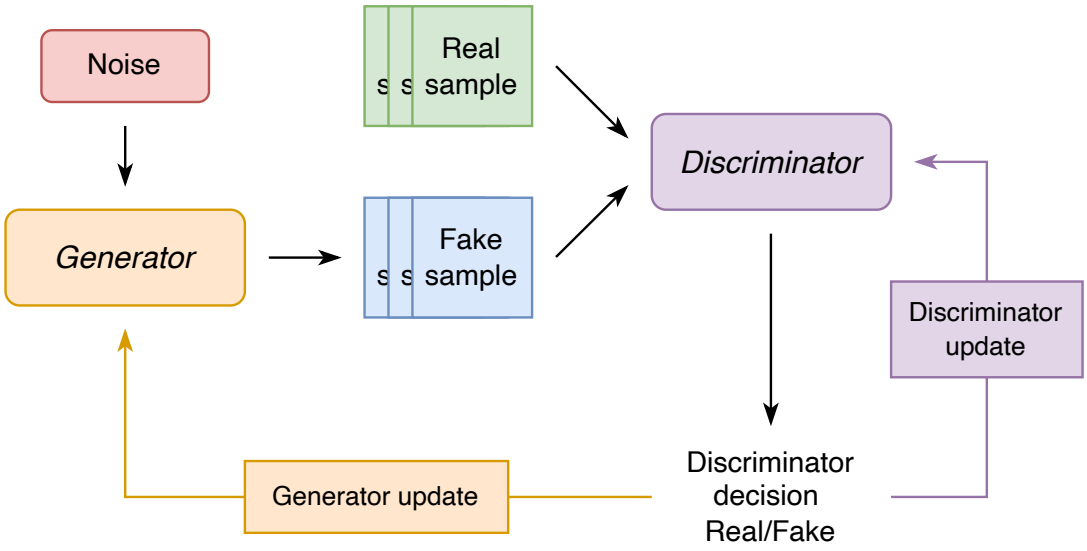


Figure 17: Training principle of a Generative Adversarial Networks. During the discriminator (purple) training, the network takes real and fake samples and evaluates them regarding their realness. The output is then used for the update of the discriminator. The generator (orange) is trained and updated by using the feedback of the discriminator after evaluating fake samples provided by the generator after injecting noise.

3.4.1 Generative adversarial networks (GAN)

A generative adversarial network is a special generative model and was introduced in [8]. It is a framework comprising two adversarial networks, usually called generator G and discriminator D. The basic principles of both networks are illustrated in Figure 17.

Generator and discriminator The generator, highlighted in orange, aims to generate realistic-looking samples by approximating the real data distribution p_r with samples x_r with a model distribution. It takes noise z sampled from a simple distribution p_z as input and outputs a generated fake sample $x_\theta = G_\theta(z)$ from the model distribution p_θ. The parameters θ are the adaptive parameters of the network.

The discriminator, highlighted in purple, is used to provide feedback for the generator. The network takes real and generated samples x and outputs a probability of the sample being from the real data distribution.

GAN training As the name of the GAN implies, the two networks are trained in an adversarial fashion. The loss function of the GAN framework is defined as follows:

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{x_r \sim p_r}[\log(D(x_r))] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G_\theta(z)))]. \tag{21}$$

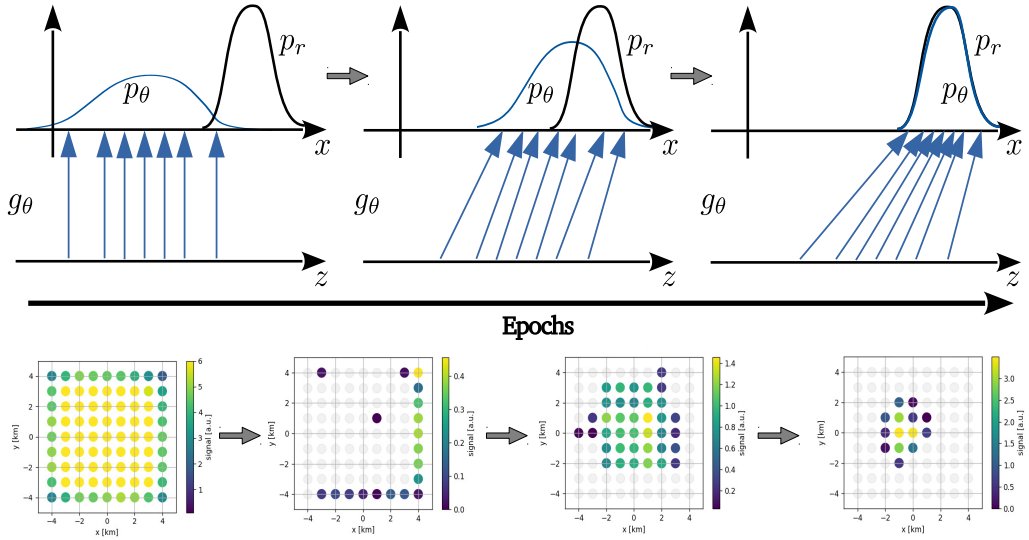


Figure 18: Visualisation of GAN training using cosmic-ray footprints as samples to be generated. Taken from [38].

In the first step of the training, the discriminator is trained while the parameters of the generator are fixed. For that, fake images are needed, which are generated by giving noise to the generator. The resulting fake images are put into the discriminator together with real images to identify their "realness". Using the output, the loss is calculated, and the parameters of the discriminator are updated. This is performed by maximising the GAN loss since the maximum of $\mathcal{L}_{\text{GAN}} = 0$ would mean the network can identify every real and fake image.

Next, the training of the generator follows. The network takes noise as input, generates fake images, and inputs them into the discriminator. However, this time, no real samples are used, and the discriminator parameters are fixed, so only the fake images' realness is evaluated. The output is used to calculate the GAN loss for the generator to update the network parameters accordingly. The generator aims to minimise the GAN loss in Equation (21) since this would mean the discriminator identifies the generated images as real ones with a probability of 100 %.

After the generator update, one iteration of the WGAN training is finished. From now on, the cycle is repeated. The discriminator is trained, followed by the generator, the discriminator and the generator. In a perfect scenario, this training would continue until the generator produces perfectly real samples and the discriminator can no longer distinguish between real and fake samples

An example of such a GAN training is shown in Figure 18. In this training, the generator g_θ tries to generate cosmic-ray footprints. In the beginning, the model distribution p_θ does not match the real distribution p_r since the generator does not yet know how to map the noise z properly. Thus, only noisy images are generated. After some training, both distributions match better, and the footprints start to look more real. At the end of the training, the generator

knows how to map from noise to realistic-looking samples and the real and model distribution match. The generator is now able to generate cosmic-ray footprints.

Challenges of GAN training Even though the use of GAN for image generation can already come with remarkable results, training such a generative model comes with many challenges [38, 48].

In mode collapsing, the generator learns to produce only certain samples of the data. Since the generator is fixated on specific modes during the training, the discriminator focuses on distinguishing real and fake samples from this mode. However, this causes the generator to produce samples from another mode. This mode collapsing continues, resulting in the generator not being able to produce samples from the whole data set simultaneously.

Another challenge of GAN training is vanishing gradients. If the discriminator is trained till it can optimally decide between real and fake, the generator cannot improve anymore since its gradients go towards zero.

Furthermore, the monitoring of the similarity between the real and fake distributions is a challenge of GAN training. Due to the distributions being high-dimensional and disjoint at the beginning of the training, it is difficult to find a suitable metric to measure their similarity.

3.4.2 Wasserstein GAN

Due to all the training challenges of traditional Generative Adversarial Networks, an improved version of a GAN called Wasserstein GAN [49] is chosen for image generation in this work. The major difference between the WGAN and the GAN is the output of the discriminator and the used objective. In the WGAN, the discriminator takes real and fake images and tries to approximate the Earth-Mover (EM) distance, also called Wasserstein distance (WD). The WD is used as a measure of optimally transforming the model distribution into the real data distribution. A high value for the Wasserstein distance means the distributions do not match very well, and a value of zero means both distributions are equal. Since the discriminator no longer classifies and gives feedback about the distributions, the network is called the critic C. The Wasserstein distance can be approximated by the critic using the Kantorovich-Rubenstein duality

$$W = \sup_{\|C\|_L \geq 1} \mathbb{E}_{x_r \sim p_r}[C(x_r)] - \mathbb{E}_{x_\theta \sim p_\theta}[C(x_\theta)] \tag{22}$$

and training the network to be a 1-Lipschitz function $C(x)$. In the initial WGAN work, the Lipschitz constraint is enforced using weight clipping, so the weights are restricted to be in a predetermined range. However, this is not an optimal way to do this, and that is why, for this work, the Lipschitz constraint is enforced using the gradient penalty

$$\mathcal{L}_{\text{GP}} = \lambda \cdot \mathbb{E}_{x_u \sim p_u} \left[\left(\|\nabla_{x_u} C(x_u)\|_2 - 1 \right)^2 \right] \tag{23}$$

introduced in [50]. The gradient penalty weight λ is used for the scaling of the loss and is a hyperparameter. The mixed samples x_u are taken from a distribution p_u, which is obtained by sampling uniformly from lines between pairs of points from p_r and p_θ. The mixed samples are defined as:

$$x_u = \epsilon \cdot x_r + (1 - \epsilon) \cdot x_\theta. \tag{24}$$

The parameter ϵ is randomly sampled form $U[0, 1]$. This way of constraining is chosen to ensure that the critic provides in between the two distributions a gradient with norm 1, e.g. adequate feedback. The loss of a Wasserstein GAN with gradient penalty (WGAN-GP) looks as follows:

$$\begin{aligned} \mathcal{L}_{\text{WGAN-GP}} = \mathbb{E}_{x_r \sim p_r}[C(x_r)] - \mathbb{E}_{z \sim p_z}[C(G_\theta(z))] \\ + \lambda \cdot \mathbb{E}_{x_u \sim p_u}\left[(\|\nabla C(x_u)\|_2 - 1)^2\right] \end{aligned} \tag{25}$$

The training of a WGAN works very similar to the normal GAN with only minor differences. During the training of the Wasserstein GAN, the generator aims to minimise $\mathcal{L}_{\text{WGAN-GP}}$ since a Wasserstein distance of zero would mean the model distribution matches with the real data distribution. On the contrary, the critic tries to maximise the loss to learn the differences between the real data and model distributions. Normally, the critic is trained using several batch iterations before a generator training. This is done since a change in the model distribution would also change the Wasserstein distance between the two distributions, and, therefore, the critic needs the extra training steps to approximate the Wasserstein distance sufficiently. This ensures precise feedback from the critic. In contrast to the WGAN, the GAN setup cannot be trained in a similar way since a perfect discriminator yields gradients of norm 0 [51].

3.4.3 Conditioning of physical properties

Even though an optimally trained WGAN is able to produce realistic-looking samples, it is not possible for the framework to generate images for specifically chosen labels. The generator takes noise and produces images with random properties implemented. That is why the framework in this work is conditioned on the physics labels using the AC-GAN approach explained in [52]. In this work, the physical labels to be implemented into the IACT images during their generation are the primary particle energy E and the coordinates of the air shower impact point $I = (I_x, I_y)$, which is the location of the air shower core on the surface. So, first of all, the labels are given to the generator as additional input besides the noise, resulting in images $x_\theta = G_\theta(z, E, I)$. However, nothing guarantees that these physical properties are implemented correctly into the respective images during the generation process. This can be enforced with the help of two additional networks, $\hat{E}$ and $\hat{I}$, which have to be added to the WGAN framework.

These two so-called constrainer networks take samples x from the real data and try to reconstruct the respective physical label. The quality of the

reconstruction is estimated using the MSE, i.e.,

$$\begin{aligned} \mathcal{L}_{\text{con, E}} &= [E - \hat{E}(x, I)]^2 \\ \mathcal{L}_{\text{con, I}} &= [I - \hat{I}(x, E)]^2, \end{aligned} \tag{26}$$

during the training. As an additional input for the constrainer, the label not being reconstructed is given, e.g., the primary particle energy is given to the impact constrainer during impact point reconstruction. This choice is made since it improves the overall reconstruction of the networks. The reconstructed labels are then used to update the parameters of the constrainers, resulting in the networks being able to improve their reconstruction.

During the generator training, the fake samples x_θ are given to the constrainer together with fake labels, and the physical labels are reconstructed. However, this time, the parameters of the networks are fixed, and they are only used for evaluating the input. The feedback for the generator is obtained by using the outputs of the reconstructions for the calculation of the auxiliary loss

$$\mathcal{L}_{\text{aux}} = \alpha_I \cdot \mathcal{L}_{\text{con, I}}(x_\theta, E, I) + \alpha_E \cdot \mathcal{L}_{\text{con, E}}(x_\theta, E, I). \tag{27}$$

This loss is added to the default generator loss, and by minimising it, the generator learns to correctly implement the given physical labels into the images during the generation. The auxiliary parameters α_i are used for the scaling of the individual constrainer losses and are hyperparameters. Lastly, the physical labels are also given to the critic as additional input. More information for the network improves the estimation of the Wasserstein distance, resulting in better feedback for the generator.

4 Simulation and preparation of IACT events

As explained before, within this work, a novel tool for generating IACT images is developed to support MC simulations and make the process of event simulations more efficient. The programs for the standard MC simulations are explained in Section 4.1. In Section 4.2, the H.E.S.S. simulations, which result in the data for the training of our framework, are discussed in detail. The pre-processing of the simulated data to make it suitable for machine learning usage is presented in Section 4.3.

4.1 CORSIKA and sim_telarray

Simulations are a powerful tool for better understanding electronics, instrumentation, and physics around gamma-ray astronomy. Many different programs are available to simulate air showers and instrument response. The packages commonly used in the astroparticle physics community, and for this work, are CORSIKA [4] and `sim_telarray` [5].

CORSIKA CORSIKA (Cosmic Ray SImulations for KAscade) is a Monte Carlo program for simulating and studying extensive air showers. The two major tasks of the program are the simulation of the particle interactions happening in the atmosphere and the particles' travelling between interactions. The primary particle initiating the air shower in the atmosphere can be freely chosen, resulting in the possible simulation of various air shower types. Besides using protons and photons as primary particles, heavy nuclei and even neutrinos can be used. At high energies, hadronic interactions are modelled using a phenomenological approach. There are several interaction models for hadronic showers and electromagnetic showers that can be used. In the case of hadronic air showers, there are seven different interaction models. Five of these models are used for the simulation of hadronic interactions at high energies, and the remaining two models exist for hadronic interactions at lower energies. Electromagnetic interactions can be simulated with one of the two existing interaction models.

Another simulation option of CORSIKA, which is especially important for gamma-ray astronomy, is the simulation of Cherenkov light. The specific models chosen to simulate the data used in this work will be discussed in more detail. An important extension for CORSIKA is the IACT/ATMO package [53]. This package allows a proper simulation of IACT configurations. The detectors are simulated as spheres, which measure the photon bunches from the Cherenkov light with bunches usually consisting of $5 - 10$ photons.

Several examples of air showers simulated with CORSIKA are shown in Figure 19. The first two showers are of electromagnetic nature with primary particle energies of 100 GeV and 10 TeV. In contrast, the last shower is induced by a proton with an energy of 100 GeV. Comparing the two gamma showers shows how an increase in energy affects the shower structure. The increase and decrease

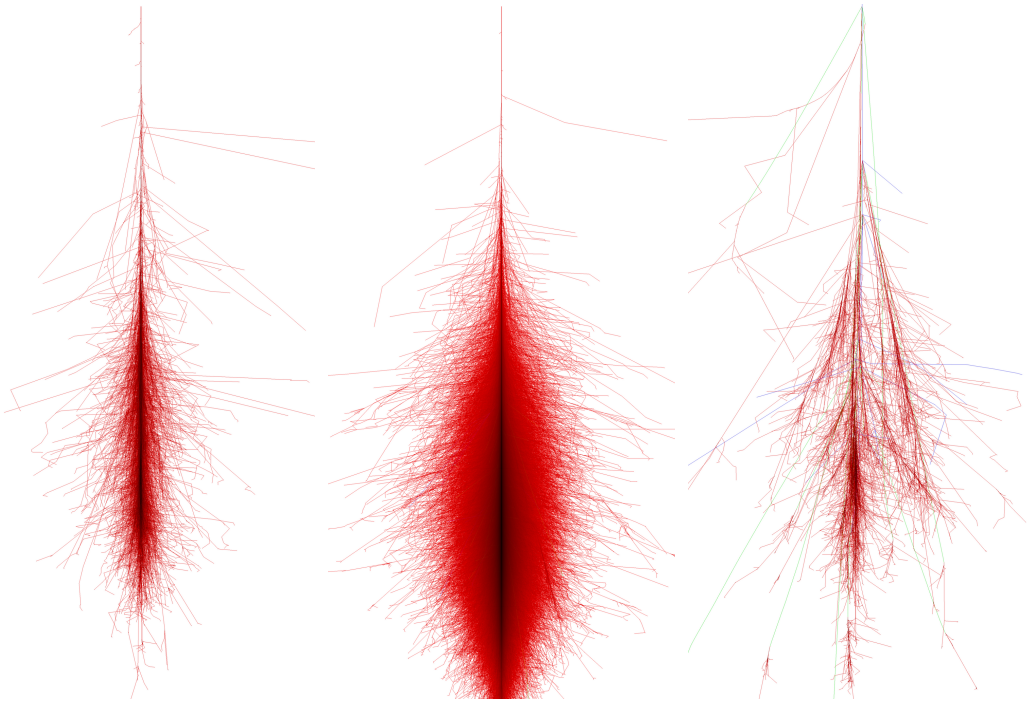


Figure 19: Examples of extensive air showers simulated with CORSIKA. Left: Electromagnetic air shower induced by a primary photon with an energy of 100 GeV. Middle: Electromagnetic air shower induced by a primary photon with an energy of 10 TeV. Right: Hadronic air shower induced by a primary proton with an energy of 100 GeV. Taken from [54].

in the number of particles during the shower development is very visible in the case of the higher energy primary particle. The comparison of the electromagnetic and hadronic shower with energies of 100 GeV highlights the more high lateral distribution of the latter.

sim_telarray The photon bunches generated with CORSIKA are given to `sim_telarray`, which is a package for simulating the instrument response, including the mirrors and the camera. Like in real measurements, at the end of the instrument response, the IACT camera images are obtained.

ML-based simulations Even though simulations are an irreplaceable tool, using Monte Carlo code has a major downside. Simulations can take quite a while (up to several days or weeks) [55], especially when using protons and heavy nuclei as primary particles in the air shower simulation. Furthermore, a lot of memory space is needed to store the large shower libraries. A machine-learning approach can significantly speed up the simulation time by several orders of magnitude. Several other benefits come with such an approach. An ML model is memory efficient since it is smaller than 100 MB and due to the small size, it is also easy

to distribute the model. Furthermore, using an ML model allows even people without knowledge of specific simulation programs like CORSIKA to simulate events. Another interesting advantage is the possible generation of events with properties typically not directly settable [56] in simulations, e.g. the maximum shower depth X_0.

Optimally, one would use two machine learning models for event simulations, one for the air shower simulation and one for the simulation of the instrument response. However, due to that complexity, a simplification can be done as a first step by combining these models into one. This approach is followed in this work.

4.2 Simulation of training data

The H.E.S.S. events used in this work were simulated using CORSIKA and `sim_telarray` with the latter one using the latest H.E.S.S. configuration. Since gamma events are simulated, the most occurring interactions in the air shower are electromagnetic. For their interactions during the CORSIKA simulations, the Electron-Gamma-Shower program (EGS4) [57] was chosen as the interaction model. Hadronic interactions, which can also happen during the shower development with a smaller probability, were simulated using the QGSJET-II-04 [58] interaction model. The parameters and settings used for simulating the air showers and instrument response are as follows:

- The energies of the primary particles inducing the air showers lie in the range of 10 GeV to 300 TeV.
- The spectrum of the particle flux follows E^{-2}.
- The zenith angle, which denotes the angle between the zenith and the induced air shower, is set to 20°.
- The H.E.S.S. observation mode is *mono*, so only events triggering CT5 are read out.
- The azimuthal angle of the telescope direction is 0°, so they are oriented into northern direction.
- The opening angle of the telescopes, so their field of view, is 5°.

After the H.E.S.S. events were simulated, the raw data were obtained. The raw events were calibrated and cleaned using the H.E.S.S. Analysis Program (HAP). In the calibration step, the measured analog-to-digital counts are converted into the physical unit, photoelectrons (p.e.). The simulated event images were cleaned with the extended 4/7 cleaning, with the extension being four rows of pixels. Lastly, the simulated data were analysed in HAP and prepared for this work.

4.3 Pre-processing of data

After obtaining the processed events used for the image generation in this work, they are prepared for machine learning usage. The IACT images have to be changed so that they can be used as input for a CNN. Furthermore, we also want to ensure stable training [38], and that is why the input data will be normalised. If the data varies over several orders of magnitude, the update of the network parameters can get problematic. After the networks are trained, and new images are generated, the pre-processing steps have to be reverted to get the initial input formats.

The data are split into training, validation, and test data sets with a ratio of 70:15:15. Before applying any other changes to the different data sets, all events with a primary particle energy lower than $10^{-1.5}$ TeV and higher than 10^{2} TeV are filtered out. These cuts are chosen because of the statistics and the low signal detected by the telescopes at low energies. After the cut, in total, 515727 events remain for training, validation and test data sets. Only the CT5 images were taken from the events because we decided to generate images of a single telescope as a first step. Furthermore, the primary particle energy and the air shower impact point are taken since the physical representation of these two parameters is chosen to be implemented into the images during their generation.

4.3.1 IACT images

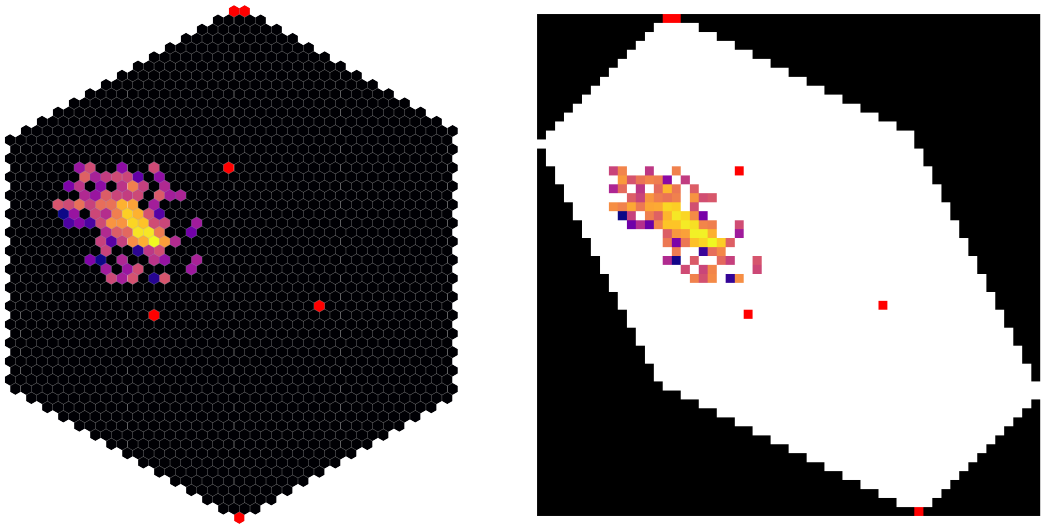
Since the pixels of the images are hexagonally arranged, the images have to be transformed using axial addressing [59] so that they can be used as input for convolutional neural networks. During this transformation, which is illustrated in Figure 20, the 1764 camera pixels are rearranged to be on a grid of shape 56×56. To fill this grid, zeros are added around the rearranged image. It is important to mention that no information about the images and their features is lost using this transformation.

After the calibration process and the cleaning, some pixels may contain negative values after the analysis in HAP. Since it is easier for the ML model to generate images with pixel values of zero or higher, the values of the negative pixels mostly contain noise and are clipped to zero. Furthermore, the pixels p_i of all training, validation and test images are normalised as follows:

$$p_{i,\text{normalised}} = \frac{\log_{10}(1 + p_i)}{\sqrt{\sum_{j=1}^{N_p} \left(p_{j,\text{train}} - \bar{p}_{\text{train}}\right)^2 / N_p}} \tag{28}$$

The parameter N_p is the number of all pixels values $p_{j,\text{train}}$ in the training data, so $N_t \times 56 \times 56$ with N_t being the number of images in the training data. The parameter $\bar{p}_{\text{train}}$ is the mean value of all these pixels.

There are also six highlighted pixels in Figure 20, which are turned off during the simulations. The three upper and lower ones are most of the time masked in the actual camera, while the inner three pixels are mount pixels [31].



(a) (b)

Figure 20: Illustration of axial addressing, which transforms a H.E.S.S. CT5 camera image from (a) hexagonal shape into (b) rectangular shape needed for usage in CNNs. The brightness of the pixels corresponds to their signal values. The red pixels are turned off. The black pixels in the hexagonal image do not contain any signal. The white space in the rectangular image corresponds to the camera pixels, and the black area is the needed padding of zeros to obtain the rectangular image shape.

During the event calibration in HAP, the three inner pixels get a signal value by taking the average of the neighbouring pixels. This is done so that these anomalies do not affect the follow-up analysis of the image. Since we want to simulate the most realistic-looking images, we decided to mask these three pixels to zero in the simulated data as we do no consider such second-order effects in the analysis.

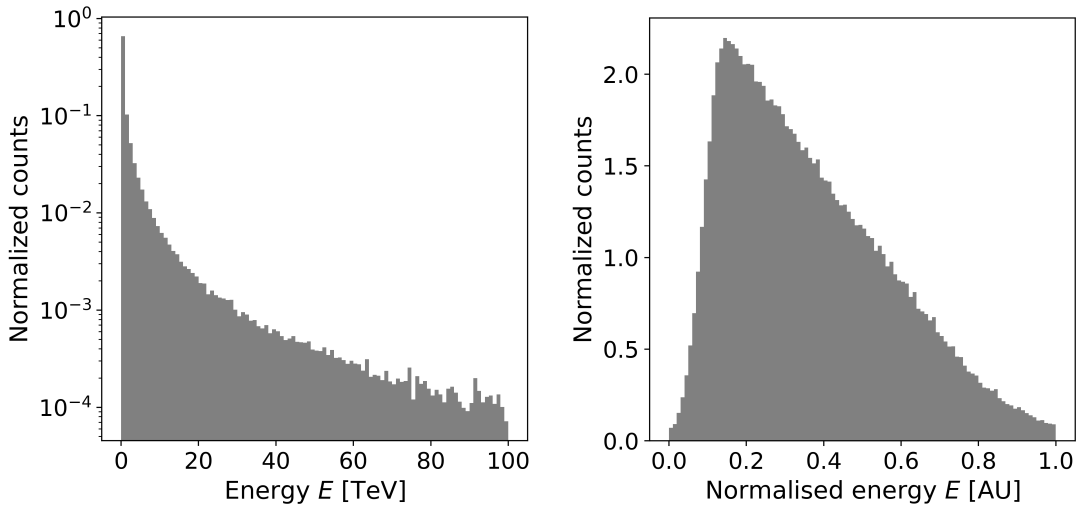
4.3.2 Energy and impact point

Both physical labels used to condition the generator process are prepared differently for machine learning usage. For all three data sets, the primary particle energy lies between $10^{-1.5}$ TeV and 10^{2} TeV as shown in Figure 21a. The normalisation of the energy E_i works as follows:

$$E_{i,\text{normalised}} = \frac{\log_{10}(E_i) + 1.5}{3.5} \tag{29}$$

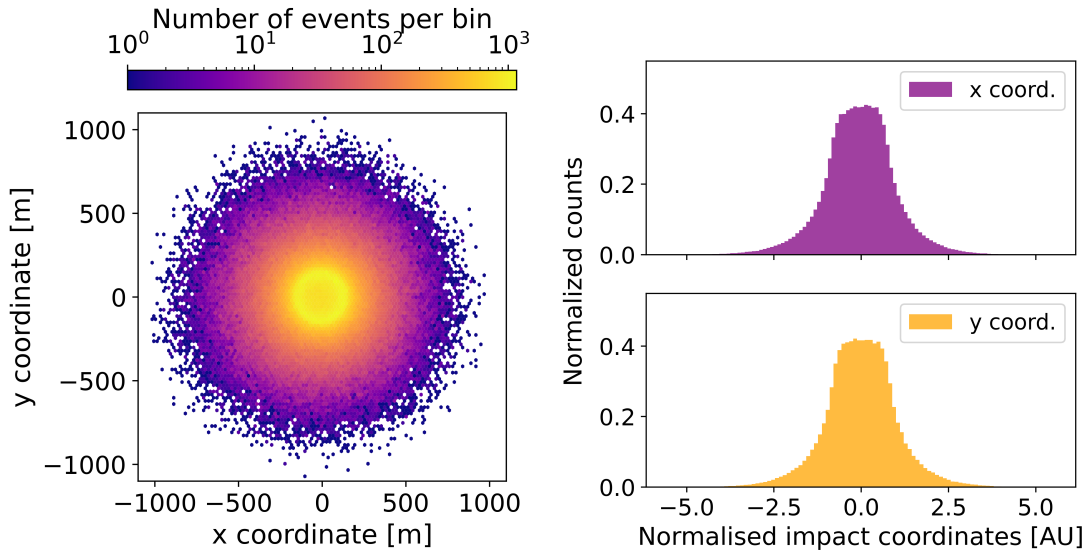
The normalisation function takes the logarithm with base 10 of the energy and shifts the resulting data to the interval [0, 1]. The distribution of the normalised energy, which is now prepared for machine learning usage, is shown in Figure 21b.

The coordinates x_i and y_i of the air shower impact point that roughly range from -1000 m to 1000 m in both directions as shown in Figure 22a, are normalised



(a) (b)

Figure 21: Distribution of the primary particle energy of the simulated data set after applying the energy cut (a) before the pre-processing and (b) after the pre-processing.



(a) (b)

Figure 22: (a) Map of the impact points of the simulated data set before pre-processing. (b) Distributions of the impact point coordinates of the simulated data set after pre-processing.

separately:

$$x_{i,\text{normalised}} = \frac{x_i}{\sqrt{\sum_{j=1}^{N_t} \left(x_{j,\text{train}} - \bar{x}_{\text{train}}\right)^2 / N_t}} \tag{30}$$

$$y_{i,\text{normalised}} = \frac{y_i}{\sqrt{\sum_{j=1}^{N_t} \left(y_{j,\text{train}} - \bar{y}_{\text{train}}\right)^2 / N_t}} \tag{31}$$

To obtain the normalisation factors for each coordinate, the respective standard deviation of the training data set is calculated. After applying the factors, both coordinates roughly range from $-5\,\text{m}$ to $5\,\text{m}$ as shown in Figure 22b.

5 WGAN framework for the generation of IACT images

For the generation of IACT images in this work, we use a conditional Wasserstein Generative Adversarial Network with a gradient penalty. The theory of this type of generative model and conditioning of physical properties is explained in Section 3.4. In this section, everything around our framework, illustrated in Figure 23, is explained in detail. The four networks – the generator, the critic, and the two constrainers – which are part of the framework are explained in Section 5.1. In Section 5.2, an overview of the setup for training the framework is given, followed by the detailed training procedure and a short look at the post-processing of the generated images. Lastly, the computational speed-up of a machine learning approach for image generation over the standard simulations is discussed in Section 6.4. Parts of this section are in a similar or the same form in [1].

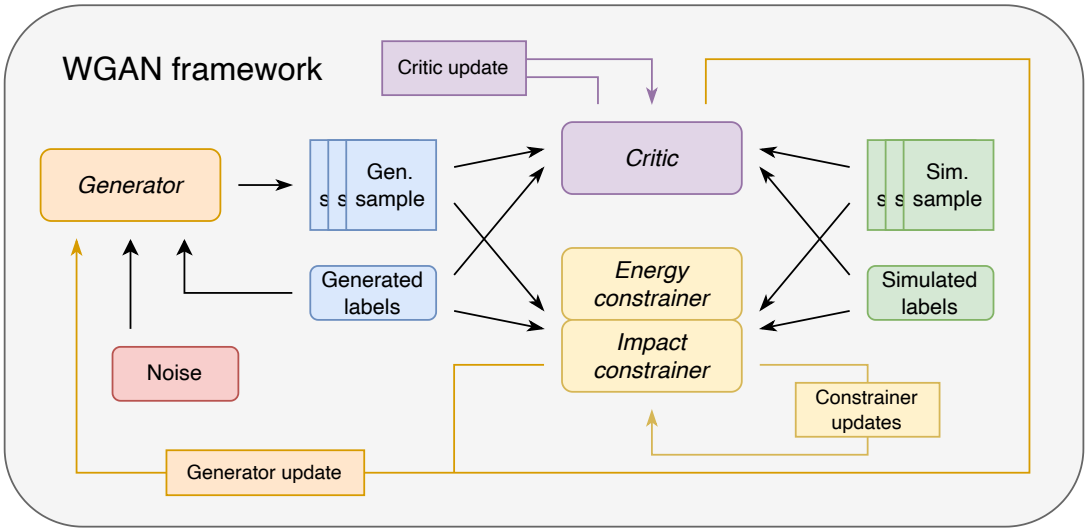


Figure 23: Illustration of the WGAN framework. The generator (orange) and the critic (purple) are used for the generation of IACT images (blue), while both constrainers (yellow) enforce the correct implementation of the physical properties into the images. The simulated images (green) are for the training of the different networks.

5.1 Network designs

Our framework for the IACT image generation consists of four networks: the generator, the critic, the energy constrainer, and the impact constrainer. The generator and critic are the main parts of the WGAN framework used for the actual image generation. At the same time, the two constrainers are utilised

Nr	Input shape	Layer	Output shape	Features
1	100+1+2	Concatenate	103	–
2	103	Dense + ReLU	12544	–
3	12544	Reshape	7×7	256
4	7×7	Conv2D + ReLU	7×7	256
5	7×7	Conv2D + ReLU	7×7	256
6	7×7	Upsampling2D	14×14	256
7	14×14	Conv2D + ReLU	14×14	128
8	14×14	Conv2D + ReLU	14×14	128
9	14×14	Upsampling2D	28×28	128
10	28×28	Conv2D + ReLU	28×28	64
11	28×28	Conv2D + ReLU	28×28	64
12	28×28	Upsampling2D	56×56	64
13	56×56	Conv2D + ReLU	56×56	32
14	56×56	Conv2D + ReLU	56×56	32
15	56×56	Conv2D + ReLU	56×56	1
16	56×56	Multiply	56×56	–

Table 1: Generator architecture. This network takes a 100-dimensional noise vector, the primary particle energy, and the coordinates of the air shower impact point as input and outputs and an IACT image of dimension 56×56. Adapted from [1].

for enforcing the correct implementation of physics labels during the generation process.

5.1.1 The generator

The generator, which is highlighted in orange in Figure 23, is a convolutional neural network with about three million parameters whose purpose is the generation of IACT images. The network inputs noise (red) and physical labels (blue) and outputs a 2D image (blue).

The exact architecture of the network is shown in Table 1. The input noise is a 100-dimensional vector sampled from a simple distribution. In our case, the noise distribution is a multivariate Gaussian with a mean value of zero and a standard deviation of one. The physical labels are the primary particle energy and the air shower impact point coordinates. They are given into the generator alongside the noise vector, resulting in an input of dimension 103 for the first actual layer. After the first fully connected layer, the 12544 nodes are rearranged in a Reshape layer to be in shape 7×7 with 256 features. Next, Conv2D layers and ReLU activations follow so that the generator can create the features of the IACT image. Additionally, Upsampling2D layers are used to increase the width and height of the image. The output image is an IACT image of dimension 56×56.

A very important part of this network is the second to last and last layer.

Nr	Input shape	Layer	Output shape	Features	
1	56×56	Conv2D + LeakyReLU	56×56	32	
2	56×56	Conv2D + LeakyReLU	56×56	32	
3	56×56	Conv2D	56×56	32	$2\times$
4	56×56	Add	56×56	32	
5	56×56	LeakyReLU	56×56	32	
6	56×56	AvgPooling	28×28	–	
7	28×28	Conv2D + LeakyReLU	28×28	64	
8	28×28	Conv2D + LeakyReLU	28×28	64	
9	28×28	Conv2D	28×28	64	$2\times$
10	28×28	Add	28×28	64	
11	28×28	LeakyReLU	28×28	64	
12	28×28	AvgPooling	14×14	–	
13	14×14	Conv2D + LeakyReLU	14×14	128	
14	14×14	Conv2D + LeakyReLU	14×14	128	
15	14×14	Conv2D	14×14	128	$2\times$
16	14×14	Add	14×14	128	
17	14×14	LeakyReLU	14×14	128	
18	14×14	AvgPooling	7×7	–	
19	7×7	Conv2D + LeakyReLU	7×7	256	
20	7×7	Conv2D + LeakyReLU	7×7	256	
21	7×7	Conv2D	7×7	256	$2\times$
22	7×7	Add	7×7	256	
23	7×7	LeakyReLU	7×7	256	
24	7×7	Flatten	12544	–	
25	12544	Dense	100	–	
26	1	Dense	50	–	
27	2	Dense	50	–	
28	100+50+50	Concatenate	200	–	
29	200	Dense + LeakyReLU	100	–	
30	100	Dense + LeakyReLU	100	–	
31	100	Dense	100	–	
32	100	Add	100	–	
33	50	Dense	1	–	

Table 2: Critic architecture. The network takes an IACT image of dimension 56×56, the primary particle energy, and the air shower impact point as input and outputs a value, which is used for the approximation of the Wasserstein distance. The layers of the architecture marked with "$2\times$" are used twice in a row. Adapted from [1].

In the former layer, we decided to use the ReLU activation since the pre-processed images of our data contain pixel values of zero and higher due to the zero clipping.

A maximum value is chosen for the activation function because of the saturation of pixel values in the simulated data set caused by the used electronics. The maximum pixel value of the whole data set is set to the maximum value in the ReLU activation. Furthermore, in the last layer, the output of the ReLU activation layer is multiplied by a mask, which sets some pixels to zero and others to one. The pixels set to zero are the ones on the rectangular image, which are not part of the actual camera, and the mount pixels. All other pixels are multiplied by one to keep their value as they are output by the ReLU activation layer. This mask is visualised in Figure 20b by the black and red areas.

5.1.2 The critic

The critic, which is highlighted in purple in Figure 23, is a convolutional neural network with almost five million parameters used to provide feedback to the generator for image generation. The network takes simulated (green) or generated (blue) images and physical labels as input and outputs a value used for the approximation of the Wasserstein distance.

In Table 2, the architecture of the network is shown. The network input is an image with dimension 56×56, the primary particle energy and the coordinates of the impact point. However, the physical labels are not directly given to the first layer and are instead used later on. The image is given to the critic, and its dimension is reduced using Conv2D layers with an increasing number of features and LeakyReLU activations as part of residual learning blocks. After reducing the images to size 7×7, they are flattened, and the resulting 12544 nodes are used as input of another fully connected layer with 100 output nodes. Now, the physical labels are given as input to fully connected layers with 50 output nodes each. The resulting 200 nodes from the image and the labels are concatenated, and a few more fully connected layers follow. The last layer is a dense layer with one node without activation to enable the approximation of the Wasserstein distance.

5.1.3 The energy and impact constrainer

The energy and impact constrainer, which are highlighted in yellow in Figure 23, are convolutional neural networks with about 200k parameters each. Their purpose is to enforce the correct implementation of the physical label properties (blue) provided during the image generation. They work as feedback for the generator and are trained to reconstruct the physical labels given an IACT image (blue/green).

The architectures of both networks, which are almost identical, are shown in Table 3 and Table 4. The general structure of the networks is similar to the critic structure. They take simulated and generated images of dimension 56×56 as input, as well as the physical label not being reconstructed, e.g. the energy constrainer takes an image and impact point as input and reconstructs the energy from that. The architecture features residual learning blocks and consists of Conv2D layers with LeakyReLU activation and AveragePooling layers. The image is flattened and concatenated with the remaining physical label after reducing

Nr	Input shape	Layer	Output shape	Features
1	56×56	Conv2D + LeakyReLU	56×56	16
2	56×56	Conv2D + LeakyReLU	56×56	16
3	56×56	Conv2D	56×56	16
4	56×56	Add + LeakyReLU	56×56	16
5	56×56	AvgPooling	28×28	–
6	28×28	Conv2D + LeakyReLU	28×28	16
7	28×28	Conv2D + LeakyReLU	28×28	16
8	28×28	Conv2D	28×28	16
9	28×28	Add + LeakyReLU	28×28	16
10	28×28	AvgPooling	14×14	–
11	14×14	Conv2D + LeakyReLU	14×14	32
12	14×14	Conv2D + LeakyReLU	14×14	32
13	14×14	Conv2D	14×14	32
14	14×14	Add + LeakyReLU	14×14	32
15	14×14	AvgPooling	7×7	–
16	7×7	Conv2D + LeakyReLU	7×7	32
17	7×7	Conv2D + LeakyReLU	7×7	32
18	7×7	Conv2D	7×7	32
19	7×7	Add + LeakyReLU	7×7	32
20	7×7	AvgPooling	3×3	–
21	3×3	Conv2D + LeakyReLU	3×3	64
22	3×3	Conv2D + LeakyReLU	3×3	64
23	3×3	Conv2D	3×3	64
24	3×3	Add + LeakyReLU	3×3	64
25	3×3	Flatten	576	–
26	576	Dense + LeakyReLU	50	–
27	2	Dense + LeakyReLU	50	–
28	50+50	Concatenate	100	–
29	100	Dense + LeakyReLU	50	–
30	50	Dropout	50	–
31	50	Dense	1	–

Table 3: Energy constrainer architecture. The network takes an IACT image of dimension 56×56 and the air shower impact point as input and outputs primary particle energy represented in the IACT image. Adapted from [1].

the dimension to 3×3. The energy constrainer has one output node, while the impact constrainer has two, and there are no activation functions applied to the nodes. Unlike in the critic, the output values of the constrainer networks have a meaning, with them being the reconstructed physical values.

Nr	Input shape	Layer	Output shape	Features
1	56×56	Conv2D + LeakyReLU	56×56	16
2	56×56	Conv2D + LeakyReLU	56×56	16
3	56×56	Conv2D	56×56	16
4	56×56	Add + LeakyReLU	56×56	16
5	56×56	AvgPooling	28×28	–
6	28×28	Conv2D + LeakyReLU	28×28	16
7	28×28	Conv2D + LeakyReLU	28×28	16
8	28×28	Conv2D	28×28	16
9	28×28	Add + LeakyReLU	28×28	16
10	28×28	AvgPooling	14×14	–
11	14×14	Conv2D + LeakyReLU	14×14	32
12	14×14	Conv2D + LeakyReLU	14×14	32
13	14×14	Conv2D	14×14	32
14	14×14	Add + LeakyReLU	14×14	32
15	14×14	AvgPooling	7×7	–
16	7×7	Conv2D + LeakyReLU	7×7	32
17	7×7	Conv2D + LeakyReLU	7×7	32
18	7×7	Conv2D	7×7	32
19	7×7	Add + LeakyReLU	7×7	32
20	7×7	AvgPooling	3×3	–
21	3×3	Conv2D + LeakyReLU	3×3	64
22	3×3	Conv2D + LeakyReLU	3×3	64
23	3×3	Conv2D	3×3	64
24	3×3	Add + LeakyReLU	3×3	64
25	3×3	Flatten	576	–
26	576	Dense + LeakyReLU	50	–
27	1	Dense + LeakyReLU	50	–
28	50+50	Concatenate	100	–
29	100	Dense + LeakyReLU	50	–
30	50	Dropout	50	–
31	50	Dense	2	–

Table 4: Impact constrainer architecture. The network takes an IACT image of dimension 56×56 and the primary particle energy as input and the air shower impact point represented in the IACT image. Adapted from [1].

5.2 Training and image generation strategy

Setup of training For the NN training, TensorFlow [60] on version 2.11.0 and Keras [40] on version 2.11.0 are used. Our framework is trained for 1000 epochs on an NVIDIA A100-SXM4-40GB, which takes about 62 hours. The training data is split into batches of size 512. While the critic is trained using all batches, the generator and the constrainer networks are trained only every 10th batch,

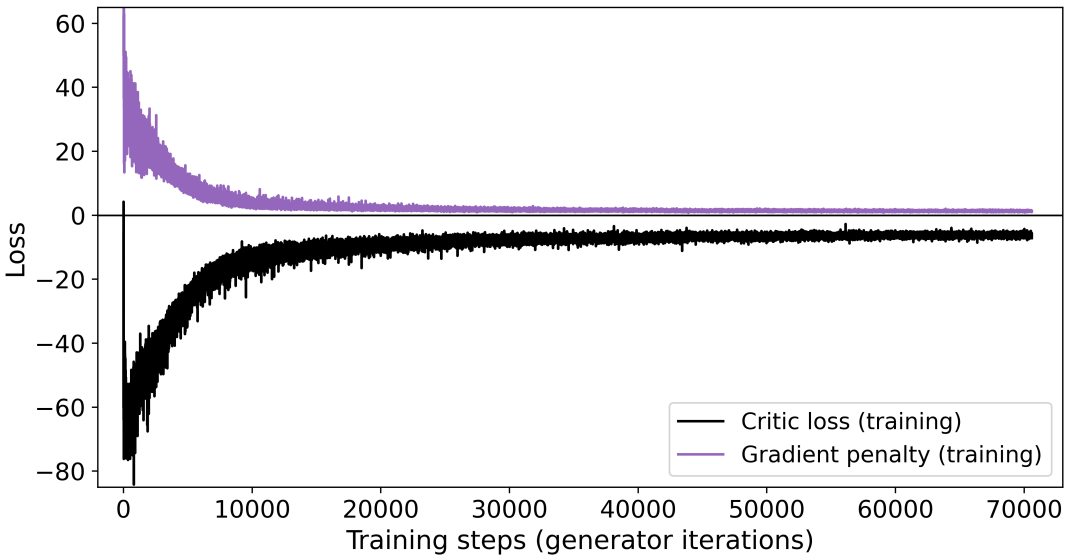


Figure 24: Training loss curves of the critic and the gradient penalty for one training of the framework, which takes 1000 epochs.

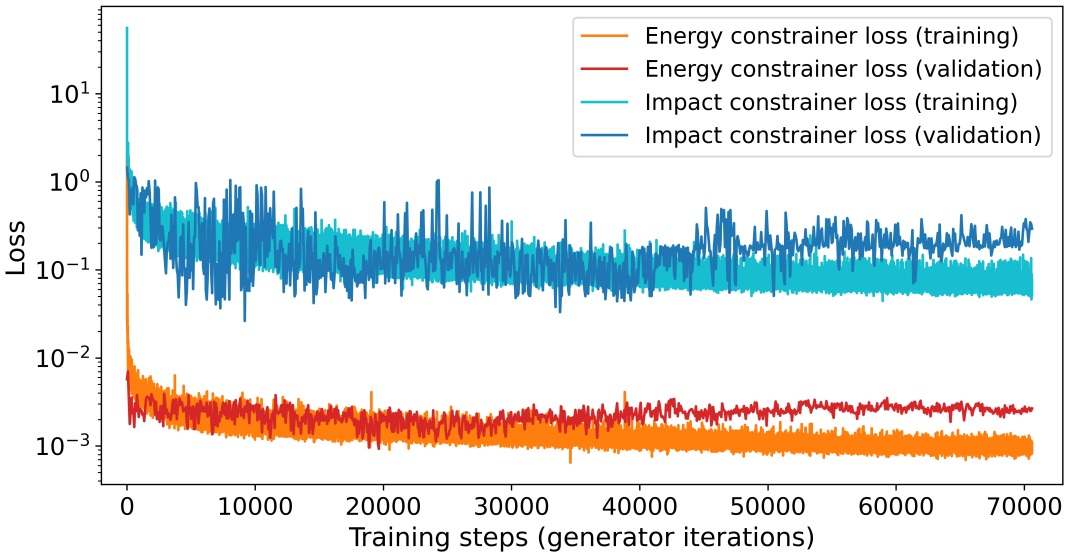


Figure 25: Training and validation loss curves of the energy and impact constrainer for one training of the framework, which takes 1000 epochs.

so $n_{\text{critic}} = 10$. This is done so that the critic can sufficiently approximate the Wasserstein distance. After ten batches, one iteration of the network is finished after updating the generator a single time. The weight of the gradient penalty is set to $\lambda = 10$. The scaling parameters for the auxiliary loss are $\alpha_E = 500$ and $\alpha_I = 50$. All four networks use Adam as the optimiser. The critic and the generator have an initial learning rate of 0.0001 with $\beta_1 = 0.5$ and $\beta_2 = 0.9$. The constrainer networks, however, start training with a learning rate of only

0.001 with $\beta_1 = 0.5$ and $\beta_2 = 0.5$. All learning rates decay roughly one order of magnitude during the training.

Training procedure Here, the training of our framework, shown in Figure 23, is described in detail. Initially, the **critic** (purple) is trained using ten batches of simulated and generated images as follows:

- The simulated images and labels, so the energy and the impact point coordinates are taken from the training data set.
- For the simulation of the generated images, random noise, as well as the physical labels of the training data set smeared with a Gaussian, are taken as input. For the Gaussian smearing, a mean of zero and a standard deviation of 10% for the energy and standard deviations of 10 m for both impact point coordinates are used.
  This approach for generating images is chosen due to the correlation of image, energy and impact point. A random sampling of the physical labels would not consider this relation.
- The simulated and generated images are then given to the critic and evaluated by the network, followed by the estimation of the Wasserstein distance. Moreover, the gradient penalty and the critic loss are calculated, and both are shown in Figure 24.
  The gradients of the network parameters are determined, and the critic parameters are updated with the Adam optimiser accordingly.

After training the critic for ten batches, the critic can describe the Wasserstein distance well enough. Next, the training of the **generator** (orange) and both **constrainers** (yellow) follows. All three networks are trained with the last batch the critic was trained on. The procedure looks as follows:

- Both constrainers are trained by giving them simulated images and the remaining labels, respectively, as input and reconstructing the physical property.
  The loss for both networks, shown in Figure 25, and the gradients are calculated but not yet applied since their current state is still needed for the training of the generator.
- Generated images and labels are input to the critic. The network outputs values used to calculate the default generator loss.
- Furthermore, these inputs are fed to the constrainers, resulting in reconstructed energies and impact points. With these outputs, the auxiliary loss, working as feedback for the generator, is calculated.
- The auxiliary loss is added to the default generator loss, resulting in the

generator loss of our framework. This loss is then used for calculating the gradients of the parameter.

- The parameters of the generator and the constrainer networks are updated using the Adam optimiser and the previously calculated gradients.

One training iteration is finished after the training of these three networks. Once the training has iterated over the training data set, one epoch is finished, and the validation data is used to check the constrainers for overtraining. The validation data set is given to the networks as one without using batches, and the resulting validation loss is shown in Figure 25. Now, everything repeats till the framework goes through the 1000 epochs.

Post-processing The generator uses a post-processing function to overcome minor limitations. In this work, an extended 4/7 cleaning is applied to the generated IACT images, which requires only a small computational effort.

6 Analysis of the image quality of the generated IACT events

After discussing the network training, the quality of the generated IACT images is investigated. First, in Section 6.1, the IACT images generated with our WGAN framework are shown and compared to the images from the simulated data set. The correct implementation of the physical labels – the primary particle energy and the air shower impact point – during the generation process is verified in Section 6.2. Various camera image parameters, including the Hillas parameters and their correlations, are examined in Section 6.3. Lastly, the computational speed-up of IACT image generation achieved by our framework is presented in Section 6.4.

6.1 Generated camera images

Studying the quality of the generated images is essential since it is a central factor of our developed algorithm. For the simulated images and labels, we take the 77330 events from the test set, which is a part of our initial data set. For the generated data set, the same number of images is generated. The images are obtained by inputting noise and the simulated labels into the generator. This approach of label sharing is chosen due to the correlation of image, energy, and impact point coordinates. "In [Figure 26], we present four representative types

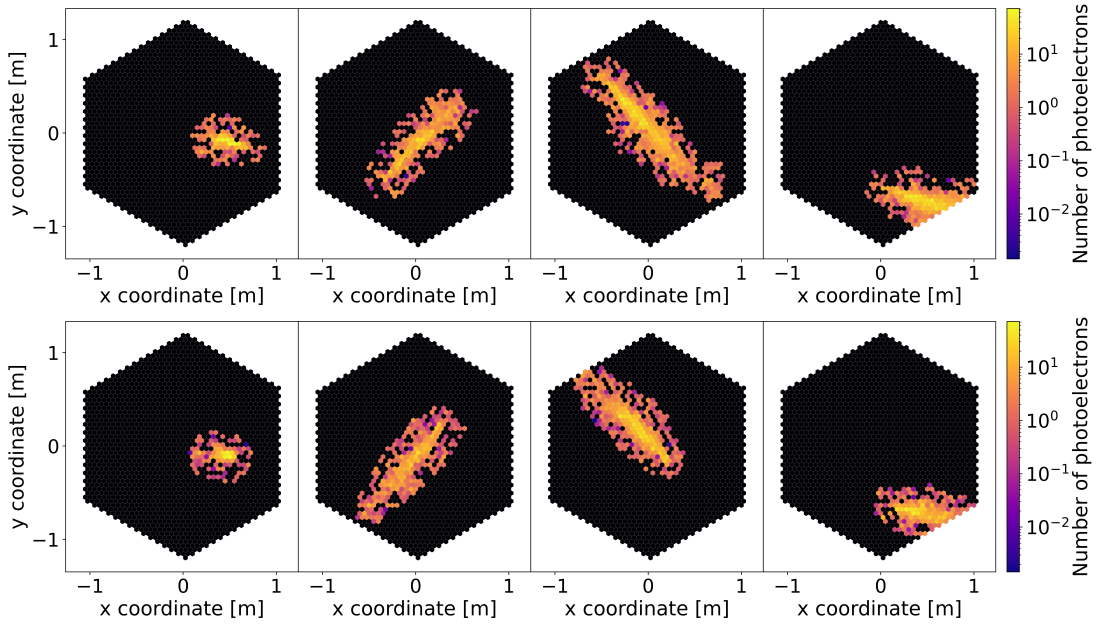


Figure 26: Four IACT images from the simulated data set (top) and four images from the generated data set (bottom). The images are handpicked to show their similarity. Taken from [1].

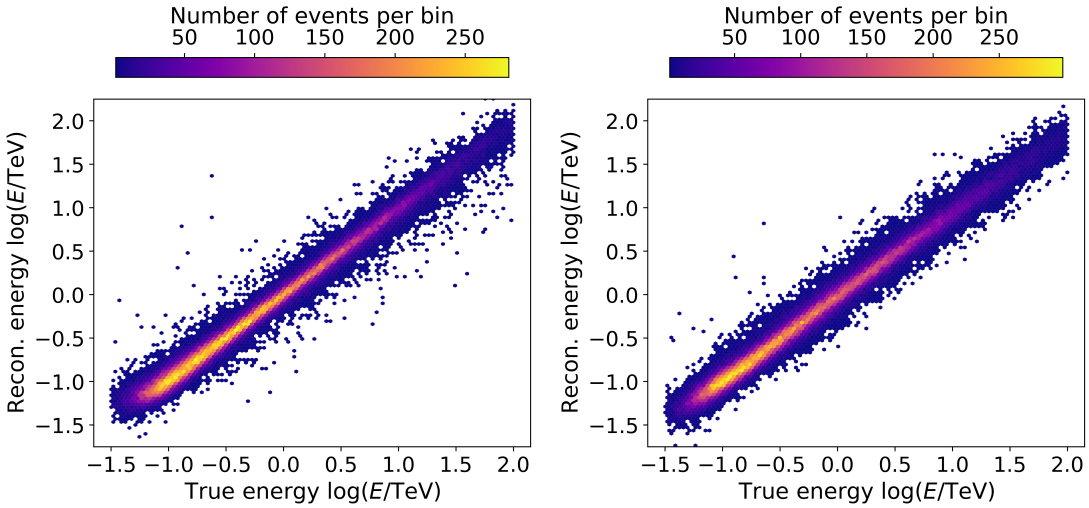
of showers in the data set — specifically chosen to highlight the different shapes typically represented in IACT images — and compare them to similar images of the generated samples, whereas as generated images, the sample with the smallest pixel-wise MSE w.r.t. the test image was selected out of the generated data set. In the first column, the signals in both images resemble a circular shape, corresponding to a shower developing in parallel to the telescope's line of sight with a close-by impact point. Images with the typical elliptical-shaped signals can be seen in the second and third columns. A representative image for so-called truncated images, in which the Cherenkov light falls to the edge of the telescope's field of view, is shown in the last column. These samples represent a large variety of images, which could be successfully generated using the WGAN. Additionally, for an unbiased comparison, randomly chosen samples from the test data set and the generated samples can be found in the Appendix A in Figure 41." Elflein et al. [1].

6.2 Verification of physics implementation

"Verifying the correct implementation of physical properties in the generated images is essential. For the IACT image generation in this work, two physical labels, namely the primary particle energy E and the air shower impact point $I = (I_x, I_y)$, are used. For generating new images, randomly sampled noise and these physical labels are input into the generator. During training, the energy and impact constrainer networks — trained to reconstruct the respective label using simulated data only — are utilized to enforce the label representation in the final image. Using the constrainer networks, the physical labels of the generated images can be reconstructed. Comparing the reconstructed labels with the labels initially given into the generator provides information about the implementation of the physical labels into the generated samples." Elflein et al. [1]. In Section 6.2.1 and Section 6.2.2, the reconstructions of the primary particle energy and the air shower impact point are investigated.

6.2.1 Primary particle energy

The energy of the primary particles in our final data set ranges from $10^{-1.5}$ TeV to 10^{2} TeV. "The reconstruction performance of the primary particle energy for the simulated test set and the generated data sets is shown in [Figure 27]. The reconstruction works well over the whole energy, showing that the energy is correctly represented in the generated IACT images. In direct comparison, the energy of the generated data set shows a slightly better reconstruction performance than the simulated test data, which is mainly visible through the better energy resolution. Considering the generator receives feedback from the energy constrainer for the enforced implementation of the energy label, this indicates that the generator is likely underestimating the fluctuations for a given energy, even being able to generate events from the bulk of the distribution successfully." Elflein et al. [1].



(a) **(b)**

Figure 27: Energy reconstruction of the simulated (left) and generated (right) images using the energy constrainer network. Taken from [1].

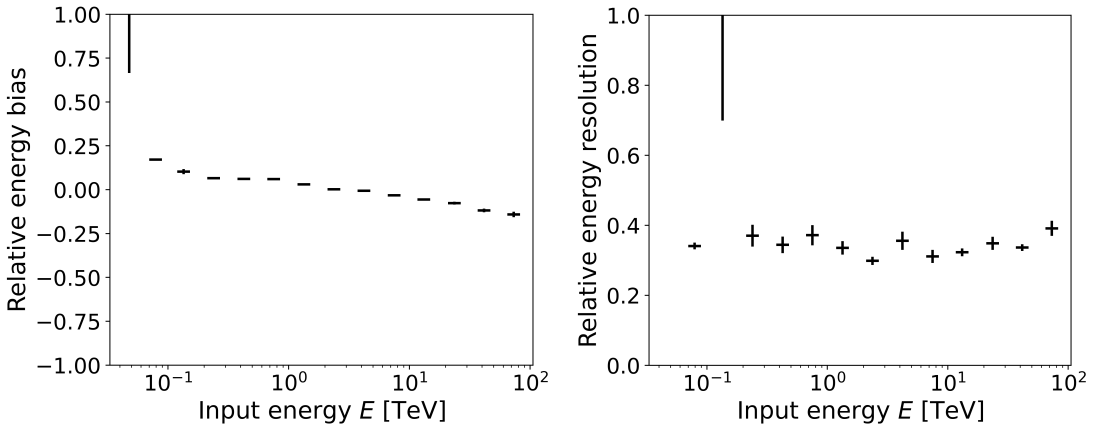
For a more quantitative evaluation of the energy reconstruction, we directly investigate the relative energy bias and resolution. These parameters are shown in Figure 28 for the simulated data set and in Figure 29 for the generated data set. In both cases, the data is split into 14 bins. The relative energy bias is the mean of the difference between reconstructed energy E_{rec} and true energy E_{true} over the true energy in one bin. Thus, for a bin with N_{bin} data points, it is defined as follows:

$$E_{\text{bias}} = \left(\sum_{i=1}^{N_{\text{bin}}} \frac{E_{\text{rec, i}} - E_{\text{true, i}}}{E_{\text{true, i}}} \right) / N_{\text{bin}} \tag{32}$$

The trend for the bias is very similar for both data sets. The bias is largest at the low energies and decreases quickly, going to slightly higher energies. After about 100 GeV, the bias decreases only slowly. Except for the first bin, the relative energy bias is below 0.25 and above -0.50 in both cases. Looking at the relative energy resolution

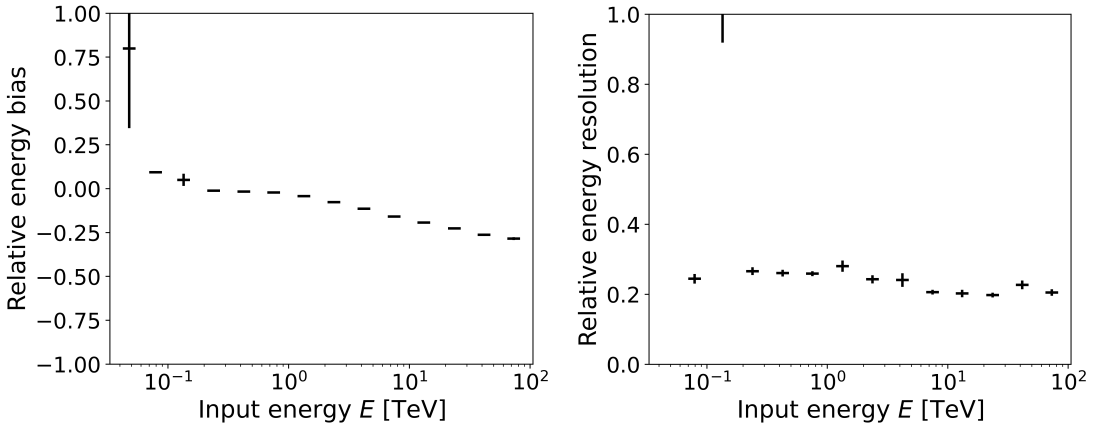
$$E_{\text{resolution}} = \sqrt{\sum_{i=1}^{N_{\text{bin}}} \left(\frac{E_{\text{rec, i}} - E_{\text{true, i}}}{E_{\text{true, i}}} - E_{\text{bias}} \right)^2 / N_{\text{bin}}} \tag{33}$$

which is the standard deviation of the energy reconstruction difference over the true energy in one bin, more noticeable differences can be spotted. On average, the resolution of the generated data set is between 0.2 and 0.3, which is better than for the simulated data set since the resolution is between 0.3 and 0.4. Furthermore, there are bins in which the resolution explodes and is higher than 1. This is caused by the outlier images, which were already visible in Figure 27.



(a) **(b)**

Figure 28: Relative energy reconstruction (a) bias and (b) resolution of the simulated data set.



(a) **(b)**

Figure 29: Relative energy reconstruction (a) bias and (b) resolution of the generated data set.

These outliers [1] and high values for some bins in the simulated case are mostly caused by the reconstruction of non-gamma images. In Figure 30, two of these outliers are shown, and in both images, there is a ring-like signal present caused by muons [61]. Since we want to generate gamma images only, in an optimal case, all these outlier images would be filtered out from the data set before using it for training. Even though we did not exclude any outliers from the training in the end, we want to mention it to emphasise its importance for optimising IACT image generation using an ML approach. Our approach for finding outlier images in the data set is rather simple. First, the training is carried out for once normally. Then, the energy constrainer network is used to reconstruct the energy of every single image from all three data sets. Afterwards, the energy

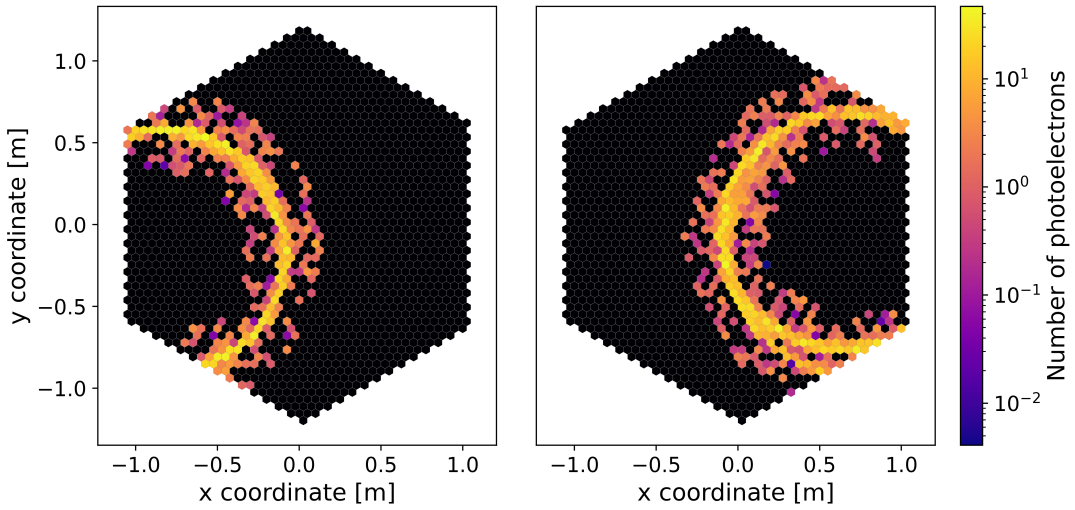


Figure 30: Two IACT outlier images containing a ring-like signal from a muon.

reconstruction difference is calculated by subtracting the true energy from the reconstructed energy and taking the absolute value of the result. Every image with a reconstruction difference above a certain threshold is marked as an outlier run. The threshold, which affects the resulting number of outliers, can be chosen freely. As mentioned above, such outliers are filtered out of the data set in an optimal scenario before starting the training.

Overall, ignoring the energy reconstruction of the non-gamma images, the reconstruction of the primary particle energy works well, indicating that the conditioned energy label is implemented accordingly into the IACT images during their generation.

6.2.2 Air shower impact point

Besides the primary particle energy, the impact point of the air shower, so the location of it hitting the ground, is implemented during the image generation. Both the x and y coordinates of the impact point go from about $-1000\,\mathrm{m}$ to $1000\,\mathrm{m}$ with increasing occurrence towards the centre, which lies at the location of the CT5 telescope. “In [Figure 31], the reconstruction of the air shower impact point for simulated (left) and generated (right) images are shown using the constrainer network and compared to the MC truth (middle). For both cases, a larger density of impact points close to the telescope is visible due to the fact that only high-energy showers trigger the telescope from further distances. The visible ring with a diameter of roughly 150 m represents the typical radius of a Cherenkov cone for a vertical shower at the H.E.S.S. site. Further, more events are reconstructed in the array’s center for the simulation and the generated images. This visualizes a slight bias of the reconstruction, causing a more likely estimate in the center of the array if the constrainer network is not able to infer which direction to reconstruct the respective impact. This effect is commonly referred to as *regression to the*

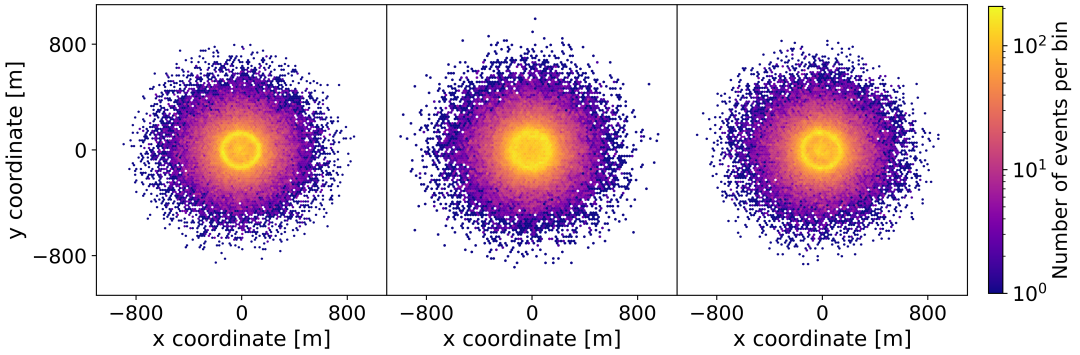


Figure 31: Reconstruction of the air shower impact point for simulated (left) and generated (right) images. The true impact points are shown in the middle. Taken from [1].

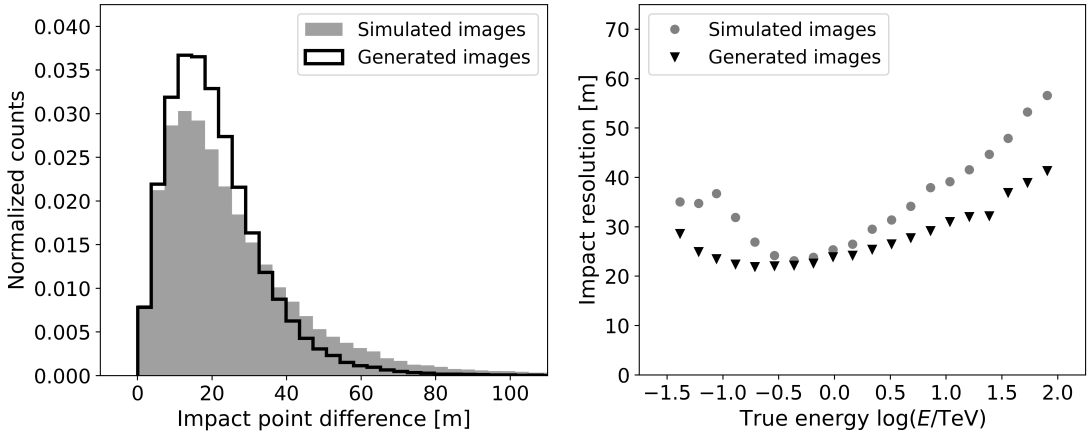


Figure 32: (a) Distributions of the distance difference between the reconstructed and true impact point for simulated (grey) and generated (black) images. (b) Resolution of the impact point difference for simulated (grey circles) and generated (black triangles) images.

mean. The finding that this effect is slightly more pronounced for simulated events likely indicates, similar as found for the energy, that the WGAN is not able to precisely model all fluctuations w.r.t. the input point in the image." Elflein et al. [1].

A more detailed and quantitative investigation of the impact point reconstruction is depicted in Figure 32. The distribution for the impact point difference

$$I_{\text{diff.}} = \sqrt{(x_{\text{rec.}} - x_{\text{true}})^2 + (y_{\text{rec.}} - y_{\text{true}})^2} \tag{34}$$

between the reconstructed impact point $I_{\text{rec.}} = (x_{\text{rec.}}, y_{\text{rec.}})$ and true impact point $I_{\text{true}} = (x_{\text{true}}, y_{\text{true}})$ is shown in Figure 32a. It is evident that both distributions follow a similar trend of having a peak around $10-20\,\text{m}$ and decreasing occurrences

going to lower and higher values. Nevertheless, the distribution for the simulated images has a lower peak and more occurrences at higher values, which shows that the reconstruction of the impact point is slightly worse for this data set. This trend is also seen for the impact point resolution, shown in Figure 32b, which tells us how well the impact point is reconstructed. Besides the data points matching at energies of high GeV and low TeV, most of the data points of the generated data set are lower, meaning the impact point resolution is on an average lower of this data set.

Despite these few differences present for the impact point reconstruction, the overall similarity shows that the air shower impact points are implemented during the image generation.

6.3 Investigation of camera image parameters

After visually comparing the simulated and generated images and verifying the implementation of the physical labels during the generation process, a last study of investigating camera image parameters is carried out to prove the successful generation of realistic IACT images. In Section 6.3.1, several pixel parameters are examined. The Hillas parameters, commonly used in event reconstruction as explained in Section 2.4, are investigated in Section 6.3.2. The correlations of the Hillas parameters with themselves and the physical labels are discussed in Section 6.3.3

6.3.1 Pixel parameters

"The pixel occupancy, i.e., how often a pixel holds signals averaged over the whole dataset, is shown in [Figure 33], where the color bar ranges from -5 to 5 σ deviation (statistical). Overall, the pixel occupancies of the simulated and generated data sets are very similar and show the same behavior of a smaller occupancy at the edges of the camera and a larger occupancy in the central area of the camera. Combined with the finding of the image quality. These findings show that our generative algorithm does not suffer from mode collapsing, as present in previous works [62, 63]. The three vertically outermost pixels in the simulated image, as well as the mount pixels, i.e., the three triangularly-arranged pixels in the center of the camera, are not simulated in its current design. In this approach, we, therefore, are not considering the pixels and they are masked to zero in the last layer of the generator network. Comparing both plots, the generated image is more dispersed, which is especially shown in the middle of the camera. However, the fluctuations are, in comparison to the overall gradient from the outside to the inside of the camera, small.

The distribution of triggered pixels per event is shown in [Figure 34a]. For this and all the following plots, we use the same style, in which the simulation is shown as a colored grey histogram, and the generated histogram is shown as a solid black line. The distributions of the number of signal pixels per image follow the same trend for both simulated and generated images and show a good

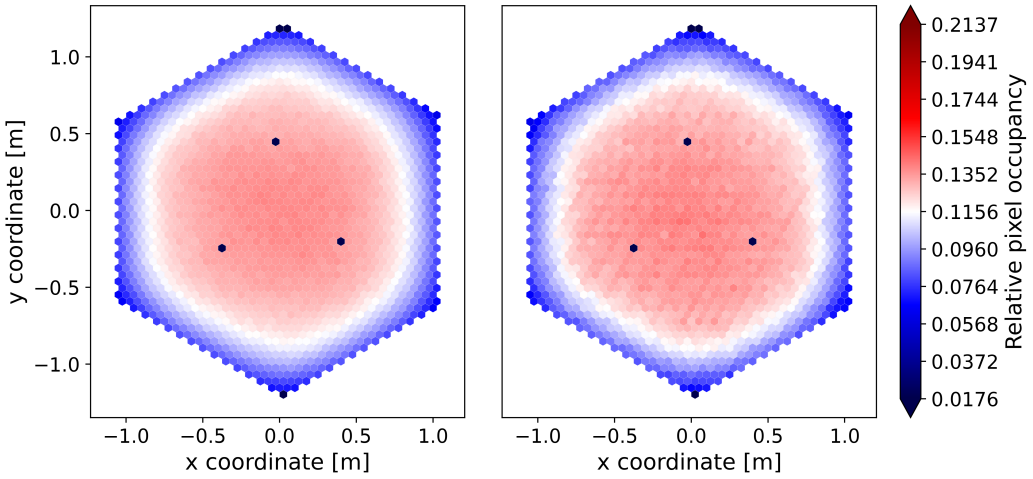
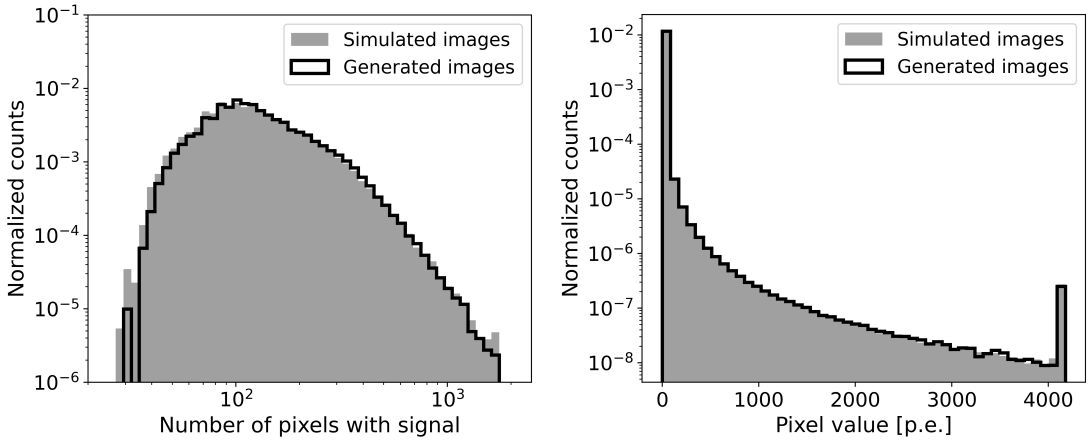


Figure 33: Illustration of the relative pixel occupancy for the simulated data set (left) and the generated data set (right). The colorbar is chosen to show the 5σ range of the mean occupancy value of the simulated occupancy image. Taken from [1].



(a) **(b)**

Figure 34: Distributions of the (a) number of pixels per image containing any signal and (b) pixel values of the whole data set for generated (black) and simulated (grey) events. Taken from [1].

agreement. The most occurring number of signal pixels is around 100, with a decreasing trend to lower and higher values. The cut-off of the distributions at 1,758 is due to the number of pixels of the CT5 camera. The most noticeable but still moderate difference between the distributions can be seen for images with fewer signal pixels, where the generated data set features fewer images with few signal pixels. Nevertheless, the overall match is good.

In [Figure 34b], the distributions of pixel values are compared for both data sets. Events without signals are included in the smallest bin. Both distributions

match well over the whole range up to the limit at ~ 4200 p.e. caused by saturation. Overall, we find a good agreement between the simulated test and the generated data set for the studied low-level parameters over the entire range, which covers several orders of magnitude." Elflein et al. [1].

6.3.2 Hillas parameters

"To derive the Hillas parameters, as a first step, image cleaning is applied to remove signal pixels containing a large fraction of the night sky background. Since we used CT5, we are applying the same 9/16 cleaning procedure as performed by the H.E.S.S. collaboration. We further set a threshold cut (after cleaning) of 250 p.e. and discard events below the threshold. In the second step, the Hillas parameters are derived using ctapipe [6], modeling the Cherenkov light in the camera after cleaning as an ellipsis using the position and values of the pixels containing a signal. In this analysis, we focus on the most important Hillas parameters: intensity, length, width, radial coordinate, polar coordinate, and rotation angle. An illustration of these six parameters can be found in Figure 10.

Intensity The Hillas intensity (size), shown in [Figure 35a] for both data sets, is defined by integrating the remaining pixel intensities after cleaning and strongly depends on the particle energy and the distance of the telescope to the shower core. Both low particle energies and shower cores far away from the telescope typically result in small image sizes due to the little Cherenkov light being detected. On the contrary, large image sizes are obtained when the primary particle has high energy or the shower core is close to the telescope. The cut-off at low intensities is caused by the size cut described above. With a decreasing trend to higher values, the sizes reach up to 10^6 p.e.. The generated and simulated image distribution shows a very good agreement over the whole range, comprising more than three orders of magnitude. Only at the highest intensities, small deviations are visible. These differences are, however, small given the low statistics at the tail of the distribution.

Length and width of the ellipses The Hillas length L and width W characterize the shape of the signal and are the standard deviations of the measured signal along the major and minor axis of the ellipsis on the camera image, respectively; i.e., they are estimated by taking the square root of the two eigenvalues obtained after estimating the two largest components using a principal component analysis of the image. The distributions for the Hillas length and width match for the simulated and generated images quite well are shown in [Figure 35b] and [Figure 35c]. Whereas the bulk of the distribution can be precisely reproduced by the WGAN, at the edges, small differences remain. These are, however, statistically not very significant, indicating that the WGAN is able to correctly reproduce the shapes of gamma-ray images.

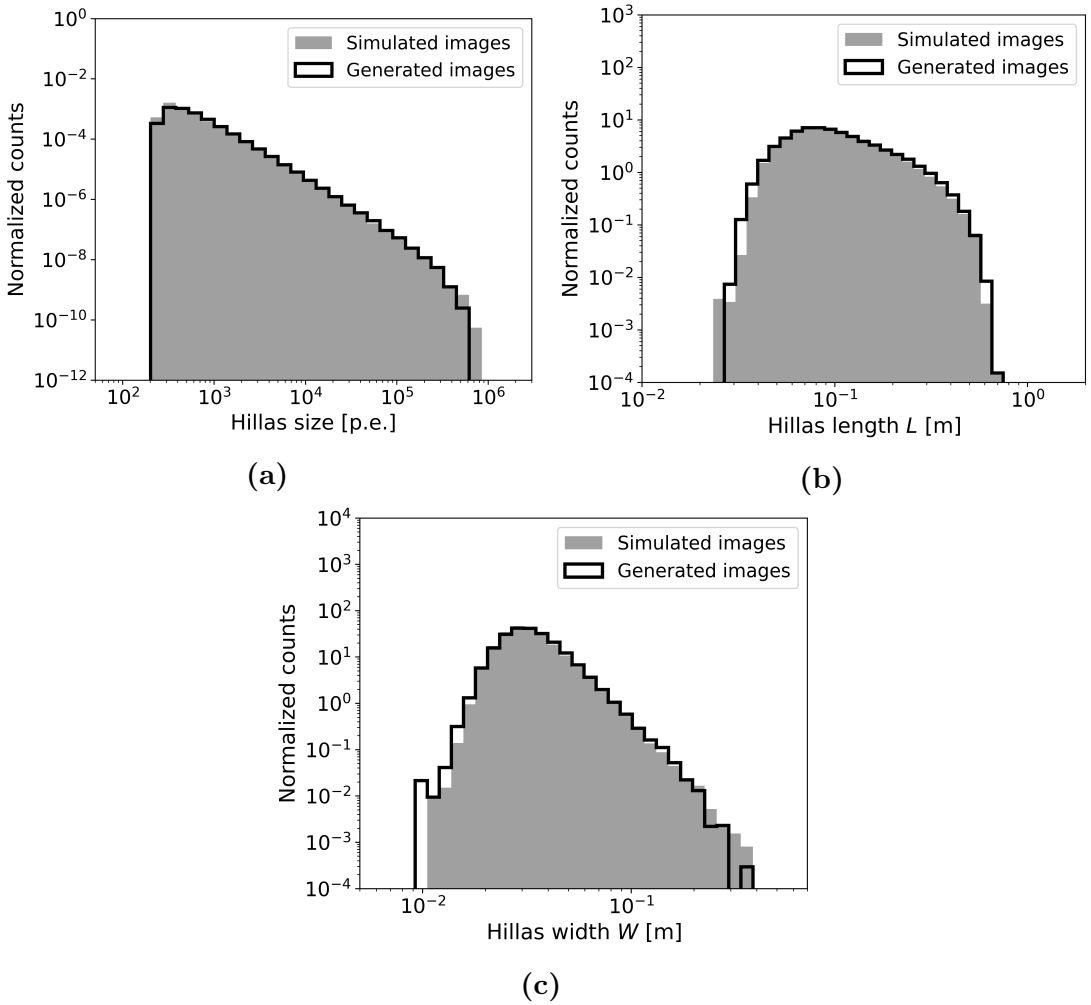


Figure 35: Comparison of the Hillas parameters (a) size, (b) length and (c) width for the generated (black) and simulated (grey) events. The images of both data sets are cleaned with 9/16 tail cuts cleaning. Taken from [1].

Rotation and position of the ellipses The radial coordinate r and polar coordinate ϕ describe the location of the center of the ellipsis. While the radial coordinate, often referred to as local distance, is the distance from the camera center to the ellipsis center, the radial coordinate defines the angle between the $x-$axis of the camera and the ellipsis center. The parameter distributions for both data sets are shown in [Figure 36a] and [Figure 36b], reaching from 0 m, ellipsis center at the camera center to roughly 1.2 m the edge of the telescope. Both distributions show a very good agreement, denoting that the whole camera frame is adequately covered by the generated showers of the WGAN, well in line with the finding of a realistic occupancy (cf. [Figure 33]).

Additionally, the distributions of the rotation angle ψ of the ellipsis match well, as shown in Figure 36c. The coordinate is defined as the angle between the $x-$axis on the camera and the major axis of the ellipsis and, therefore, features

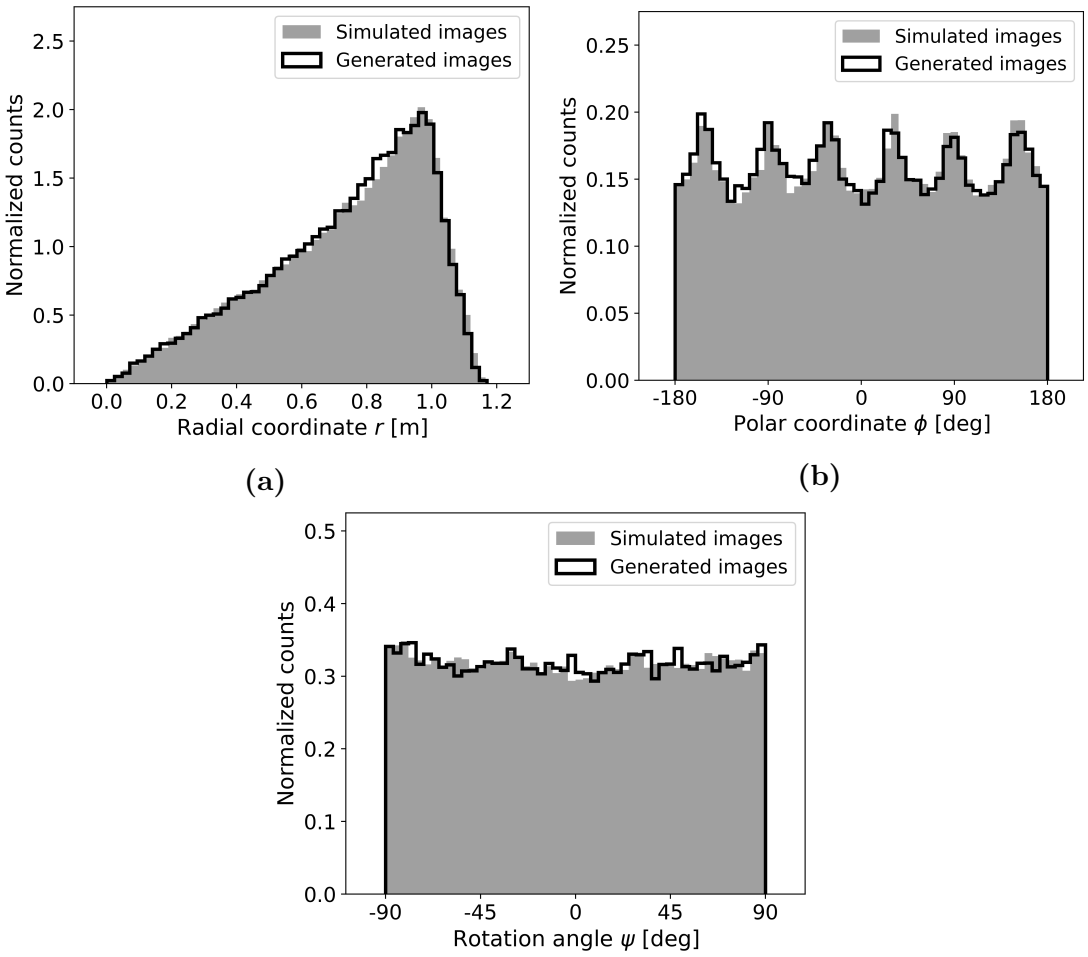


Figure 36: Comparison of the Hillas parameters (a) radial coordinate r, (b) polar coordinate ϕ and (c) rotation angle ψ for the generated (black) and simulated (grey) events. The images of both data sets are cleaned with 9/16 tail cut cleaning. Taken from [1].

six maxima due to the hexagonal camera design of FlashCam. Combined with the previous observations, this finding indicates that many different shower geometries are generated by the algorithm, excluding mode collapsing present in previous approaches [63].” Elflein et al. [1].

X and Y coordinate The X and Y coordinates of the Hillas centroid are shown Figure 37a and Figure 37b. Since both coordinates contain the same information as the radial and polar coordinates, their distributions also match very well for the simulated and generated data set, and there are only some small differences. The peaks in the distributions correspond to the corners of the camera.

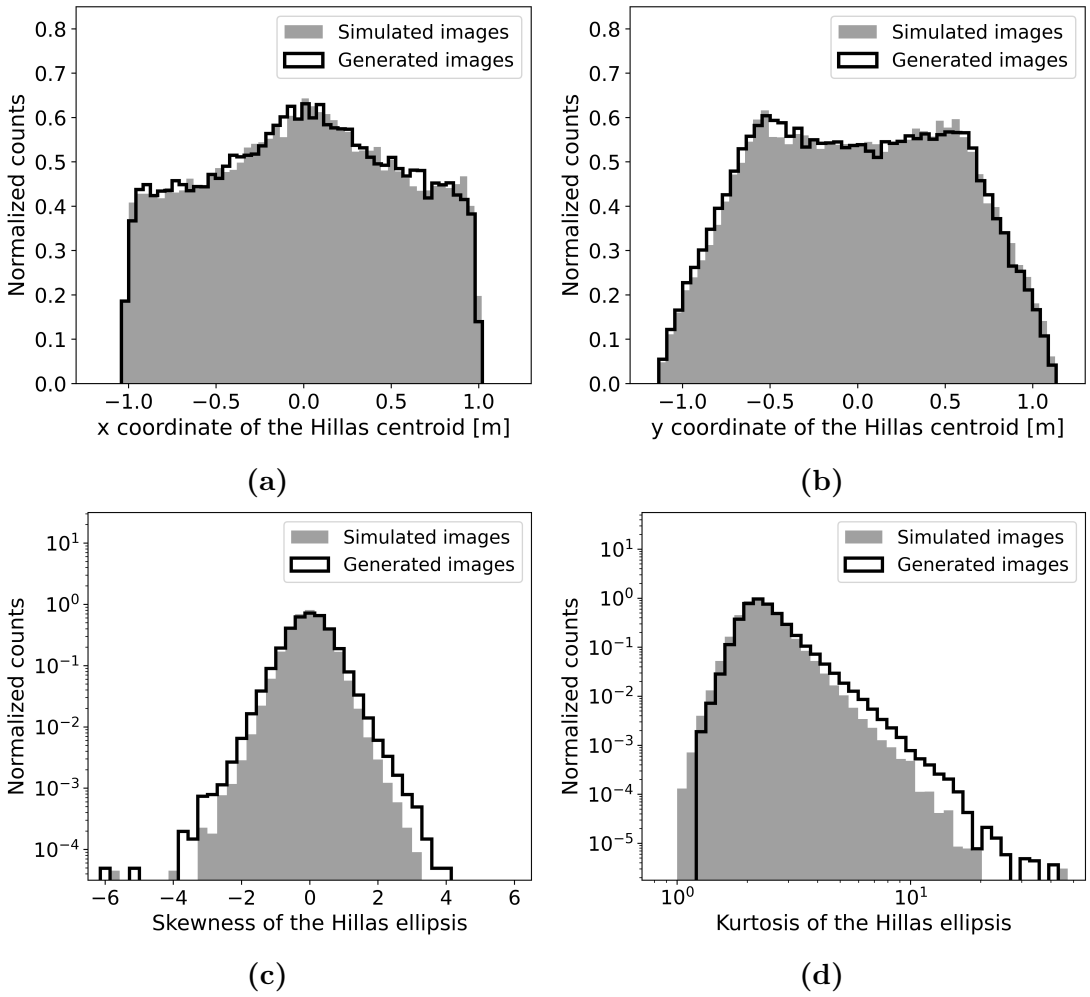
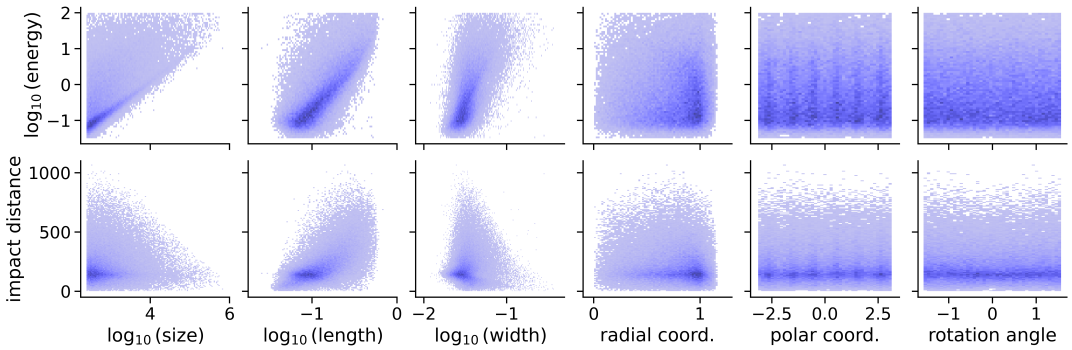
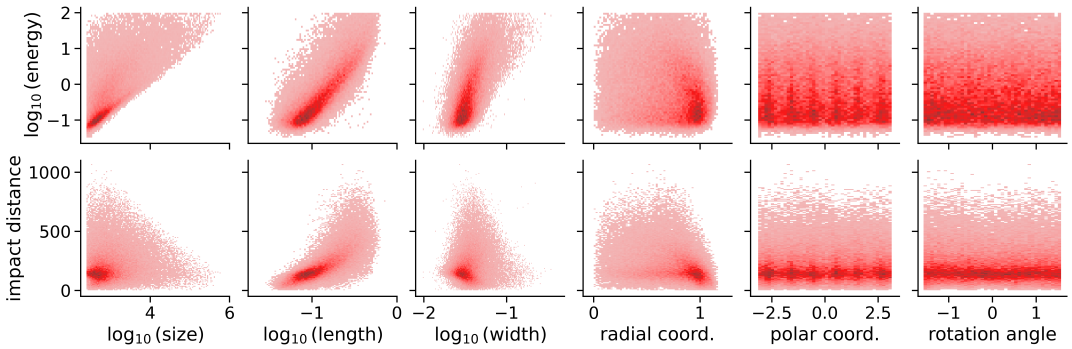


Figure 37: Comparison of the Hillas parameters (a) x coordinate, (b) y coordinate, (c) skewness, and (d) kurtosis of the Hillas centroid for the generated (black) and simulated (grey) events. The images of both data sets are cleaned with 9/16 tail cut cleaning.

Skewness and kurtosis The skewness and the kurtosis, shown in Figure 37c and Figure 37d, give information about the symmetry and tailedness of the Hillas ellipsis [6]. Even though the correct distribution trend for both parameters for the generated data set is present, the difference in the distribution of the simulated data set is more significant. This comes from the fact that the skewness and the kurtosis are calculated using the third-order moments of the Hillas parameterization [36]. This makes the representation of the parameter in the generated images more difficult. However, again, the distributions of these parameters match well for simulated and generated images.



(a)



(b)

Figure 38: Correlations between the physical labels (E and impact distance) and the Hillas parameters. The correlations of the simulated data are shown at the top in blue, while the correlations for the generated data are shown at the bottom in red. Taken from [1].

6.3.3 Correlation of Hillas parameters

"Lastly, the correlation of the high-level parameters, so the energy, the impact point, and the Hillas parameters, is investigated to verify that the generator is able to reproduce the most complex relations regarding IACT images. Since the images and their characteristics mostly depend on the distance of the telescope to the impact point, we investigate $I = \sqrt{I_x^2 + I_y^2}$ hereafter.

After studying the implementation of the physics conditions during the image generation using the two constrainer networks as discussed in [Section 6.2] by reconstructing the labels, we now investigate their correlation with high-level parameters. The corresponding correlations for simulated and generated images are shown in [Figure 38a] and -[Figure 38b], in which simulated events are shown in blue and events generated using the WGAN in red. When comparing both Figures, the inter-dependencies of each of the various Hillas parameters are very diverse. However, these very different correlations are qualitatively similar in both the simulated and generated data sets. In particular, the bulk of the distribution is well reproduced for all cases. Differences remain, especially for the length and

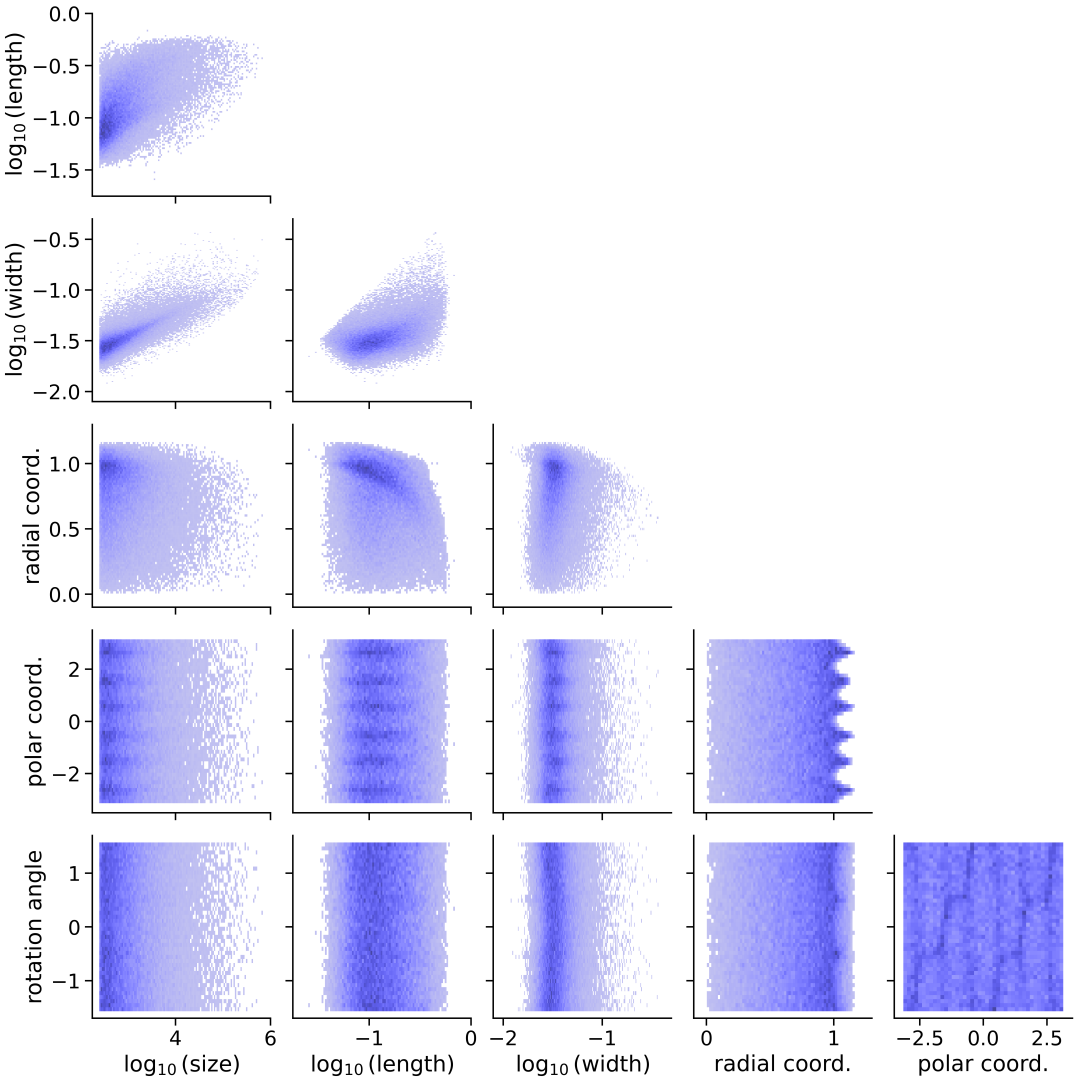


Figure 39: Correlations between the Hillas parameters for the simulated data set. Taken from [1].

the local distance. These identified differences might be related to the small but still noteworthy performance difference discussed in [Section 6.2]. Further changes in the training strategy, e.g., by enhancing the quality of the feedback of the constrainer networks using spectral normalization [64], could possibly improve the performance. Nonetheless, the overall correlations are reproduced in much detail.

Next, the correlations between the Hillas parameters themselves are examined. The parameter correlations of the simulated images are shown in [Figure 39] in blue, while the correlations using the images generated with the WGAN are shown in red in [Figure 40]. The overall correlations seem to be very well-produced using the WGAN, and the covered phase space matches, i.e., no outliers are visible in the generated events. Also, in the case where no strong correlations are expected, e.g., for the radial coordinates and the rotation angle, the WGAN does

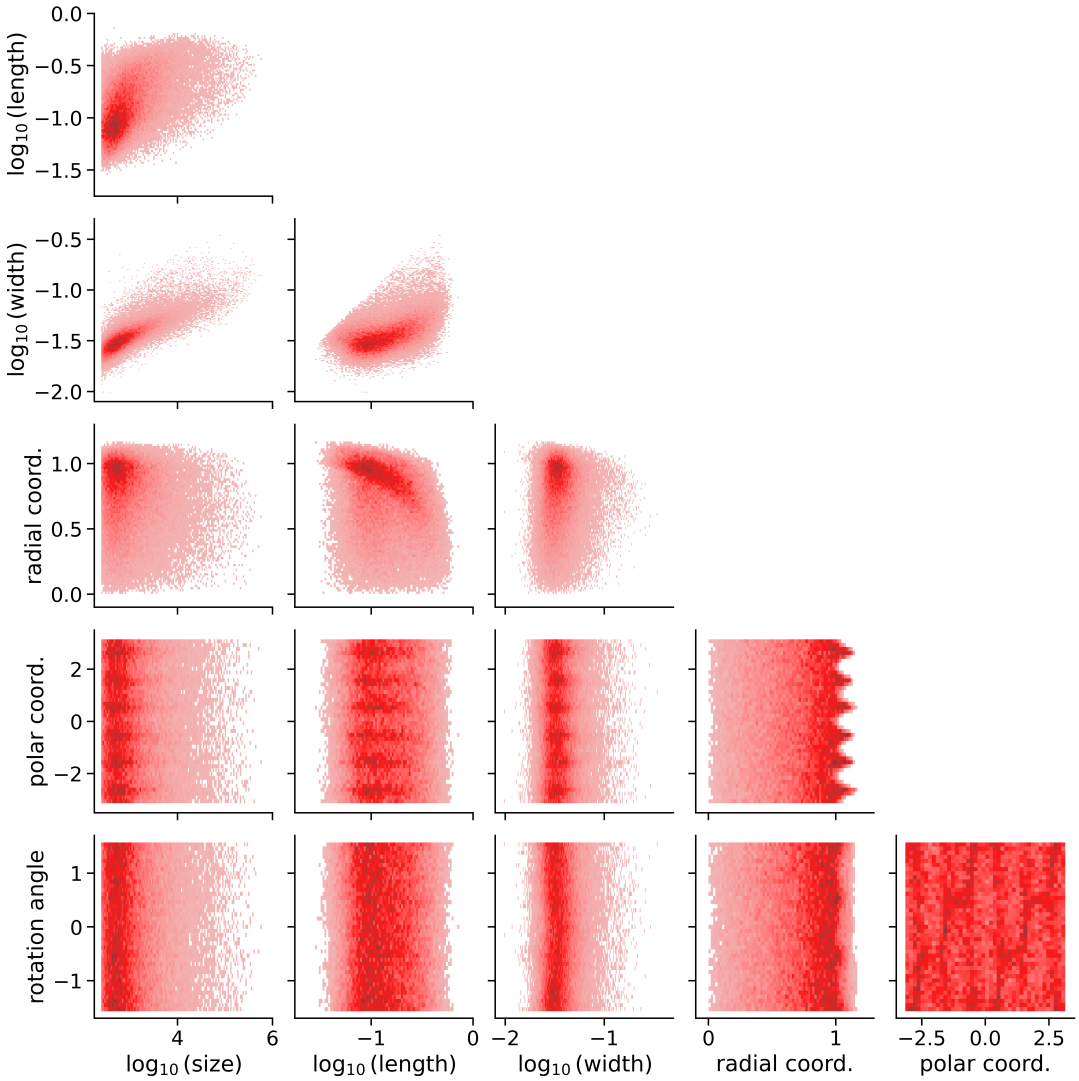


Figure 40: Correlations between the Hillas parameters for the generated data set. Taken from [1].

not induce artificial correlations not present in the simulated test data set. Only very minor differences remain, e.g., in the correlation between length and size.

Conclusively, we have demonstrated that the WGAN is not only able to reproduce the 1-dimensional distributions of low- and high-level parameters with good quality but also shows high fidelity with respect to their correlations. This is a significant improvement in comparison to previous work since not only IACT images with a high resolution of more than 1,500 pixels without mode collapsing were generated, but also the generated images were comprehensively analyzed using low-level and image parameters and their correlation. The finding that IACT images can be generated with high fidelity using deep generative models opens up promising opportunities for fast and efficient simulations in the era of CTA." Elflein et al. [1].

Method	Hardware	Time	**Speed-up**
CORSIKA & `sim_telarray`	Intel Xeon Gold 6230	70h	–
WGAN framework	AMD EPYC 7713 Milan	86.06 s	**x 2930**
WGAN framework	NVIDIA A100-SXM4-40GB	2.34 s	**x 108000**

Table 5: Computational time required for the generation of 100,000 IACT images using different methods. Adapted from [1].

6.4 Computational speed-up

The computational speed-up of the image generation with the presented deep learning approach [1] is the major advantage over the standard simulations. The simulation time of 100,000 IACT images is compared for three different methods in Table 5. The first one is the standard simulation with CORSIKA and `sim_telarray`, which takes about 70 h [55] on a CPU. The other two methods are the generation of IACT images with the WGAN framework, once on a CPU and once on a GPU. The generation time of 100,000 images on a CPU takes about 86 s, which is almost 3000 times faster than for the standard MC simulation. The generation of images with our framework on a GPU is the fastest method due to the parallelised computations. It takes only $2 - 3$ s to generate 100,000 images, which is a speed-up of more than five orders of magnitude compared to the standard simulation. This comparison shows that the use of an ML-based approach for IACT image generation can drastically decrease the overall simulation time .

7 Summary and outlook

Learning more about the highest energy processes in various astronomical sources is crucial to better understand our universe. Gamma rays [2], highly energetic photons, are produced close to these sources and travel undeflected to Earth. After penetrating the atmosphere, they induce a particle cascade, resulting in secondary particles moving down to the surface. These new particles emit Cherenkov light, that ground-based instruments like IACTs detect. The cameras of these telescopes image the air shower Cherenkov light, resulting in IACT images. Analysing the signal in these images provides information on the primary gamma ray.

Simulations are essential to fully reconstruct IACT events and get a comprehensive understanding of the instrument. However, running such Monte Carlo simulations is computationally intensive and takes a long time to run (usually days for running a small library). Adopting deep learning techniques for enabling the acceleration of these simulations by augmenting a small simulation library.

In this work, IACT images from the H.E.S.S. [3] CT5 telescope were generated using a deep-learning-based approach and Monte-Carlo simulated data. The model used to generate IACT events is a conditional Wasserstein Generative Adversarial Network (WGAN) [52, 50]. The framework developed within this thesis consists of four different neural networks. Two of the networks, the generator and the critic, are used to generate the image. In contrast, the two remaining networks enforce the physical labels, the primary particle energy and the air shower impact point, into the images. After training the framework, 100 k IACT images can be generated in about 2.34 s (86.06 s) seconds on the GPU (CPU), which is roughly 108,000 (2930) times faster than the MC simulation code.

The quality of the successful generation of IACT images was verified in detail. First, the visual similarity between the simulated and generated images was investigated by inspecting typical IACT image characteristics. The correct and accurate implementation of the physical properties – the primary particle energy and the air shower impact point – during the image generation was verified by reconstructing the parameters with the constrainer networks and by performing correlation studies. Moreover, several different camera image parameters, including the Hillas parameters [36], which are used for event reconstruction, were analysed. While small differences are still present, the overall distributions of these parameters for simulated and generated images match very well. The results showed that the generated IACT images are very similar to the simulated images on a visual and physical level. Overall, the developed machine-learning approach manifested to be a promising technique for the ultra-fast generation of IACT images, reaching acceleration factors of up to 10^5 without losing much quality.

Building on this work, the next step after generating images from a single telescope is the generation of event images from an array of telescopes. Even though the generation of images works similarly, the complex part is the implementation of stereoscopy, which has to be represented in all images. Furthermore, instead of gamma images, the generation of images resulting from cosmic ray-induced air

showers is another step. This could lower the computing costs of CTA significantly since a large number of hadronic showers is needed to estimate the background rejection precisely. Another enhancement would be the separate generation of air showers and the instrument response. This would enable a more flexible and cost-efficient simulation and make the algorithm valuable outside the IACT community, where air showers need to be simulated.

Continued exploration of generative algorithms in the field of astroparticle physics, thus, offers great potential to accelerate event simulations in the long term, making them fast, sustainable, and cost-efficient.

A Appendix

A.1 Simulated and generated IACT images

16 additional IACT images are shown in Figure 41, with eight being from the simulated data set and eight from the generated data set. All images are randomly chosen to give an unbiased view of the images generated in this work.

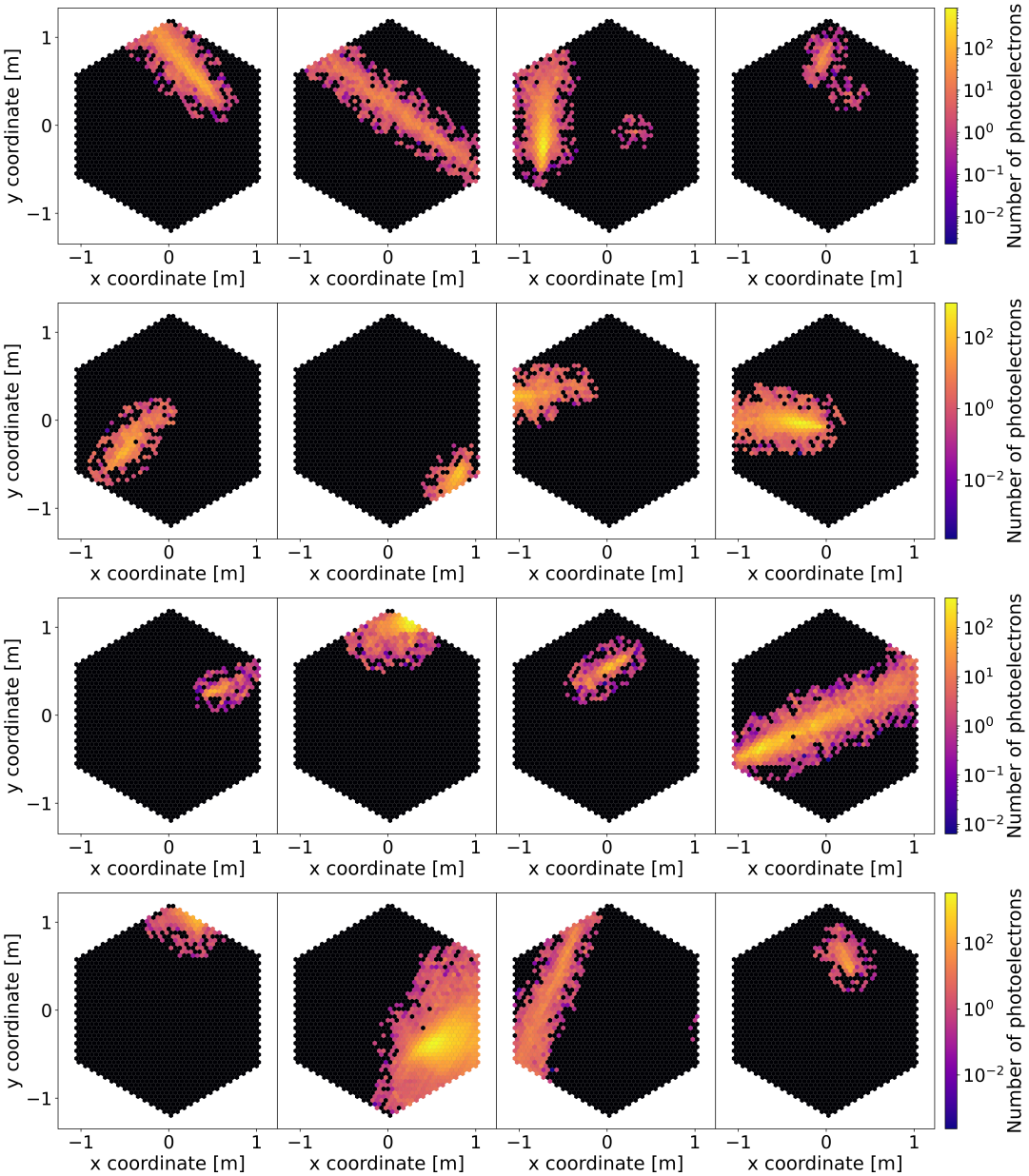


Figure 41: 16 different IACT images. Eight IACT images randomly chosen from the simulated data set are shown in the two top rows. The two bottom rows contain eight random IACT images from the generated data set. Taken from [1].

Bibliography

[1] C. Elflein, S. Funk, and J. Glombitza. „Ultra-fast generation of Air Shower Images for Imaging Air Cherenkov Telescopes using Generative Adversarial Networks“. In: (2023). Prepared for submission to JINST.

[2] D. Bose, V. R. Chitnis, P. Majumdar, and B. S. Acharya. „Ground-based gamma-ray astronomy: history and development of techniques“. In: *The European Physical Journal Special Topics* 231.1 (Dec. 2021), pp. 3–26. DOI: 10.1140/epjs/s11734-021-00396-3. URL: https://doi.org/10.1140%2Fepjs%2Fs11734-021-00396-3.

[3] H.E.S.S. website. *H.E.S.S. - High Energy Stereoscopic System*. [Online; accessed 15-September-2023]. 2020. URL: https://www.mpi-hd.mpg.de/HESS/.

[4] D. Heck, J. Knapp, J. N. Capdevielle, G. Schatz, T. Thouw, et al. „CORSIKA: A Monte Carlo code to simulate extensive air showers“. In: *Report fzka* 6019.11 (1998).

[5] K. Bernlöhr. *sim_telarray*. Accessed: 08.09.2023. May 2022. URL: https://www.mpi-hd.mpg.de/hfm/~bernlohr/sim_telarray/Documentation/sim_telarray_refman.pdf.

[6] M. Nöthe, K. Kosack, L. Nickel, and M. Peresano. „Prototype Open Event Reconstruction Pipeline for the Cherenkov Telescope Array“. In: *Proceedings, 37th International Cosmic Ray Conference*. Vol. 395. 744. Berlin, Germany, 2021. DOI: 10.22323/1.395.0744.

[7] K. Kosack. *cta-observatory/ctapipe: v0.19.0 – 2023-03-31*. Version v0.19.0. Mar. 2023. DOI: 10.5281/zenodo.7788918. URL: https://doi.org/10.5281/zenodo.7788918.

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[9] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. „Deep Unsupervised Learning using Nonequilibrium Thermodynamics“. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2256–2265. URL: https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[10] M. Paganini, L. de Oliveira, and B. Nachman. „CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks“. In: *Physical Review D* 97.1 (Jan. 2018). DOI: 10.1103/physrevd.97.014021. URL: https://doi.org/10.1103%2Fphysrevd.97.014021.

[11] M. Erdmann, J. Glombitza, and T. Quast. „Precise Simulation of Electromagnetic Calorimeter Showers Using a Wasserstein Generative Adversarial Network“. In: *Computing and Software for Big Science* 3.1 (Jan. 2019). DOI: 10.1007/s41781-018-0019-7. URL: https://doi.org/10.1007%2Fs41781-018-0019-7.

[12] P. Musella and F. Pandolfi. „Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks“. In: *Computing and Software for Big Science* 2.1 (Nov. 2018). DOI: 10.1007/s41781-018-0015-y. URL: https://doi.org/10.1007%2Fs41781-018-0015-y.

[13] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, et al. *CaloClouds: Fast Geometry-Independent Highly-Granular Calorimeter Simulation*. 2023. arXiv: 2305.04847 [physics.ins-det].

[14] NASA. *Gamma-ray sources in the night sky*. [Online; accessed 20-September-2023]. URL: https://fermi.gsfc.nasa.gov/ssc/.

[15] F. Aharonian, J. Buckley, T. Kifune, and G. Sinnis. „High energy astrophysics with ground-based gamma ray detectors“. In: *Reports on Progress in Physics* 71.9 (Aug. 2008), p. 096901. DOI: 10.1088/0034-4885/71/9/096901. URL: https://dx.doi.org/10.1088/0034-4885/71/9/096901.

[16] C. Li and B. Ma. „LHAASO discovery of highest-energy photons towards new physics“. In: *Science Bulletin* 66.22 (Nov. 2021), pp. 2254–2256. DOI: 10.1016/j.scib.2021.07.030. URL: https://doi.org/10.1016%2Fj.scib.2021.07.030.

[17] G. Di Sciascio. „Ground-based Gamma-Ray Astronomy: an Introduction“. In: *Journal of Physics: Conference Series* 1263.1 (June 2019), p. 012003. DOI: 10.1088/1742-6596/1263/1/012003. URL: https://dx.doi.org/10.1088/1742-6596/1263/1/012003.

[18] M. Longair. *High Energy Astrophysics - Third Edition*. Cambridge University Press, 2011. ISBN: 978-0-521-75618-1.

[19] S. Funk. „Ground- and Space-Based Gamma-Ray Astronomy“. In: *Annual Review of Nuclear and Particle Science* 65.1 (2015), pp. 245–277. DOI: 10.1146/annurev-nucl-102014-022036. eprint: https://doi.org/10.1146/annurev-nucl-102014-022036. URL: https://doi.org/10.1146/annurev-nucl-102014-022036.

[20] W. Heitler. *The quantum theory of radiation*. Courier Corporation, 1984.

[21] J. Matthews. „A Heitler model of extensive air showers“. In: *Astroparticle Physics* 22.5-6 (2005), pp. 387–397.

[22] T. Niggemann. „The silicon photomultiplier telescope FAMOUS for the detection of fluorescence light“. Veröffentlicht auf dem Publikationsserver der RWTH Aachen University; Dissertation, RWTH Aachen University, 2016. Dissertation. Aachen: RWTH Aachen University, 2016, 1 Online–Ressource (viii, 215 Seiten) : Illustrationen, Diagramme. URL: https://publications.rwth-aachen.de/record/673852.

[23] E. I. Lipatov, V. F. Tarasenko, M. V. Erofeev, V. S. Ripenko, and M. A. Shulepov. „Vavilov–Cherenkov Radiation in the Region 200–300 nm in the Earth's Atmosphere“. In: *Atmospheric and Oceanic Optics* 33 (2020), pp. 195–197.

[24] S. T. Spencer, J. J. Watson, G. Giavitto, G. Cotter, and R. White. *Advanced Analysis of Night Sky Background Light for SSTCAM*. 2023. arXiv: 2302.07611 [astro-ph.IM].

[25] A. M. Hillas. „Cerenkov light images of EAS produced by primary gamma“. In: *19th Intern. Cosmic Ray Conf-Vol. 3*. OG-9.5-3. 1985.

[26] *H.E.S.S. II Telescope Array*. This work is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported License. To view a copy of this license, visit `https://creativecommons.org/licenses/by-sa/3.0/deed.en`. [Online; accessed 15-September-2023]. 2012. URL: `https://commons.wikimedia.org/wiki/File:H.E.S.S._II_Telescope_Array.jpg`.

[27] H. Abdalla, F. Aharonian, F. A. Benkhali, E. O. Angüner, et al. „First Ground-based Measurement of Sub-20 GeV to 100 GeV γ-rays from the Vela Pulsar with HESS II“. In: *Astronomy & Astrophysics* 620 (2018), A66.

[28] P. Vincent. „HESS Phase II“. In: *29th International Cosmic Ray Conference (ICRC29), Volume 5*. Vol. 5. 2005, p. 163.

[29] G. Giavitto, S. Bonnefoy, T. Ashton, M. Backes, et al. „Performance of the upgraded H.E.S.S. cameras“. In: *Proceedings of 35th International Cosmic Ray Conference — PoS(ICRC2017)*. Sissa Medialab, Mar. 2018. DOI: `10.22323/1.301.0805`. URL: `https://doi.org/10.22323%2F1.301.0805`.

[30] MPIK Heidelberg website. *FlashCam*. [Online; accessed 15-September-2023]. 2020. URL: `https://www.mpi-hd.mpg.de/mpi/de/forschung/abteilungen-und-gruppen/nichtthermische-astrophysik/projekte/cta/flashcam`.

[31] B. Bi, M. Barcelo, C. Bauer, F. A. Benkhali, et al. „Performance of the new FlashCam-based camera in the 28m telescope of H.E.S.S.“ In: *Proceedings of 37th International Cosmic Ray Conference — PoS(ICRC2021)*. Sissa Medialab, July 2021. DOI: `10.22323/1.395.0743`. URL: `https://doi.org/10.22323%2F1.395.0743`.

[32] R. D. Parsons, M. Gajdus, and T. Murach. *HESS II Data Analysis with ImPACT*. 2015. arXiv: `1509.06322 [astro-ph.IM]`.

[33] S. Funk, G. Hermann, J. Hinton, D. Berge, et al. „The trigger system of the H.E.S.S. telescope array“. In: *Astroparticle Physics* 22.3-4 (Nov. 2004), pp. 285–296. DOI: `10.1016/j.astropartphys.2004.08.001`. URL: `https://doi.org/10.1016%2Fj.astropartphys.2004.08.001`.

[34] Cherenkov Telescope Array Observatory. *Proposed CTAO Telescopes*. This work is licensed under the Attribution-NonCommercial-NoDerivs 2.0 Generic (CC BY-NC-ND 2.0) License. To view a copy of this license, visit `https://creativecommons.org/licenses/by-nc-nd/2.0/`. [Online; accessed 15-September-2023]. 2016. URL: `https://www.flickr.com/photos/cta_observatory/30329828142/in/album-72157672713462861/`.

[35] W. Hofmann and R. Zanin. *The Cherenkov Telescope Array*. 2023. arXiv: `2305.12888 [astro-ph.IM]`.

[36] D. Malyshev and L. Mohrmann. *Analysis Methods for Gamma-ray Astronomy*. 2023. arXiv: `2309.02966 [astro-ph.IM]`.

[37] H.E.S.S. confluence website. Internal setting.

[38] M. Erdmann, J. Glombitza, G. Kasieczka, and U. Klemradt. *Deep Learning For Physics Research*. World Scientific, 2021.
[39] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.
[40] F. Chollet et al. *Keras*. https://keras.io. 2015.
[41] B. Xu, N. Wang, T. Chen, and M. Li. *Empirical Evaluation of Rectified Activations in Convolutional Network*. 2015. arXiv: 1505.00853 [cs.LG].
[42] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].
[43] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].
[44] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. arXiv: 1207.0580 [cs.NE].
[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. „Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
[46] H. Gholamalinezhad and H. Khosravi. *Pooling Methods in Deep Neural Networks, a Review*. 2020. arXiv: 2009.07485 [cs.CV].
[47] L. Ruthotto and E. Haber. *An Introduction to Deep Generative Modeling*. 2021. arXiv: 2103.05180 [cs.LG].
[48] D. Saxena and J. Cao. *Generative Adversarial Networks (GANs Survey): Challenges, Solutions, and Future Directions*. 2023. arXiv: 2005.00065 [cs.LG].
[49] M. Arjovsky, S. Chintala, and L. Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].
[50] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028 [cs.LG].
[51] M. Arjovsky and L. Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. arXiv: 1701.04862 [stat.ML].
[52] A. Odena, C. Olah, and J. Shlens. *Conditional Image Synthesis With Auxiliary Classifier GANs*. 2017. arXiv: 1610.09585 [stat.ML].
[53] K. Bernlöhr. „Simulation of imaging atmospheric Cherenkov telescopes with CORSIKA and sim_telarray". In: *Astroparticle Physics* 30.3 (Oct. 2008), pp. 149–158. DOI: 10.1016/j.astropartphys.2008.07.009. URL: https://doi.org/10.1016%2Fj.astropartphys.2008.07.009.
[54] Kalsruhe Institute of Technology. *KIT website - CORSIKA*. [Online; accessed 15-October-2023]. URL: https://www.iap.kit.edu/corsika/.
[55] J. Schaefer. Personal correspondence. 2023.
[56] D. Heck and T. Pierog. „Extensive Air Shower Simulation with CORSIKA: A User's Guide (Version 7.7500 from April 14, 2023)". In: ().
[57] W. R. Nelson and Y. Namito. „The EGS4 Code System: Solution of gamma-ray and electron transport problems". In: (Feb. 1990). URL: https://www.osti.gov/biblio/7162777.

[58] S. Ostapchenko. *LHC data on inelastic diffraction and uncertainties in the predictions for longitudinal extensive air shower development*. Apr. 2014. URL: https://journals.aps.org/prd/abstract/10.1103/PhysRevD.89.074009.

[59] E. Hoogeboom, J. W. T. Peters, T. S. Cohen, and M. Welling. *HexaConv*. 2018. arXiv: 1803.02108 [cs.LG].

[60] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[61] R. Chalme-Calvet, M. de Naurois, and J. -P. Tavernet. *Muon efficiency of the H.E.S.S. telescope*. 2014. arXiv: 1403.4550 [astro-ph.IM].

[62] J. Dubenskaya and A. Kryukov and A. Demichev. „Fast simulation of gamma/proton event images for the TAIGA-IACT experiment using generative adversarial networks“. In: *37th International Cosmic Ray Conference*. Vol. 395. SISSA Medialab, 2021, p. 874. DOI: 10.22323/1.395.0874.

[63] J. Dubenskaya et al. „Using a Conditional Generative Adversarial Network to Control the Statistical Characteristics of Generated Images for IACT Data Analysis“. In: *Proceedings of The 6th International Workshop on Deep Learning in Computational Physics —PoS(DLCP2022)*. Sissa Medialab, Nov. 2022. DOI: 10.22323/1.429.0004. URL: https://doi.org/10.22323%2F1.429.0004.

[64] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. *Spectral Normalization for Generative Adversarial Networks*. 2018. arXiv: 1802.05957 [cs.LG].

Acknowledgements

I want to take this opportunity to thank everyone involved in my master's thesis in one way or another. I want to thank my family for making all of this possible in the first place. I could have never come this far without their constant support through all these years. Furthermore, I want to thank the following people:

- Prof. Dr. Stefan Funk for making it possible for me to work on such an interesting topic and be a part of the gamma group.
- Dr. Jonas Glombitza for literally everything. There are so many things I have to thank him for, but I would probably forget to mention half of it anyway. Anyway, thanks for helping me get into machine learning, for helping me produce better code, presentations and texts, for giving motivational speeches, for writing a paper with me about my work, ...
- Dr. Johannes Schäfer for always being ready to talk about anything, work-related or not, for reviewing the gamma chapter of my thesis, for all the wisdom he has shared and for basically being a life advisor throughout all these last years.
- Dr. Samuel Spencer for answering many work-related questions, for reviewing the summary chapter of my work, for the interesting talks on the bus in the mornings and afternoons, and for his constant positivity throughout the last year.
- Benedetta Bruno for processing the simulated data so that I could use it in my work, for helping me with work-related stuff, and for reviewing my data chapter.
- Mario Engelmann, for basically being my office partner from another office and always being ready to procrastinate with me.
- Simon Steinmassl for simulating, calibrating and cleaning the IACT images I used in my work and helping me understand why the mount pixels have signals.
- Franzi, Martin, 2×Andi, Katrin, Jelena, Gabi, Matheus, Rodrigo for making this last year a better experience.
- Every other person from ECAP who was somehow involved in my master's thesis and not mentioned above.

Statutory declaration

I hereby declare that I have written this Master's thesis independently and have not used any sources other than those indicated. All text passages taken verbatim and in spirit from external sources are marked. The thesis has not been submitted to any other examination authority in the same or similar form and has not been recognized as an examination performance by any other examination authority.

Erlangen, 16 October, 2023

Eigenständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle wörtlich und sinngemäß aus fremden Quellen übernommenen Textpassagen sind gekennzeichnet. Die Arbeit ist in gleicher oder ähnlicher Form bei keiner anderen Prüfungsbehörde eingereicht worden und sie wurde nicht von einer anderen Prüfungsbehörde als Prüfungsleistung anerkannt.

Erlangen, 16. Oktober, 2023

Christian Elflein