Modern Deep Learning techniques for fitting multi-dimensional data distributions in Particle Physics applications

von

David Josef Schmidt

Masterarbeit in Physik

vorgelegt der
Fakultät für Mathematik, Informatik und Naturwissenschaften
der
Rheinisch-Westfälischen Technischen Hochschule Aachen

im Oktober 2018

angefertigt am

III. Physikalischen Institut A

Professor Dr. Martin Erdmann

Erstgutachter
Prof. Dr. Martin Erdmann
III. Physikalisches Institut A
RWTH Aachen

Zweitgutachter
Prof. Dr. Alexander Schmidt
III. Physikalisches Institut A
RWTH Aachen

David Josef Schmidt
Modern Deep Learning techniques for fitting multi-dimensional data distributions in Particle Physics applications

Masterarbeit in Physik
Rheinisch-Westfälische Technische Hochschule Aachen
III. Physikalisches Institut A

Bearbeitungszeitraum: 16.04.2018 bis 15.10.2018

Contents

Page

1 Introduction **1**

2 Experiments and Analyses **3**
2.1 High Energy Physics . . . 3
2.1.1 The Standard Model of Particle Physics . . . 3
2.1.2 The Compact Muon Solenoid . . . 7
2.1.3 Particle Reconstruction . . . 11
2.1.4 Monte Carlo Data Generation . . . 13
2.2 Astroparticle Physics . . . 13
2.2.1 Cosmic Particle Accelerators . . . 13
2.2.2 Propagation of Cosmic Rays through the Universe . . . 15
2.2.3 Galactic Magnetic Field . . . 16
2.2.4 Extensive Air Showers . . . 17
2.2.5 The Pierre Auger Observatory . . . 19

3 Deep Learning **23**
3.1 Basics . . . 23
3.1.1 Networks . . . 23
3.1.2 Loss Functions . . . 26
3.1.3 Backpropagation . . . 27
3.1.4 Gradient Descent . . . 28
3.2 Advanced Concepts . . . 29
3.2.1 Normalization and Initialization . . . 29
3.2.2 Optimizer . . . 30
3.2.3 Training and Validation . . . 30
3.2.4 Regularization . . . 31
3.2.5 Early Stopping . . . 32
3.2.6 Hyperparameter Optimization . . . 32
3.2.7 Adversarial Networks . . . 33
3.3 Software . . . 35
3.3.1 TensorFlow . . . 35
3.3.2 Luigi and Law . . . 36

4 Event Simulation with Generative Adversarial Networks **37**
4.1 Generalization . . . 37
4.2 Improved Wasserstein-GAN . . . 38
4.2.1 Wasserstein Distance . . . 38
4.2.2 Critic replaces Discriminator . . . 40
4.3 Training and Data Setup . . . 43
4.3.1 Wasserstein-GAN Architecture and Training . . . 43
4.3.2 Preprocessing and Shape Normalization . . . 44
4.3.3 Datasets . . . 45

4.4 Metrics and Results . . . 46
4.4.1 Wasserstein Distance as Metric . . . 46
4.4.2 Fisher Transformation . . . 48
4.4.3 Classification Benchmark . . . 50

5 Deep Scale Factors **53**
5.1 Generalization . . . 53
5.2 DeepSF Architecture . . . 54
5.3 Traditional Scale Factors and Data Setup . . . 56
5.4 Training . . . 57
5.5 Results . . . 58
5.5.1 Comparison to Traditional Scale Factors . . . 59
5.5.2 Extension to more Input Variables . . . 62
5.5.3 Summary and Outlook . . . 64

6 Fit of the observed Cosmic Ray Skymap **67**
6.1 Generalization . . . 68
6.2 Fit Architecture . . . 68
6.3 Fit Procedure . . . 71
6.4 Results . . . 72
6.4.1 1-dimensional Scenario . . . 72
6.4.2 Scenario of Spherical Rotation . . . 75
6.4.3 Full Galactic Magnetic Field Scenario . . . 78
6.4.4 Summary . . . 81

7 Conclusion **83**

A. High-level Variables **A**

B. Bins and Selection Cuts for b-tagging SFs **A**

References **C**

Acronyms **H**

1 Introduction

Modern particle physics analyses make extensive use of computing and data storing resources. Larger amounts of data reduce statistical fluctuations and make the discovery of rare physics possible. Large analysis workflows and detailed physics simulations improve the search for new physics, the precision-measurement of physical constants or simply enable physicists to evaluate datasets that have only scientific value after aggregation and reconstruction.

A more recent development is the application of Deep Learning methods within such analyses. In particular, they had a big success in replacing classical discrimination tasks based on a large number of input variables, as the development within a relatively short amount of time is able to outperform traditional algorithms. The main reason for their advance is the fact that they can exploit local correlations in high-dimensional data much faster and more accurate than humans do through hand-crafted algorithms [1, 2].

Analysis workflows do not only consist of discrimination tasks but many others that operate on high-dimensional data as well. A promising subfield of such tasks seems to be the fitting and comparing of large datasets for different purposes.

In this thesis, three different tasks concerning the fitting of high-dimensional data in the field of particle physics are presented. The first two are concerning the production of simulated data in the setting of a High Energy Physics experiment, the CMS detector [3]. The first method will investigate the possibility of fitting an output distribution of a neural network to a given data distribution of simulated events, resulting in a possible increased production efficiency of simulations while retaining the necessary structure of the data. The second study focuses on a post-processing step of simulated events, called reweighting. An alternative approach using deep neural networks is proposed, to create event weights that better fit the simulated data distribution to the measured data of the experiment. The third method is in the context of Astroparticle Physics and the Pierre Auger Observatory [4] and aims at reproducing the measured data of thousands of cosmic rays through a simulated propagation of parameterized origins and charges, that are fitted in a novel way using Deep Learning techniques.

While all three tasks are situated around the general task of fitting high-dimensional data, their goals in terms of physical results are inherently different. The first task deals with a computational efficient simulation of data. The second task takes an already simulated set and tries to refine it to measured data to avoid transfer learning problems between data and simulation. The third task is about inverting a functional dependence of the measured data to retrieve the hidden physics information by performing a high-dimensional fit.

Before going into the details of the different methods the thesis will start with the presentation of the two particle physics experiments that provide the physical background. Afterward, a section on Deep Learning introduces the general theory and techniques that are used throughout this thesis. Then each method will be demonstrated while pointing out the differences and similarities to the other methods

in the more general context of data analyses. We will understand what aspects of the methods work well and why some results do not live up to their expectations. The conclusion will point out the most promising methods for future analyses based on the results of this thesis.

2 Experiments and Analyses

This first section will give an introduction to the particle physics experiments which feature some multi-dimensional fitting tasks and are later on approached with the Deep Learning (DL) techniques. The first experiment (CMS) will be presented in a bit more detail as two of the three multi-dimensional fits are based on it. Afterward, a short overview of the Pierre Auger Observatory will be given.

2.1 High Energy Physics

High Energy Physics focuses on the investigation of the Standard Model of particle physics (SM) and its possible extensions. Therefore, particle colliders are built to probe particle interactions and decays by detecting the corresponding end products and reconstructing the particle interactions based on this data. The Compact Muon Solenoid (CMS) detector is one such detector and part of the Large Hadron Collider (LHC). Scientists from roughly 200 institutes from around the world are participating in developing the detector structure, the data analyses, and the corresponding theoretical physics models.

This chapter is starting with a short overview of the theoretical background of elementary particle physics followed by a presentation of the CMS detector. It will be concluded with a simplified outline of a typical analysis pointing out the parts where the work of this thesis hooks in.

2.1.1 The Standard Model of Particle Physics

Particle physics is the field of research that tries to find and explain the behavior of the most fundamental building blocks of our universe. The currently best-describing theoretical model of the observed phenomena is the SM. The SM is a quantum field theory, meaning that it describes matter and forces as excitations of fields which behave according to quantum mechanics. This model also includes special relativity as it is completely Lorentz-invariant, however, fails to include the gravitational force and thereby only covering the strong and the unified electroweak force.

Matter and Forces Today there are a relatively small number of truly elementary particles incorporated in the SM that add up to the matter and the propagated forces in our universe. Figure 2.1 gives a good overview of all particles.

The matter as we know it is build up of quarks organized in mesons (2 quarks) and baryons (3 quarks). Because of the strong force quarks do not exist as free particles they are only found in bound states. Additionally, charged leptons build up the outer layers of atoms and hold them together through the electromagnetic force while also being able to interact with the neutral leptons, called neutrinos, through the weak force. Each of these particles has a corresponding anti-particle which all together can be organized in three generations consisting each of a negatively charged lepton and the corresponding neutrino as well as an up-type quark with charge $\frac{2}{3}$ and a

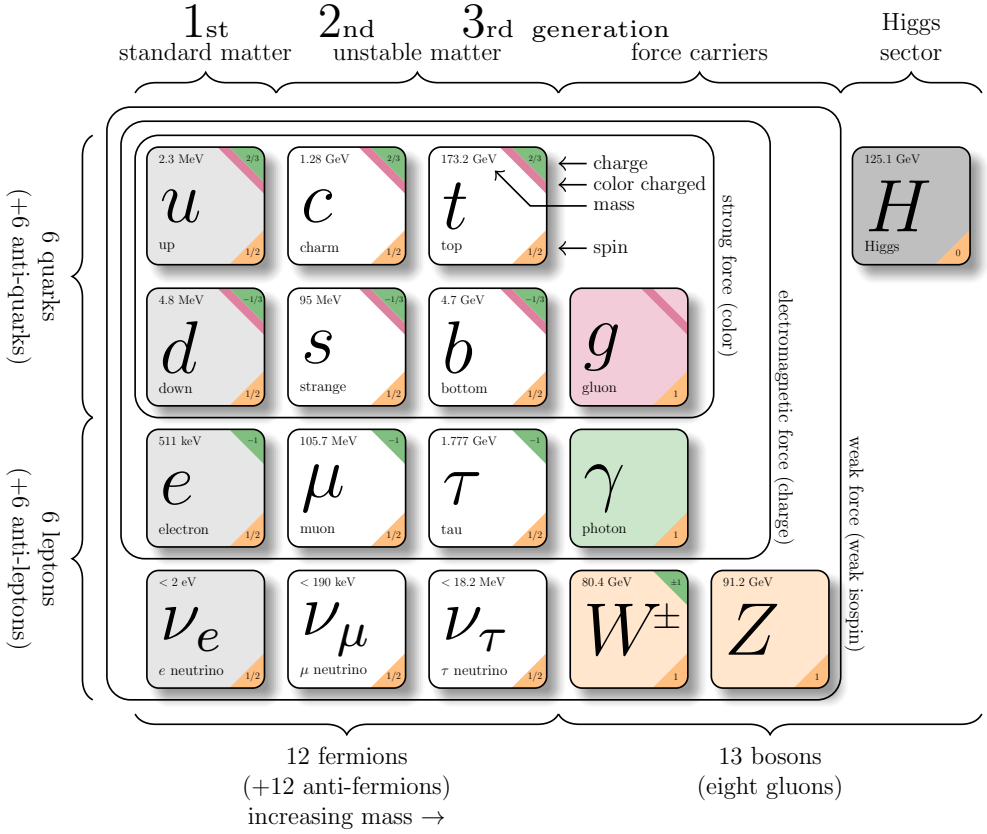

Figure 2.1: Particles and forces of the SM. [5]

down-type quark with charge $-\frac{1}{3}$. The model describes the neutrinos as massless particles while recent experiments have shown that they, in fact, are not [6] resulting in one of the model's shortcomings.

The forces are introduced through bosons, each one of them being responsible for one of the fundamental forces. The weak force is propagated through the $W^{\pm}$ and the Z^0 bosons and interacts with all left-handed (negative chirality) fermions of the SM. It is the only force that provides a link between the different fermion generations. The Higgs boson being the particle that breaks the symmetry in the electroweak force also interacts with all fermions. This spontaneous symmetry breaking delivers also an explanation for the masses of the $W^{\pm}$ and the Z^0 bosons while the photon (γ) stays massless. The photon is the mediator of the electromagnetic force and, therefore, only interacts with charged fermions. The quarks, being the only fermions with color charge, are the only particles that interact with the 8 gluons that mediate the strong force and the color charges [7].

Mathematical Theory As it is a field theory, it is mathematically best described by a Lagrangian density $\mathcal{L}(x)$ as a function of space-time coordinates x. When expanding the Lagrangian density (often abbreviated as simply the Lagrangian)

into a power series[1] one can derive an upper bound on the order of the power that can appear in this series expansion. This relation depends on the exact structure of the field and other involving fields but needs to hold to get finite answers for well-defined physical observables. This significantly reduces the possible Lagrangians for the SM.

Another important aspect is symmetries. There are observed symmetries in our universe that should hold in any theory, e.g. spacial translation and rotation as well as the CPT symmetry. Symmetries greatly reduce the space of possible models and are simple and elementary. Following the guideline of constructing the simplest theory that can describe all physical phenomena it is natural to build up a model by postulating a few symmetries that lead to a single possible Lagrangian that correctly describes our world.

The SM is build up from the symmetry group

$$\underbrace{SU(3)}_{\text{QCD}} \otimes \underbrace{SU(2) \otimes U(1)}_{\text{Electroweak \& Higgs}} \tag{2.1}$$

which establishes Quantum Chromodynamics (QCD) with the gluons as force mediators between color-charged particles and features the unified theory of Electromagnetism, the weak nuclear force and the Higgs mechanism [8].

Quantum Chromodynamics The $SU(3)$ symmetry group is 8-dimensional and therefore has 8 linear independent generators $T_c = \frac{\lambda_c}{2}$ for $c = 1 \dots 8$ of the corresponding Lie algebra defining the basis for the infinitesimal transformations under which gauge invariance is required. Here the Gell-Mann matrices can be used to describe the generators in a systematic way. Therefore, the theory of the so-called quark field $\psi(x)$ as the fundamental representation of the $SU(3)$ gauge group has to be symmetric under

$$\psi(x) \to [1 + \mathrm{i}\alpha_a(x)T_a]\,\psi(x) \tag{2.2}$$
$$\partial_\mu\psi \to [1 + \mathrm{i}\alpha_a(x)T_a]\,\partial_\mu\psi + \mathrm{i}T_a\psi\partial_\mu\alpha_a(x). \tag{2.3}$$

Starting from the kinematic term $\bar{\psi}(\mathrm{i}\gamma^\mu\partial_\mu - m_\psi)\psi$ one has to introduce 8 gauge fields $\mathcal{A}^c_\mu$ and the corresponding gauge covariant derivative $D_\mu = \partial_\mu + \mathrm{i}g_sT_a\mathcal{A}^a_\mu$ to complete the kinetic QCD Lagrangian to be gauge invariant.

$$\mathcal{L}_{QCD} = \bar{\psi}(\mathrm{i}\gamma^\mu D_\mu - m_\psi)\psi - \frac{1}{4}G^a_{\mu\nu}G^{\mu\nu}_a \tag{2.4}$$
$$= \underbrace{\bar{\psi}(\mathrm{i}\gamma^\mu\partial_\mu - m_\psi)\psi}_{\text{quark kinematics}} - \underbrace{g_s(\bar{\psi}\gamma^\mu T_a\psi)\mathcal{A}^a_\mu}_{\text{interaction } \psi\psi\mathcal{A}} - \underbrace{\frac{1}{4}G^a_{\mu\nu}G^{\mu\nu}_a}_{\text{gluon kinematics}} \tag{2.5}$$

[1] Computing observables from Lagrangians is done in perturbation theory and extremely tedious. A lot of mathematical tools were developed to ease this step and better understand the theory, e.g. Feynman diagrams and corresponding rules, renormalization schemes etc.

One can specifically check the gauge invariance by using the transformation law $\mathcal{A}^a_\mu \to \mathcal{A}^a_\mu - \frac{1}{g_s}\partial_\mu \alpha_a - f_{abc}\alpha_b \mathcal{A}^c_\mu$ and the definition of the gluon field strength tensor $G^a_{\mu\nu} = \partial_\mu \mathcal{A}^a_\nu - \partial_\nu \mathcal{A}^a_\mu + g_s f_{abc}\mathcal{A}^b_\mu \mathcal{A}^c_\nu$. Here f_{abc} are the structure constants of the algebra and g_s is the quark-gluon coupling strength.

Electroweak Unification and Spontaneous Symmetry Breaking The $SU(2)$ symmetry group is 3-dimensional and $U(1)$ is 1-dimensional, on has therefore 4 generators them being $T_i = I_i = \frac{\sigma_i}{2}$ for $i = 1 \dots 3$ and $T_4 = Y$ with I and Y being the operators for the weak isospin and the hypercharge, respectively. Corresponding to physical observations there exist fermions with positive chirality (right-handed) having always an isospin of 0 and not coupling to the weak force as well as fermions with negative chirality (left-handed) always having isospin $\frac{1}{2}$ and coupling to the weak force. According to these observations (and the non-existence of right-handed neutrinos)[2] one classifies the fermions into the corresponding doublets for $I = \frac{1}{2}$ and singlets for $I = 0$:

$$\begin{pmatrix} u \\ d \end{pmatrix}_L, u_R, d_R \qquad \text{and analogously for } c, s, t, b \tag{2.6}$$

$$\begin{pmatrix} \nu_e \\ e \end{pmatrix}_L, e_R \qquad \text{and analogously for } \nu_\mu, \mu, \nu_\tau, \tau \tag{2.7}$$

Similarly as with the QCD Lagrangian one has to introduce gauge bosons to fix the required local symmetry transformations

$$\psi(x) \to [1 + \mathrm{i}\beta(x)Y + \mathrm{i}\alpha_i(x)I_i]\psi(x) \quad \text{with } I_i\psi_R = 0. \tag{2.8}$$

With the covariant derivative $D_\mu = \partial_\mu + \mathrm{i}gI_jW^j_\mu + \mathrm{i}g'YB_\mu$ we can complement the kinetic electroweak Lagrangian to be gauge invariant.

$$\mathcal{L}_{EW,kin} = \sum_f \bar{\psi}_f \mathrm{i}\gamma^\mu D_\mu \psi_f - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}W^i_{\mu\nu}W^{\mu\nu}_i \tag{2.9}$$

Once again one has the corresponding field strengths $B_{\mu\nu} = \partial_\mu B_\nu - \partial_\nu B_\mu$ and $W^i_{\mu\nu}$ analogous to $G^a_{\mu\nu}$ using $f_{abc} = \epsilon_{abc}$ as the structure constants for $SU(2)$ and g instead of g_s as coupling.

This Lagrangian does not feature a mass term for the fermions nor a mass term for the gauge bosons as both would break the local gauge symmetry. The Higgs mechanism introduces a spontaneous symmetry breaking that makes it possible to write down a gauge invariant Lagrangian while also heaving mass terms for the fermions and bosons.

Introducing a complex scalar field Φ with a potential

$$V(\Phi) = -\mu^2\Phi^\dagger\Phi + \frac{\lambda}{4}(\Phi^\dagger\Phi)^2 \tag{2.10}$$

[2] All the statements are of course regarding matter, for antimatter it is the opposite.

which has its global minima at $\Phi = \mu\sqrt{\frac{2}{\lambda}}e^{\mathrm{i}\theta}$ instead of 0 causes the ground state not to be symmetric under the $U(1)$-symmetry of a scalar field. This leads to the Higgs Lagrangian

$$\mathcal{L}_H = (D_\mu \Phi)^\dagger (D^\mu \Phi) - V(\Phi) \tag{2.11}$$

Parametrizing the field Φ with respect to the ground state and removing a degree of freedom through gauge fixing of the parameter θ of the ground state one can write

$$\Phi(x) = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v + H(x) \end{pmatrix} \tag{2.12}$$

with H being the Higgs field and $v = \frac{\mu}{\sqrt{\lambda}}$ being the expectation value of the ground state. Through the constant offset v and the gauge fields in the D_μ, this Lagrangian actually creates mass terms for the bosons. Additionally, introducing the electroweak mixing angle $\theta_W = \arctan(g'/g)$ between the two neutral boson fields W^3 and B one can create the two physical bosons

$$\begin{pmatrix} \gamma \\ Z^0 \end{pmatrix} = \begin{pmatrix} \cos\theta_W & \sin\theta_W \\ -\sin\theta_W & \cos\theta_W \end{pmatrix} \cdot \begin{pmatrix} B \\ W^3 \end{pmatrix} \tag{2.13}$$

while W^1, W^2 are mixed in such a way to create the two bosons with exact charge

$$W^\pm = \frac{1}{\sqrt{2}}(W^1 \pm W^2). \tag{2.14}$$

This mechanism creates all the boson masses and couplings with just the free model parameters of θ_W, g, g', μ and λ which fit extremely well with measured data [9].

With the Higgs field, one can now also introduce mass terms for the fermions leading to the Yukawa Lagrangian:

$$\mathcal{L}_Y = -\sum_f m_f \bar{\psi}_f \psi_f - \sum_f \frac{m_f}{v} \bar{\psi}_f \psi_f H \tag{2.15}$$

The full electroweak Lagrangian reads then

$$\mathcal{L}_{EW} = \mathcal{L}_{EW,kin} + \mathcal{L}_H + \mathcal{L}_Y. \tag{2.16}$$

2.1.2 The Compact Muon Solenoid

To explore the subatomic world of particle physics physicists have built larger and larger particle accelerators and detectors. These setups are usually collision experiments creating head-on collisions of particles with high energies which in turn create particles of potentially high masses. These particles decay again into (semi-stable) particles which can be (in-)directly measured. The amount, the energy and their respective direction can then be used in physics analyses to create a deeper understanding.

The CMS experiment is a particular particle detector located at European Organization for Nuclear Research (CERN) measuring the particle collisions of the LHC accelerator, which will be briefly introduced before going into more detail of the CMS experiment itself.

Large Hadron Collider The LHC is the world's largest physics apparatus mainly consisting of a 27 km long beam pipe inside a circular tunnel around 100m deep under the ground of the French-Swiss border (see fig. 2.2). The start of construction was in 1998 using the tunnel of the Large Electron-Positron Collider (LEP) by replacing it completely. It took 10 years to built and started taking data end of 2009.

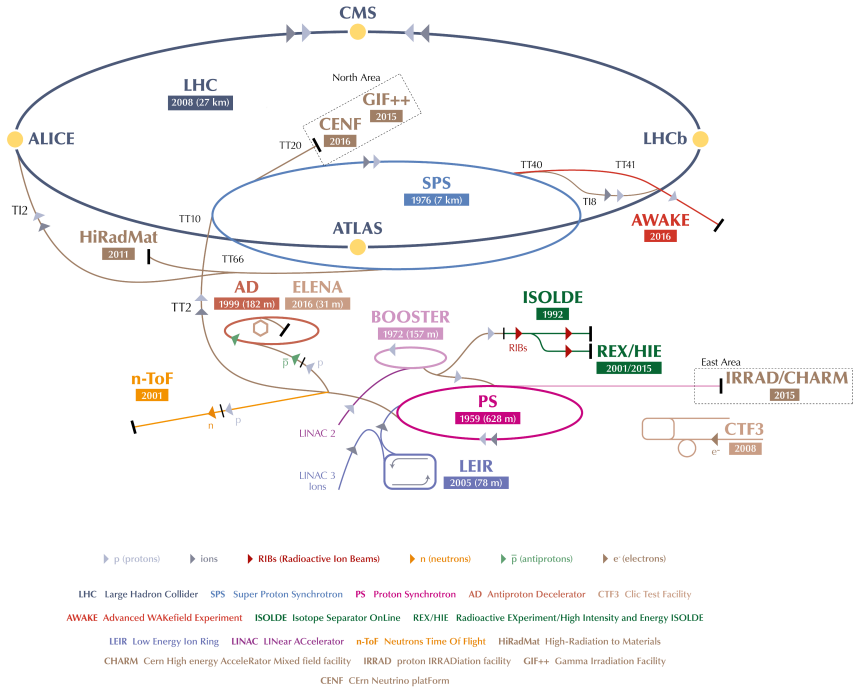

Figure 2.2: Overview of the beam pipe and accelerator infrastructure at the CERN. [10]

Most of the year the LHC is running proton-proton collision experiments with a center-of-mass energy of around 13 TeV and an instantaneous luminosity of the order of $10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$.Additionally, one of the last months every year the LHC is running heavy ion experiments by colliding lead nuclei with protons at a center-of-mass energy of around 5 TeV and with an instantaneous luminosity of $10^{27}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$.

To actually fill the LHC with highly energetic protons it is necessary to pre-accelerate them as the large ring makes it extremely inefficient to accelerate protons starting from zero. Here come the smaller accelerators at the CERN into play. The protons

start off at a linear accelerator, the Linear Accelerator 2 (LINAC2), bringing the protons up to 50 MeV. The Proton Synchrotron Booster (PSB), Proton Synchrotron (PS) and Super Proton Synchrotron (SPS) bring them up to 1.4 GeV, 25 GeV, and 450 GeV respectively, until they enter the LHC and are there brought up to the final energy of 6.5 TeV per proton.

The acceleration is done via 8 superconducting radiofrequency cavities (400 MHz) providing 485 keV per turn to each proton. They also synchronize the protons organizing them in smaller bunches to get a higher instantaneous luminosity per bunch crossing.

To keep the protons on their path in the pipe and control the points of beam crossing strong magnets are needed all around the LHC. Keeping the beam small and narrow requires focusing it from time to time. This is done by 392 quadrupoles which provide a magnetic field that acts as a lens on the protons. 1232 dipole magnets with a strength of up to 8.3 T are used to bend the beam keeping it on the circular route through the pipe. To reach these magnetic fields efficiently with electromagnets one exploits the features of the superconductive material NbTi at a temperature of 1.9 K.

The CMS is one of the two general purpose detectors with the other being A Toroidal LHC ApparatuS (ATLAS) sitting at the opposite side of the LHC. Both were built to discover and measure the Higgs boson while also investigating physics at high luminosities. The two other large detectors located next to ATLAS are the Large Hadron Collider beauty (LHCb) and A Large Ion Collider Experiment (ALICE) with the former being specialized on measurements around the bottom quark and the latter being optimized to observe the heavy-ion collisions in great detail. [11]

The CMS Detector The CMS detector [3] itself consists of multiple layers each one of them fulfilling a different purpose. They are shown in fig. 2.3 and will be discussed in the same order as the particles would pass them, from the inside out.

Tracker The tracker is the inner-most layer of CMS and consists of a large number of silicon detectors organized in pixels and readout strips. It is their responsibility to measure the exact particle track without having a big impact on its flight path. A particle passing a few of the silicon chips will create a measurable signal in each of the passed pixels which leads to an identifiable path. Measuring and discriminating the track from others with a high resolution but only a few interaction points (in order to keep the particle unaffected) requires a high granularity of the pixels (10 µm).

The particle tracks are important to reconstruct the initial particle momentum. Together with a precise energy measurement from the calorimeters, one can reconstruct the entire four-momentum. [14]

Electromagnetic Calorimeter The electromagnetic calorimeter (ECAL) is optimized to measure the energy of photons and electrons by using the dense lead

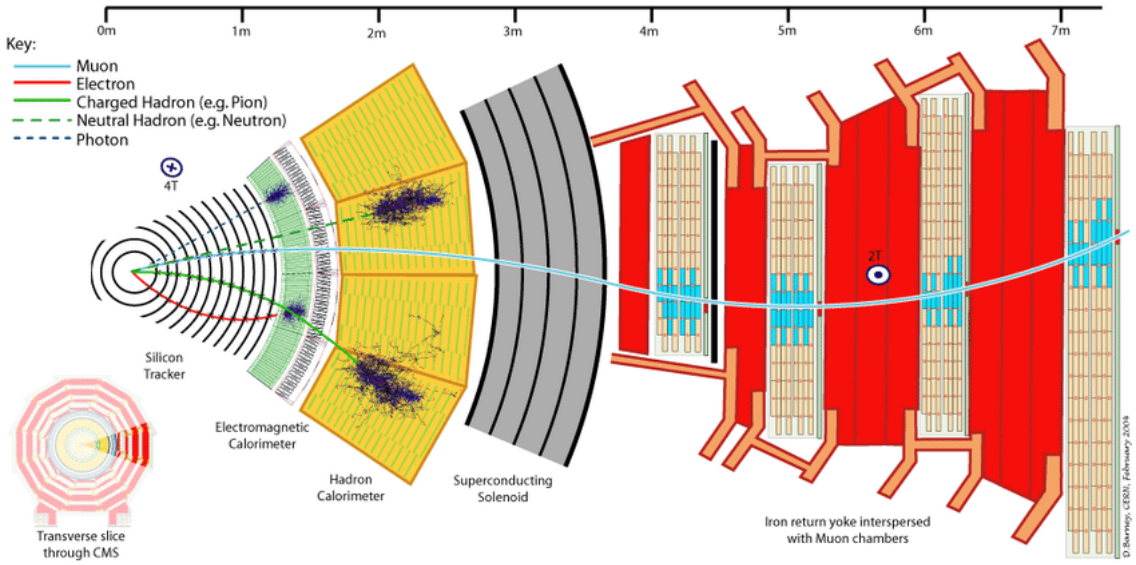

Figure 2.3: Schematic diagram of the different detector layers in CMS. [12, 13]

tungstate ($PbWO_4$) to create electromagnetic showers in this crystal scintillator material. These showers produce photons that either produce secondary showers or are in the regime of visible light to which $PbWO_4$ is transparent therefore allowing it to travel to the photomultiplier tubes (PMTs) where the energy amount of these secondary photons are measured to give an estimate on the particle energy. The energy resolution, therefore, depends on the shower structure and size resulting in an overall energy dependence [15]:

$$\frac{\sigma_E}{E} = \sqrt{\left(\frac{2.8\%}{\sqrt{E[GeV]}}\right)^2 + \left(\frac{12\%}{E[GeV]}\right)^2 + (0.3\%)^2} \tag{2.17}$$

Hadronic Calorimeter The hadronic calorimeter (HCAL) is measuring the energy of particles interacting via the strong nuclear force, e.g. the created hadrons. Similar to the ECAL it uses fluorescent scintillator material and photodetectors to measure the energy deposit via secondary photons. However, actually inducing the showers requires a different material. Layers of brass and steel have proven to work best for the CMS experiment in order to keep the uncertainty on the energy as small as possible [16]:

$$\frac{\sigma_E}{E} = \sqrt{\left(\frac{100\%}{\sqrt{E[GeV]}}\right)^2 + (0.3\%)^2} \tag{2.18}$$

Superconducting Solenoid The solenoid is a strong superconducting Niobium Titanium (NbTi) magnet that creates a magnetic field of close to 4 T inside its barrel. This strong magnetic field is needed to ensure that also highly energetic

charged particles follow a path with a measurable curvature inside the tracker as this enables a precise momentum reconstruction. The existence and direction of the curvature also identify the charge of the particle. The steel return yoke outside of the solenoid shapes the magnetic to be homogenous not only inside but also outside of the solenoid where the muon chambers are still measuring the muon flux. [17]

Muon Chambers Muons are the only particles passing through the previous layers and also the return yoke, therefore the detectors just need to be optimized for their detection. There are three types of muon chambers installed in the outer region of the CMS detector, these being drift tubes, cathode strip chambers, and resistive plate chambers. They optimize different aspects of detection like trigger response times or precise position measurements. However, all are based on muons flying through some gas and ionizing it. An applied voltage results in the ionized particles to be collected by the cathode and anode resulting in an electrical signal. The timing of the signals can be used to deduce the position more accurate. [3]

Triggers The detector produces a large amount of data. With a collision rate of 40 MHz and a storage volume of approximately 1 MB per event, this would generate an unmanageable information flow. There are two filter steps that reduce the event rate sequentially. The first trigger is implemented directly in hardware as it has to be extremely fast. It reduces the number of selected events by checking simple threshold cuts and some basic aggregation of signals. After the reduction of the event rate to about 100 kHz, it is feasible to run some more involved algorithms combining and reconstructing more accurately the events. This is done on a software level with the second trigger reducing the event rate to an order of 100 to 200 events per second. The CMS experiment, therefore, produces around 200 MB per second. [18]

2.1.3 Particle Reconstruction

The raw data of the CMS experiment basically consists of position data and in addition the corresponding energy deposition in the case of the calorimeters. These are in this raw state very hard to interpret and consequently impractical for physics analyses. Thus, a couple of algorithms for reconstructing certain aspects of physical objects exist and are briefly presented.

Particle flow algorithm The goal of this algorithm is to most of the hits in the detector with particle flight paths. This is done by using the tracker and some combinatorics to find good tracking seeds (start points for a track reconstruction) and are then extrapolated to the calorimeters where corresponding clusters of energy deposits are detected. This reconstruction is done iteratively with loosening constraints on the fits with each iteration.

Muons are reconstructed using the muon chamber and finding corresponding tracker hits on a best-fit basis, as the muon tracks are actually cleaner and easier to reconstruct than the tracker hits. [19]

Vertex reconstruction With the particle tracks identified and the knowledge of the four-momentum, one already has a good description of the type of particle and its origin. A vertex is a position in space where two or more particles originated from as a result of a decay or interaction. The reconstruction of these vertices by basically clustering intersections of reconstructed particle paths is very important for determining which particles stem from a single interaction point and are therefore related and which were produced by a different proton-proton collision in the same bunch. Additionally, secondary vertices can be reconstructed, which are produced by particles with a half-life that allows them to fly a measurable distance away from the primary vertex but decaying before passing through the whole detector. There are just a few particles (B mesons) that have this property leading to a good observable for particle identification. [19]

Muon reconstruction As the muons leave a trace in the tracker as well as the muon chambers they can be reconstructed using both information. They are divided into four categories:

Global muons leave a complete signature in the tracker and the chambers, which can decrease the uncertainty on momentum measurements.

Tracker muons are located by a full trace in the tracker and at least one hit from one of the inner muon chambers. These muons often have a low momentum and, therefore, not being able to reach the outer chambers.

Standalone and cosmic muons only hit the muon chambers and are most of the times of cosmic origin. These muons are usually easy to single out as their path does not coincide with the beam. [20, 21]

Jet clustering Quarks generally decay quite fast in a process called hadronization and fragmentation and produce many stable secondary particles. As all of this particles are boosted in the general direction of the original quark they leave a detector response of multiple particles directed in the same direction. This is called a jet and there are different algorithms that try to cluster such particles into meaningful groups. A good jet cluster algorithm is the Anti-K_t algorithm, as it is in particular infrared safe (e.g. particles with low momentum do not influence the overall jet shape) and collinear safe (e.g. a particle with similar momentum and origin as a jet should be included). [22]

B-tagging As pointed out at the vertex reconstruction, the B mesons originating from hadronized b-quarks have a half-life that is just of the size to produce a secondary vertex with a measurable distance to the primary vertex. This distance allows for a successful B-tagging algorithm, called the combined secondary vertex

(CSV) algorithm. This and other algorithms produce for most jets a number between 0 and 1 indicating the likelihood of that jet being a b jet. A more involved b tagger with a higher performance is called the DeepCSV algorithm, as it combines the best aspects of Deep Neural Networks and the CSV. As most modern studies and analyses focus around heavy particles decaying into b-quarks, B-tagging is a very important aspect in the reconstruction of data. [23]

2.1.4 Monte Carlo Data Generation

Monte Carlo (MC) events are simulated events based on a certain physics model (usually the SM) that give physicists an indication of how detector responses are supposed to look like. It is important to have an accurate description of the predictions by the model as it enables us to compare it to the taken measurements and validate our models or find new physics.

As it is impossible to write algebraic expressions for the whole detector response, a probabilistic (MC) simulation method is chosen. According to their theoretical cross sections, different hard scatterings are sampled. Additional interactions and decays are stochastically performed to generate an event with resonance decays, initial and final state radiation as well as hadronization and fragmentation. This event generation is done by programs like *Pythia* or *PowHEG*.

The generated events are then put through a detector simulation that has knowledge of the exact detector layout and material composition. This step propagates the particles through different materials (also stochastically performed) and simulates corresponding detector responses as well as particle shower developments including noise effects. A powerful yet computing-intensive software program is *GEANT4*, whereas *Delphes* offers a reduced complexity and accuracy. [24, 25]

2.2 Astroparticle Physics

A new and unique approach of fitting multi-dimensional data was developed in the context of the Pierre Auger Observatory and is going to be presented in this thesis as the last of the three methods. This section provides an introduction to the physics behind the fit.

The past decade has seen great advances in the research on ultra-high energy particle physics. However, the origin of such highly energetic, natural particles remains unclear as it is hard to backtrack the path of charged particles through the magnetic field of our galaxy. The first sections are going to introduce the known theory behind ultra-high-energy cosmic rays (UHECRs) while the last part will introduce an observatory for UHECRs and its different components.

2.2.1 Cosmic Particle Accelerators

UHECRs are particles with energies above $10^{18}\,\mathrm{eV}$ and are expected to be extragalactic as there are no known acceleration sites within the Milky Way that could

provide such highly energetic particles. [26]

There are two approaches to explain these macroscopic energy scales of particles. The first class of models is called *top-down* models and are based on the idea that a decay of a super-massive particle can produce relativistic decay products. However, with recent discoveries, these models seem unlikely. Besides postulating instable super-massive particles, they fail to explain the increased fraction of heavier elements and the cutoff at the highest energies as well as the measured photon and neutrino fluxes. [27]

Nowadays researchers consider *bottom-up* models more probable which assume that particles get accelerated in an electromagnetic field produced by cosmic objects like pulsars, neutron stars or whole galaxies. Assuming that the particles increase their energy steadily by staying a long time inside such an electromagnetic field one can derive a relation between the maximum energy $E_{\max}$ when leaving this region, the magnetic field B of the region, the charge Ze of the particle and a source efficiency coefficient β.

$$E_{\max} = \beta Z \left(\frac{B}{\mu\mathrm{G}}\right) \left(\frac{L}{\mathrm{kpc}}\right) \mathrm{EeV} \tag{2.19}$$

This relation, called the *Hillas criterion*, is derived similarly as the computation of the bending magnet strength in a particle accelerator. The requirements for a UHECR with an energy of $E_{\max} = 10^{20}\,\mathrm{eV}$ are visualized in fig. 2.4. With a realistic

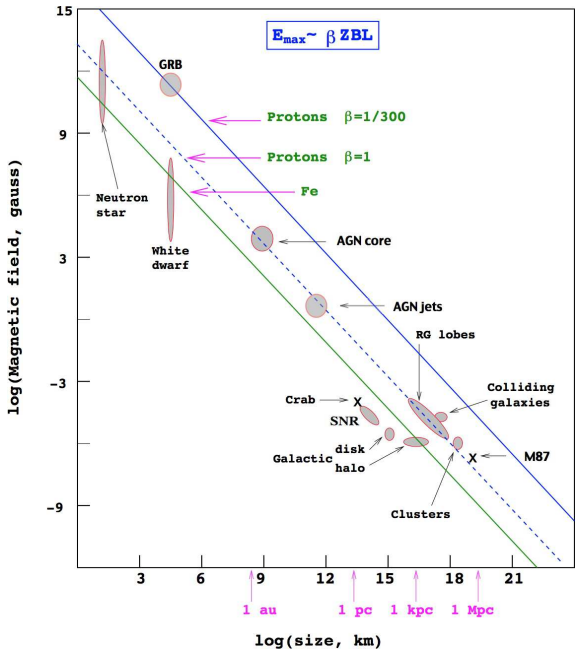

Figure 2.4: *Hillas criterion:* The requirements in terms of size and magnetic field under the assumption of a proton (blue) or iron (green) and an efficiency of $\beta = 1$ (dashed line is $\beta = \frac{1}{300}$). [27]

efficiency of $\beta = \frac{1}{300}$ only gamma-ray bursts (GRBs) seem a plausible generator of protons with an energy of $10^{20}\,\mathrm{eV}$.

The energy spectrum $N(E)dE \propto E^{\gamma}dE$ has an observed spectral index of $\gamma \approx -2.7$ which fits best to a first-order Fermi acceleration process based on a relative energy gain ϵ per cycle and a corresponding escape probability P_{esc} resulting in a spectral index of $\gamma = -1 + \frac{\ln(1-P_{esc})}{\ln(1+\epsilon)}$. With a process for which P_{esc} and ϵ both scale with $\frac{v}{c}$, like acceleration through strong shockwaves, a simulated spectral index fits well the observed value of -2.7.

2.2.2 Propagation of Cosmic Rays through the Universe

The propagation of cosmic ray (CR) is mainly influenced by particle interactions in space and the deflection by magnetic fields. Effects from gravitation can be safely neglected as this force is too weak and electric field components do not exist in open space.

Particle interactions The only relevant particles in space are photons as other particles have insignificant densities in open space with respect to their interaction cross section. The most important contribution is given by the cosmic microwave background (CMB) which provides enough energy to take part in a photo-pion or electron-positron production when blue-shifted into the rest frame of the UHECR.

$$p + \gamma_{CMB} \rightarrow \Delta^{+} \rightarrow \begin{cases} p + \pi^{0} \\ n + \pi^{+} \end{cases} \tag{2.20}$$

$$p + \gamma_{CMB} \rightarrow p + e^{+} + e^{-} \tag{2.21}$$

This is called the GZK-effect named after K. Greisen, G. Zatsepin and W. Kuzmin [28, 29].

For CRs over 10 EeV the photo-pion production is dominant. Using the mean energy of CMB photons and the mass of Δ^{+} and one can deduce an energy threshold for an efficient pion production in case of a proton CR:

$$E_{\min} = \frac{m_{\Delta}^{2} - m_{p}^{2}}{4E_{CMB}} \approx 2.7 \times 10^{20}\,\mathrm{eV} \tag{2.22}$$

This number just a rough approximation of the order of magnitude, as the exact numerical value is expected to be much smaller due to a large number of CMB photons having much larger energies than the mean $\langle E_{CMB} \rangle \approx 6 \times 10^{-4}\,\mathrm{eV}$. The overall energy loss can be simulated (see fig. 2.5) and makes the *GZK horizon* visible: UHECR protons traveling for more than 100 Mpc will never arrive at Earth with an energy higher than $10^{20}\,\mathrm{eV}$. Thus, the sources of UHECRs are expected to be nearby and consequently show some anisotropic behavior.

Magnetic deflection of charged particles occurs in strong magnetic fields. The extragalactic magnetic field is not well known but seems to be weak ($\approx 1\,\mathrm{nG}$) and only close to galaxy clusters up to $10\,\mu\mathrm{G}$. The coherence length is of the order of

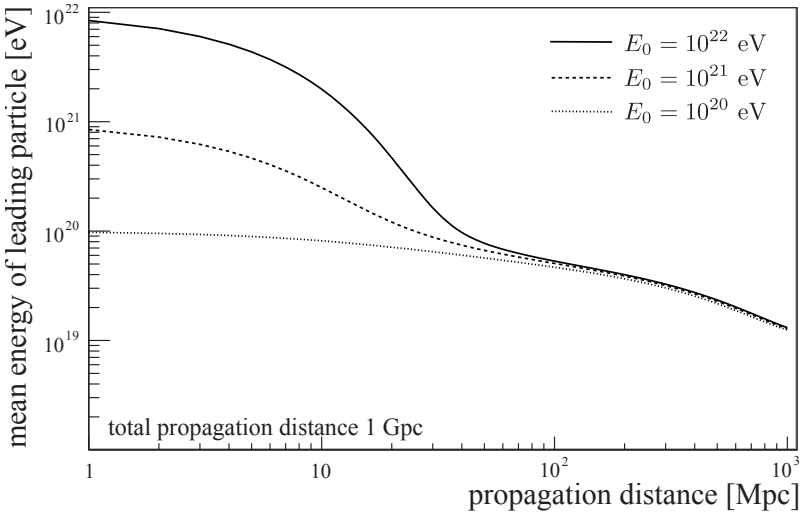

Figure 2.5: *GZK horizon:* The mean energy of a proton as a function of the traveled distance due to interactions with the CMB. [30]

a few Mpc and therefore much smaller than the Gyro radius of the UHECRs, thus leading to a neglectable deflection.

Overall the deflection through extragalactic magnetic fields is expected to be relatively small compared to the ones experienced inside our own galaxy. The total field strength close to our solar system is 5–6 µG leading therefore to deflection angles from a few degrees up to tens of degrees scaling roughly proportional to the inverse $\frac{1}{R} = \frac{1}{60\,\mathrm{EV}} \dots \frac{1}{6\,\mathrm{EV}}$ of the magnetic rigidity

$$R = \frac{E}{Ze} \tag{2.23}$$

of the CR [31]. More on the galactic magnetic field is presented in the upcoming section.

2.2.3 Galactic Magnetic Field

To predict particle trajectories through our galaxy and also being able to reconstruct extragalactic origins of measured UHECRs is an important task in Astroparticle Physics. The previous section has shown that inside our own galaxy the dominant component for the CR path is the galactic magnetic field.

Based on observations of other spiral galaxies we have a rough idea of the structure of the galactic field. However, as the Earth is inside the galactic disk there is no top-view information about the Milky Way available. The only information about the field can be inferred from a line of sight measurements which results in high uncertainties. There are different methods that can provide such measurements, e.g. optical polarization data, polarized synchrotron radiation and Zeeman spectral-line splitting in gaseous clouds with the Faraday rotation measurements providing

the most informative data. [32]

The measurements show that the magnetic field consists of a regular large-scale field and a turbulent component with a coherence length of around 50–150 pc [33]. The turbulent component can be modeled through a Gaussian blurring for the purpose of UHECR deflections as the scale of coherence is too small compared to the Gyro radius (which is roughly proportional to R).

The regular component of the field can be modeled in different ways and the corresponding parameters fitted to the observed data. One of these is the Jansson and Farrar model [34] (referred to as the JF12 field) which was published 2012 and reanalyzed on more recent data from Faraday rotations and synchrotron emissions. Its structure is based on observations of other spiral galaxies and, therefore, consisting of a spiral disk field with 10 parameters, a halo field with 6 degrees of freedom and another 4 for an X-shape field, resulting in 20 free parameters. Another commonly used model is named after Pshirkov and Tinyakov (PT11) [35] and is only based on Faraday rotations. In fig. 2.6 the measured data is compared to the model predictions and clearly shows, that PT11 does not describe the synchrotron observations as well as JF12. For CR with a high rigidity and paths perpendicular to the galactic disk these models provide good agreement, however for lower rigidities or flight paths inside the galactic plane, the exact modeling plays a more significant role and the model predictions do not coincide.

In conclusion, the JF12 provides a good description of observed data based on a reasonable model. However, it is not yet validated and should be considered a best guess, especially because of the discrepancies with respect to PT11.

2.2.4 Extensive Air Showers

UHECRs are quite rare ($5\,\mathrm{km^{-2}\,a^{-1}}$ CRs with $E > 10^{19}\,\mathrm{eV}$) making it intractable to measure them in space. In the Earth's atmosphere, these primary particles start to develop showers of secondary particles which partially arrive on the surface covering large areas and, therefore, making it feasible to detect and measure the UHECRs indirectly through their shower development. Thus, modern observatories for UHECRs like the Pierre Auger Observatory and the *Telescope Array* are built over large areas with a relatively sparse filling of the detector elements.

The extensive air showers (EASs) are created by consecutive particle interactions between the primary particle and air molecules producing secondary particles, as well as by secondary particles further decaying and interacting with the air. The emerging air shower consists mostly of a muonic, an electromagnetic and a hadronic component (see fig. 2.7).

It is important to note that the longitudinal development of the EAS is not depending on the traveled distance but the passed particle density $\rho(h)$ which depends on the vertical height above sea level. This notion is condensed into the definition of the *atmospheric depth*

$$X(h) = \int_h^\infty \rho(h')dh'. \tag{2.24}$$

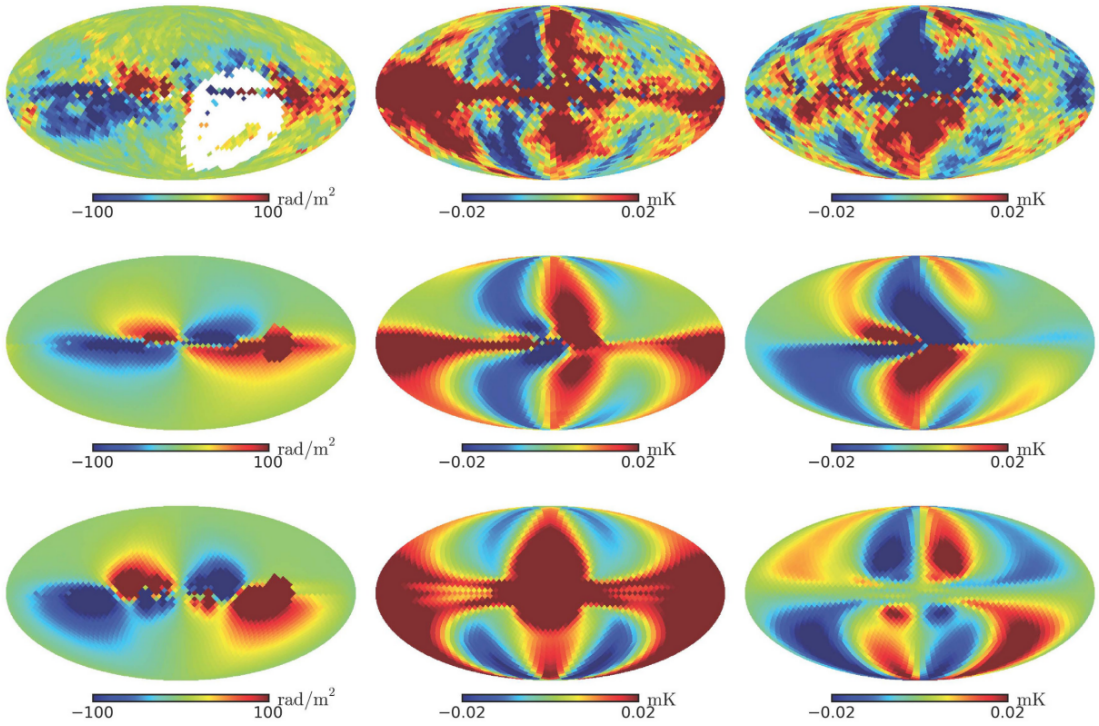

Figure 2.6: Models of the galactic magnetic field. *Left to right:* Faraday rotation measurements, Stokes parameters Q and U for the synchrotron emission. *Top to bottom:* Data, JF12 simulated, PT11 simulated maps. [36]

The *Heitler model* describes the electromagnetic cascade through a deterministic process of electrons and photons decaying into two particles (Bremsstrahlung and electron-positron pair production) after traveling through a fixed atmospheric depth X_{step}. In each of the decays, the energy of the initial particle is divided evenly on the two daughter particles (see fig. 2.8). The shower development stops when the energy of the particles falls below a critical value E_C making the decays non-efficient and resulting in an estimation of the atmospheric shower depth traveled until the shower reached its maximum size.

$$E_C \approx E_{n_{\max}} = \frac{E_0}{2^{n_{\max}}} \tag{2.25}$$

$$X_{\max} = n_{\max} X_{\text{step}} = \log_2\left(\frac{E_0}{E_C}\right) X_{\text{step}} \tag{2.26}$$

However, as this is a simplified model not even considering the muonic and hadronic components it gives only a rough idea of the scaling between $X_{\max}$ and the initial energy E_0. For a more accurate description, computational shower simulations are performed.

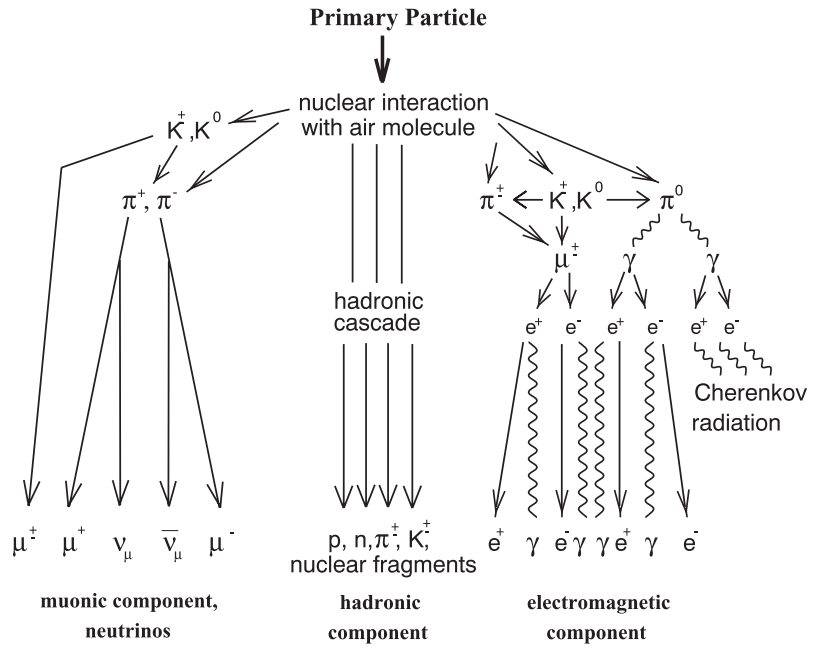

Figure 2.7: Schematic view of the development of an EAS and the decay processes of the 3 components. [37]

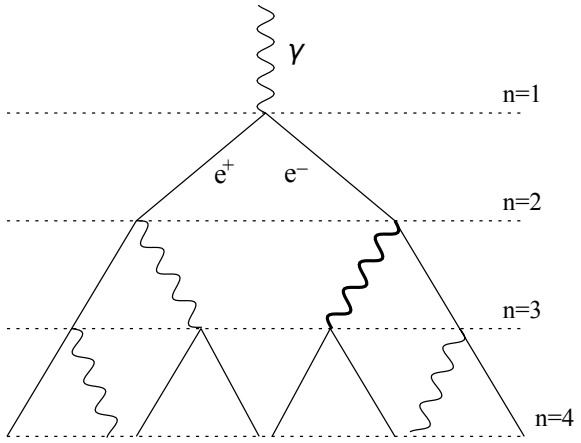

Figure 2.8: Heitler model of the electromagnetic cascade. It is assumed that at every decay the energy distributes evenly on the two daughter particles. [38]

2.2.5 The Pierre Auger Observatory

The Pierre Auger Observatory is the worlds largest detector for UHECRs covering an area of roughly 3000 km^2 in Pampa Amarilla, Argentina (see fig. 2.9). Most of the area is covered by two types of detectors complementing each other. [4]

Surface Detector One component of the observatory is the SD which comprises of 1660 water Cherenkov stations. They are situated on a hexagonal lattice with

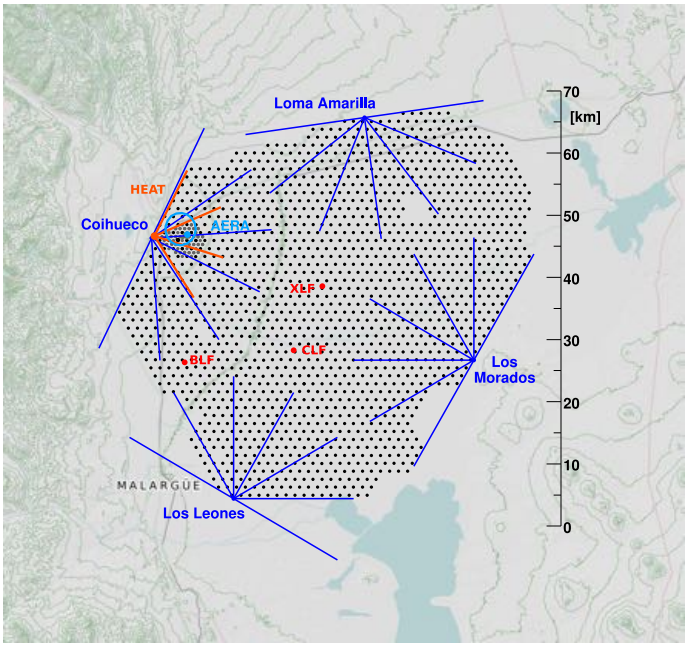

Figure 2.9: The Pierre Auger Observatory with the positions of the fluorescence detector (FD) (blue) and the surface detector (SD) (black). Additional detectors for lower energies are also located on the left (high elevation auger telescopes (HEAT), auger engineering radio array (AERA)). [4]

a separation distance of 1.5 km covering the complete area (see fig. 2.9). Each of these stations contains 12 000 l water in which Cherenkov radiation of penetrating relativistic charged particles is induced. With the stations being lightproof and reflective from the inside, the 3 PMTs installed in each of the stations can easily detect this radiation. As a shower typically hits 10 to 30 stations at once, the slightly different arrival times between the tanks can be used to reconstruct the arrival direction of the shower front which coincides with the direction of the primary particle (see fig. 2.10). The number of triggered stations gives relates to the energy of the CR and performing a cross-calibration with the FD results in an energy resolution of 10% to 24%.

Fluorescence Detector The FD consists of 4 telescopes observing the air volume above the SD (see fig. 2.9). These telescopes measure fluorescence light emitted from excited nitrogen molecules. Thus, they are able to reconstruct the shower profile in the air and can also correctly predict the incoming direction of the CR. This is done in two steps. First, the geometric shower detector plane is computed using a line fit of the activated pixels in the telescope. Afterward, the arrival times of the photons can be used to determine the angle of the shower inside this plane (see fig. 2.11). The longitudinal profile including information on $X_{\max}$ can be used to reconstruct an energy estimate. Therefore, the FD and the SD complement each other very well. The downside of the FD is, that it needs to be highly sensitive to the fluorescence light and can only be operated in the dark with a clear sky resulting in an average

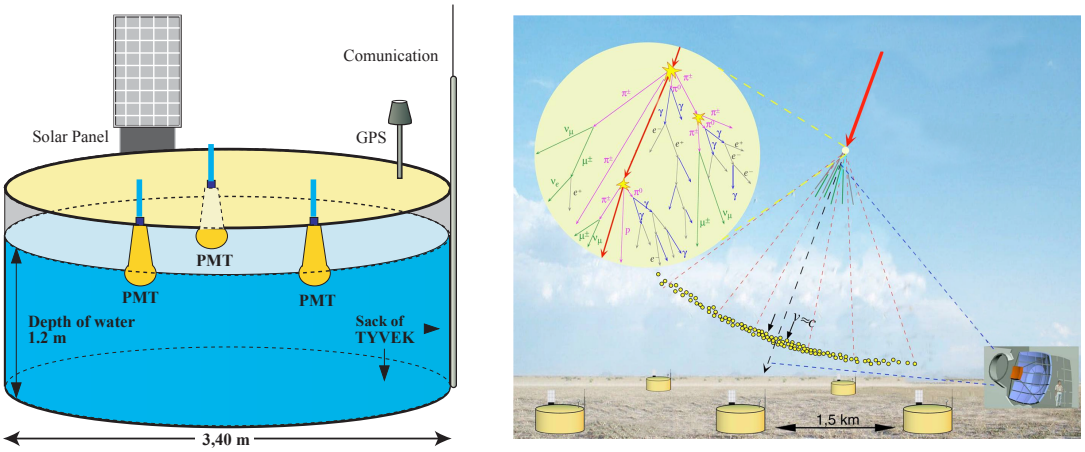

Figure 2.10: *Left:* Sketch of an SD station. [37] *Right:* Visualization of a shower front measurement by the SD and the FD (lower right corner). [39]

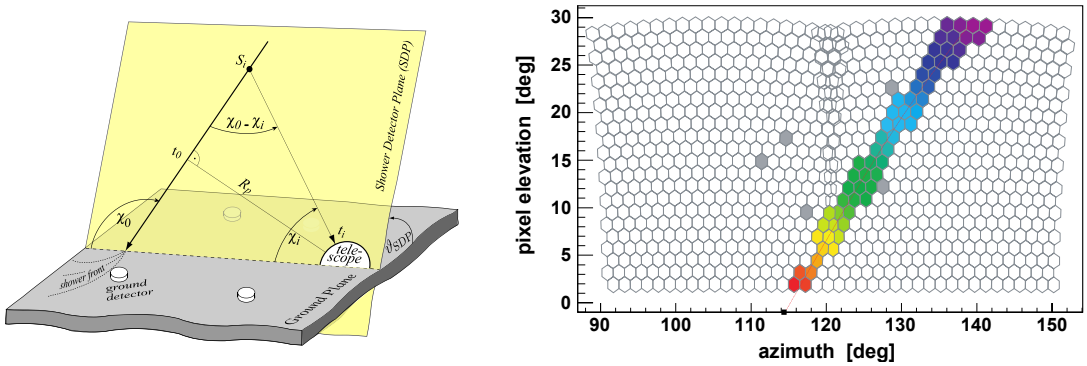

Figure 2.11: Schematic view of how the shower direction can be reconstructed with the FD. *Left:* Visualization of the shower plane and the arrival times of photons. [40] *Right:* Data from a fluorescence telescope showing the shower plane. The colors indicate arrival times, which are used to estimate the direction inside the shower plane. [30]

uptime of 13%.

3 Deep Learning

For a few years now, Machine Learning (ML) concepts like Boosted Decision Trees (BDTs) and shallow Artificial Neural Networks (ANNs) have been used to develop powerful physics analyses. The essence of ML is to build generic models with many tunable parameters and find a good set of parameter values to describe some desired function that is intractable to implement explicitly. Instead, this function is usually indirectly described by a large dataset, called training data, consisting of the input data of the function and the corresponding desired output, called the label of the input. The parameter solution of the model is usually found not only by using massive datasets but also the fast computation speed that modern computers offer and state-of-the-art optimization techniques to direct the search.

DL is a subfield of ML and focuses on ANNs with multiple layers. This area of research became very popular around 2010 with the rediscovery of Convolutional Neural Networks (CNNs), the Rectified Linear Units (ReLUs) as activation function and the use of graphics processing units (GPUs) for parallel and therefore faster optimization. The following sections introduce and explain the most important aspects of DL used throughout this thesis.

3.1 Basics

This section provides an overview of the most fundamental aspects of evaluating and training an ANN.

3.1.1 Networks

ANNs are generic models with many parameters that are loosely inspired by the basic inner workings of human brains.

An ANN usually consists of artificial neurons, often called nodes or perceptrons, and of connections between those nodes which model the synapses of the brain. The nodes are most of the time not parameterized by any numbers but may differ by an internal activation function $g(x)$, which is often taken to be the same for all nodes in the network. The connections are associated with a single real parameter, called the weight of the connection.

A single node, as shown in fig. 3.1, has many input nodes and a single output value. The output value a_{out} is computed by taking the activation values of the incoming nodes a_i multiplied by the connection weights w_i and summed up to one real number z. This is fed into the activation function $a_{out} = g(z)$ of the node and results in the final activation of the node which in turn is then used by the connected nodes following this one. The activation function is an important part of an ANN as it is the only part that introduces a non-linearity. Without its presence, the chaining of nodes would still result in an overall linear transformation and therefore inhibit an overall complex mapping from the input to the output. For a similar reason, the so-called bias b of a node is also essential. Through the created constant offset in

the input of the activation function, it gives each node the ability to move the region of the non-linearity into the right position with respect to the average of the other incoming activations.

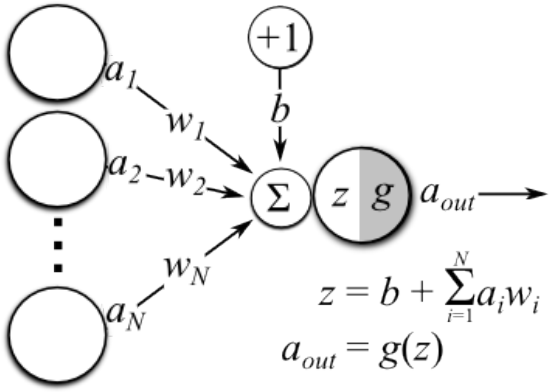

Figure 3.1: A simple node of an ANN and its inner workings. [41]

Nowadays there are many different architectures combining many nodes and connections into larger networks which are capable of representing any functional dependence by adjusting the weight parameters accordingly.

One common and fundamental network architecture is the dense network (see fig. 3.2). A dense network organizes the nodes into multiple layers which are called input, output or hidden layer depending on their role. The input layer consists of as many nodes as there are dimensions in the input data of the task. Each of these nodes represents a specific variable and takes its value as activation when evaluating the network. Following the input layer are some hidden layers each with a certain number of nodes arranged. A node from one layer is connected to all nodes from the previous layer resulting in dense connection patterns giving this architecture its name. The final layer is called the output layer and has as many nodes as the task output has dimensions. After evaluating the network from the input layer through the hidden layers to the output layer, the activation of this layer is taken as the network output. The activation function for this output layer is chosen according to the task at hand (classification, regression, etc.) and differs from the activation functions inside the network.

An important remark on this architecture is that the computation can be performed very efficiently using matrix multiplication. This is one of the reasons why GPUs boost the runtime of neural networks as they have special circuits for linear algebra operations. Using the index $l = 1 \dots n_l$ to specify the layer one can write down the equation:

$$a_{i,l+1} = g\left(b_{i,l} + \sum_j w_{ij,l} \cdot a_{j,l}\right) \tag{3.1}$$

$$\Rightarrow \qquad \boldsymbol{a}_{l+1} = g\left(\boldsymbol{b}_l + W_l \boldsymbol{a}_l\right) \quad \text{with } (W_l)_{ij} = w_{ij,l} \tag{3.2}$$

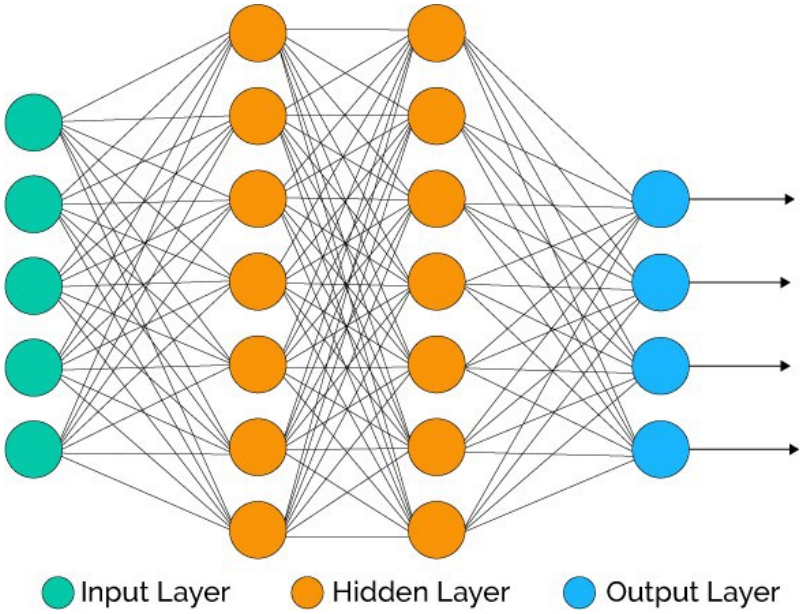

Figure 3.2: Network layout of a dense architecture. [42]

In the domain of image-like data, the CNN is an extremely successful network layout [43]. As it is not used in this thesis I will only point out some important aspects for its success. As high-resolution images have an incredibly high dimensionality ($\approx$ 1000pxl x 1000pxl x 3 colors) dense network are impractical because of their large amount of parameters. However, most of the time the investigated images have a relatively low information content that could potentially be encoded in much fewer dimensions. The CNN is particularly good in reducing the data dimensionality without loss of information by exploiting local correlations and the translational symmetry of images. This symmetry is encoded in each of the convolutional layers by weight-sharing (e.g. specific connections have by definition the same weight) and dropping of connections reducing the number of model parameters significantly while retaining a high performance. Multiple convolutional layers are then stacked with each one decreasing the dimensionality (= number of nodes) and increasing the abstractness of information.

In physics, a common application of CNNs is the processing of data from tracker and calorimeter setups which also feature pixel-wise data entries. Since CNNs have shown how powerful it can be to exploit prior knowledge of the data there is a growing interest in developing physics-specific networks incorporating physics knowledge like Lorentz-invariance and boosting of four-vectors.

As stated in the beginning, the goal of DL is to have a general model (the Deep Neural Network (DNN)) that can represent any functional dependence. This statement is theoretically proven to be true for the dense layout and a certain class of activation functions in the famous 'Universal Approximation Theorem' [44]. Subsequently, the next important step is finding these optimal parameters. The following three sections are going to introduce the basic ingredients for network optimization.

3.1.2 Loss Functions

The parameter optimization is done similar to classical fits using gradient descent over a certain quantity, e.g. the sum of squared errors in the case of a linear regression. In DL this quantity is called the loss function which is to be minimized. Consequently, large losses encode a bad performance of a network. The scale of the loss is usually chosen to be on the order of 1 with the theoretical minimum for a perfect model set to 0. This convention guarantees that one can use similar hyperparameters[3] in the training of a DNN. The two most commonly used loss functions will be presented here.

As for classical regression, the mean squared error (MSE) loss is also used in DNN setups. If $\boldsymbol{o} \in \mathbb{R}^N$ is the output vector of the DNN and $\boldsymbol{l} \in \mathbb{R}^N$ the target vector, or more commonly named the sample label, then the MSE is defined as

$$MSE = \frac{1}{N}(\boldsymbol{o} - \boldsymbol{l})^2. \tag{3.3}$$

As required by the convention its minimum is given for a perfect match $\boldsymbol{o} = \boldsymbol{l}$ and evaluating to 0 in this case. The scale of the MSE behaves approximately quadratic with respect to the scale of the label $\boldsymbol{l}$. An upcoming section (3.2.1) will introduce value normalization as an important preprocessing step resulting in $\boldsymbol{o}, \boldsymbol{l}$ and with them also the MSE being in the order of 1.

An extremely common task for DL not only in physics is classification. In these tasks, one usually has as many output values as there are classes. To obtain a probability vector whose entries sum up to one, a softmax activation is applied to the output vector $\boldsymbol{o} \in \mathbb{R}^K$.

$$\boldsymbol{p} = \text{softmax}(\boldsymbol{o}) = \frac{e^{\boldsymbol{o}}}{\sum_i e^{o_i}} \tag{3.4}$$

For classification tasks the cross-entropy loss is the most common loss function. The label is a class index $1 \dots K$ encoded into a vector of length K containing a single 1 at the corresponding position with the other entries being 0. This is a so-called one-hot encoding. The loss can then be written as

$$H = -\sum_i l_i \ln p_i = -\boldsymbol{l} \cdot \ln \boldsymbol{p}. \tag{3.5}$$

Again this loss behaves according to the convention by being 0 ($= 1 \cdot \ln 1 = \lim_{\epsilon \to 0} \epsilon \ln \epsilon$) for a perfect model $\boldsymbol{p} = \boldsymbol{l}$ and on the scale of 1 in the maximal confused case $p_i = \frac{1}{K}$ ($H = \ln K \lesssim 3$ for $K \lesssim 20$). Besides $\boldsymbol{p}$ having the properties of a discrete probability distribution ($p_i \geq 0$, $\sum p_i = 1$) a trained DNN actually produces numbers that should be close to the true probability. We can understand this theoretically by looking at input data that could be described as equivalent ($\boldsymbol{x}_i \approx \boldsymbol{x}_0$, $i = 1 \dots n$) concerning the classification task at hand. The DNN is

[3] Hyperparameters are usually numerical values that steer the training but are not part of the ANN. More on this in sec. 3.2.6.

producing roughly the same output $\boldsymbol{p}_i \approxeq \boldsymbol{p}_0$ in all these cases as it is expected to generalize well. Based on the training labels l_i which might actually differ because of noise in the data, one finds the optimal value

$$\boldsymbol{p}_0 = \arg\min_{\boldsymbol{p}_0} H = \arg\min_{\boldsymbol{p}_0} \sum_{i=1}^{n} H_i = \arg\min_{\boldsymbol{p}_0} \sum_{i=1}^{n} -\boldsymbol{l}_i \cdot \ln \boldsymbol{p}_0 = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{l}_i \tag{3.6}$$

confirming that the vector $\boldsymbol{p}_0$ is representing the probability of the class occurrences for a similar input $\boldsymbol{x}_0$.

An important aspect of losses is also their relative weighting. For example, it is simple to incorporate sample weights which are multiplied with the loss value for each data sample. These weights can be used to indicate which samples are less affected by noise and therefore should stronger influence the training with respect to more noisy data. In classification tasks, they are also often used to compensate the number of class occurrences. This is preferable for the training process as it avoids that the training simply results in an overall constant output with a high accuracy just because the input data consists mostly of samples from one class with all other classes being underrepresented.

Loss functions guide the gradient descent and are therefore extremely important. A carefully chosen loss can be much more effective than hyperparameter tuning or the right network layout.

The gradient descent not only needs a quantity to minimize but also the gradient of this loss with respect to the parameters. In low-dimensional problems, it is often feasible to compute the gradient numerically by evaluating the target function multiple times at different parameter space points. However, this is impossible for a large number of parameters (DNNs have 100,000 and more). This is where the backpropagation algorithm comes into play.

3.1.3 Backpropagation

The backpropagation algorithm allows the efficient computing of symbolic gradients for a lot of parameters and is based on the chain rule for derivatives.

To understand how it works it is helpful to think of the network as a concatenation of multiple small functions taking some inputs and returning a single real number. The input values consist of the parameters w one wants to tune as well as of outputs from previous function evaluations. The goal is to compute the gradient $\frac{\partial \mathcal{L}}{\partial w}$ where $\mathcal{L}$ is the loss. The numeric value of this derivative is then used for the gradient descent. The way this is done is by going through the network from the output to the input collecting all gradients. In the backward pass we receive for each small function building block $f(w, a_1, a_2, \ldots)$ the numeric value for $\frac{\partial \mathcal{L}}{\partial f}$ and with the knowledge of the functional gradients $\frac{\partial f}{\partial a_i}$ one can compute

$$\frac{\partial \mathcal{L}}{\partial a_i} = \frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f(w, a_i)}{\partial a_i} \tag{3.7}$$

by simple multiplication using the chain rule. This gradient is then passed further back until it reaches a network parameter which is updated according to the exact implementation of the gradient descent. This procedure is also depicted in fig. 3.3.

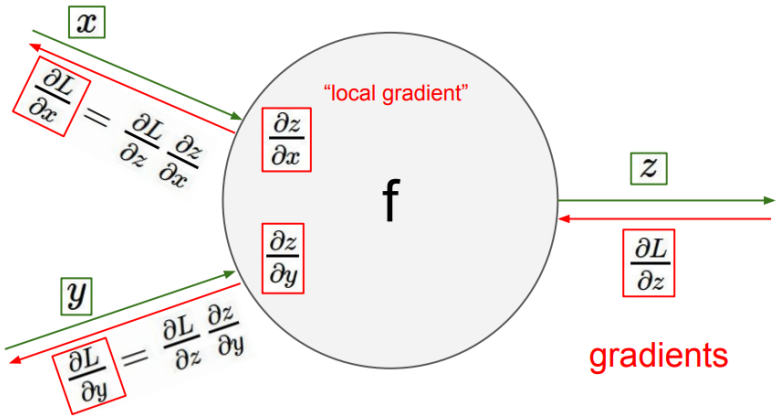

Figure 3.3: Schematic representation of the backpropagation algorithm. The forward pass of the function block is shown in green. The backward pass using the chain rule is shown in red. The numerical value in the backward pass is the gradient of the loss L with respect to the forward flowing variable. [45]

In practice, this procedure is realized by a software framework (see sec. 3.3) that has the information of the symbolic gradients for every function used inside an ANN.

ANNs and backpropagation methods have been known for some time. One of the main reasons for the recent success of DL is the usage of semi-linear activation functions inside the networks. The hyperbolic tangent or the sigmoid activation functions were both very popular choices before 2010. These functions perform not particularly well for deep networks as the gradients are smaller than 1 (for the sigmoid the gradient is smaller than $\frac{1}{4}$). In the backpropagation step, this leads to the vanishing gradient problem for deep networks as these gradients get multiplied with each other and reach the first layers with small gradients. The opposite of this behavior is the exploding gradient problem when gradients larger than 1 are multiplied in the backward pass. Nowadays there are a handful of activation functions[4] all featuring the identity for high input values and therefore enabling (next to other recent developments) the successful training of deep networks. A very simple yet successful activation is the ReLU $\text{relu}(x) = \max(0, x)$ which has the additional advantage of being extremely efficient to compute. [46]

3.1.4 Gradient Descent

With all the ingredients for the gradient descent (i.e., the loss and the gradient computation) this step itself is in principle simple. However, there are some practical

[4] Common activation functions converging towards the identity for high input values: ReLU, Leaky ReLU, SELU, ELU, SoftPlus

details which are not necessary when working with low-dimensional parameter spaces and small datasets.

As an update step changes millions of parameters at the same time it does not make sense to take the size of this update step to be in the order of the gradient itself. Before applying the gradient it gets multiplied with the learning rate of the training, a very important hyperparameter (see sec. 3.2.6) which is usually in the order of 10^{-5} to 10^{-3}.

Also, because of the typically large datasets that are processed, it is not feasible to compute the gradient over all data points and apply it. Instead one divides the data randomly into smaller batches. Then the gradient is computed on a batch and directly applied before processing the next batch. This is called stochastic gradient descent and has proven to be more successful than the traditional gradient descent. This is not only due to the large datasets but the fact that DNNs tend to have quite a large solution space. In this space exist many local minima as well as many regions where the gradient across all data is close to zero, therefore reducing the learning effect. The stochastic effect of batching mitigates these problems and effectively turns the batch size into an additional hyperparameter.

With this ingredient, it is finally possible to train large and complex DNNs, however, some more advanced concepts can further improve the training and performance.

3.2 Advanced Concepts

This section will present advanced concepts used in the setups presented by this thesis but which are generally applicable for DNNs.

3.2.1 Normalization and Initialization

The normalization of data is quite important for DNNs. Networks tend to perform better on standard normal-distributed-like data. The input data and regression labels are usually normalized feature-wise to have zero mean and unit variance sometimes applying a logarithmic transformation to reduce the dominant effects of long tails in the respective feature distribution. Additionally, with this general preprocessing rule, a lot of well-tested default values for hyperparameters can be used (e.g. learning rate) and will work well without the need of an extensive hyperparameter search.

Also, other technical details profit from data normalization, e.g. the initialization of the connection weights. If one would naively initialize all network parameters to an equal value, all subsequent gradient update iterations will behave exactly the same for all parameters. Thus, parameters should be initialized randomly to destroy symmetries in the network. However, the ideal scale for the random values depends strongly on the scale of the input data. The normalization of the data, therefore, eases the initialization of the weights. A very common initialization is the *He initialization* [47] in combination with ReLUs as activation. The *He initialization*

of a network weight is

$$w_{ij,l} \sim \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{2}{n_l}}\right) \tag{3.8}$$

with n_l being the number of nodes in the layer l and $w_{ij,l}$ being the connection weights from layer l to $l+1$.

Both the normalization and initialization are important to reduce the effects of vanishing or exploding gradients. They also make sure that the incredibly large parameter search space can be reached and reasonably used during learning, therefore leveraging the most important advantage that DNNs have to offer.

3.2.2 Optimizer

With the development of software frameworks, it became easier to use more advanced learning algorithms, called optimizers, as it became simpler to switch between different implementations. All are based on the principles of gradient descent with the most basic one being the gradient descent itself.

More advanced optimizers often use so-called momentum to also take the history of parameter update steps into account. With $\Delta \boldsymbol{w}_i$ being the i-th update step of the weights $\boldsymbol{w}$ and $\boldsymbol{g}_i$ being the gradient of the weights with respect to the loss, the update steps are computed as

$$\Delta \boldsymbol{w}_i = \boldsymbol{g}_i + \alpha \Delta \boldsymbol{w}_{i-1} \tag{3.9}$$

with α being a number between 0 and 1. This helps the optimizer to overcome regions where the loss is almost constant ($\boldsymbol{g}_i \approx 0$) and reduces noise in the update steps resulting from small batches.

Based on the momentum optimizer is the very popular Adam optimizer [48]. While employing momentum for multiple internal variables it also takes into account which parameters get larger gradient updates. A parameter that has often evaluated a gradient close to 0 in the past means for the optimizer that its value can be changed without having a negative effect on the already trained samples. Especially, if there is one single event where this weight plays an important role Adam will detect this and do a stronger update of this specific variable. This makes the Adam optimizer a general yet powerful and particularly fast optimizer which is useful in the development process of DNNs. However, for training a production-ready network the method of choice is often the basic gradient descent algorithm as it tends to find better solutions in the converged state. On the downside, it is a few orders of magnitude slower.

3.2.3 Training and Validation

DNNs are very capable of capturing fine nuances in large datasets. After a good training run, it is not unusual to get a lower loss or higher accuracy than one

might have initially expected based on the noise in the data. This phenomenon in ML is called overfitting and describes the effect that a model started memorizing single data samples and their corresponding labels. This is counter-productive for a good model which is supposed to be a good generalization of the data. Figure 3.4 visualizes the effect and also gives an intuition of why such a model can actually perform worse than a well-generalizing solution.

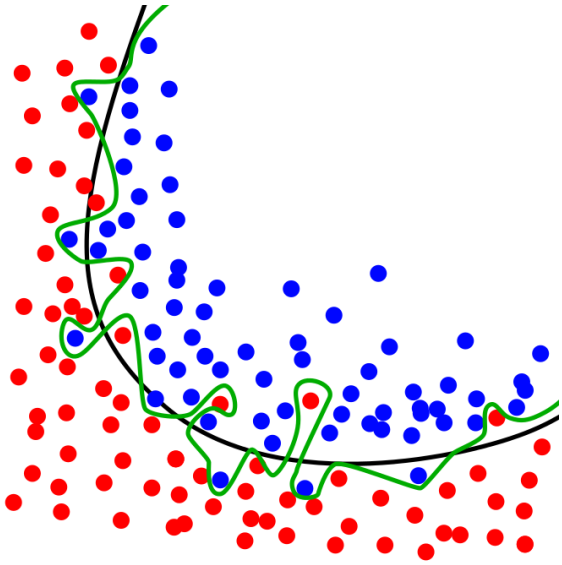

Figure 3.4: Overfitting: The green line represents an overfitted model to the shown data. It is apparent how a general description would be the black line and therefore perform better on unseen data. [49]

When presenting the performance of a network it is important to avoid this bias by not reporting the lowest observed loss during training but a loss that was generated using a validation dataset on which no training was performed. Therefore, it is very common to split the dataset into training data and validation data.

The validation set makes it possible to give a reasonable estimation of the model performance after the training and can also indicate whether the model is experiencing severe overfitting or not (cf. sec. 3.2.5). The next section introduces some methods that can be applied to pro-actively reduce these issues and therefore produce better generalization results.

3.2.4 Regularization

Over the past years, multiple successful regularization schemes were developed and are now widely used in the DL community. Regularization helps the network to avoid overfitting and focus more on the aspect of generalizing data. This is especially important for sparse or small datasets.

A very intuitive regularization is the adaptation of the network size. By reducing the number of layers and nodes in a network its ability to overfit is reduced. De-

creasing the DNN size also results in a lower model complexity which could make it impossible for the network to capture some intricate yet general aspect of the data. It is, therefore, more common to construct larger networks and apply additional regularization.

The L2 regularization is a loss term that is easily implemented as it is the sum of squares of all parameters.

$$\mathcal{L}_{tot} = \mathcal{L} + \lambda_R \cdot \frac{1}{2} \sum_i w_i^2 \tag{3.10}$$

The strength of this regularization can be tuned using the parameter λ_R. The basic principle here is the addition of a second goal (small weights) which is independent of the target goal. As memorization of data should be more difficult to encode than general rules this memory effect is the first that would be traded in to achieve small weights while still retaining an overall good performance through the general rules.

3.2.5 Early Stopping

Another simple way of reducing overfitting issues is a concept called early stopping. When tracking the validation loss (loss evaluated on the validation data) parallel to the training loss, it is possible to determine after which iteration the validation loss is not improving anymore or even getting worse. This point would mark the moment where the network stops successfully generalizing information and starts memorizing the dataset. Figure 3.5 depicts such a scenario. By looking at the last few recorded validation loss values one can automatically trigger an early training stop and not only reduce overfitting effects but also effectively reduce the computation time. This is especially useful during hyperparameter searches (see sec. 3.2.6) where the number of iterations until convergence strongly depends on those parameters. The presented dynamic early stopping allows to stop trainings that would otherwise stay in a converged state for as long as by the user initially specified.

3.2.6 Hyperparameter Optimization

As the previous sections have shown almost any method or technique comes with its own set of hyperparameters that can be tuned and adjusted to improve the overall training procedure regarding computation speed or accuracy. Hyperparameters of the training are basically all configurable values that are not trained model parameters, like the learning rate, the used optimizer, batch size, regularization strength, network size and the used activation functions inside the network. For most of these, good default values exist that work on a wide range of tasks. Still, by tuning the parameters to get the optimal training procedure with the best accuracy often improves the result with respect to the default settings. This fitting can also be computationally optimized using tools like the python module *skopt* [50] which aims at making good next guesses for hyperparameters by only using a handful of pre-evaluated training runs and their performance values. For the presented models

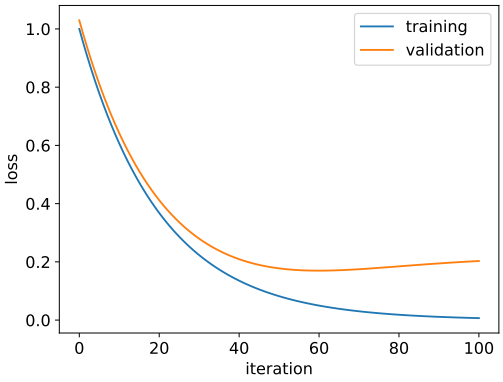

Figure 3.5: An example of how the evolution of the loss could look like. In blue, the loss history of the training data is shown and as expected steadily decreases through the use of gradient descent. Around iteration 50 the loss evaluated on the validation data starts to increase signaling that the model starts to overfit on the training data. This would be a good point to stop the training as the model is not learning any new general information.

the optimization of hyperparameters was always performed, however, depending on the runtime of the training it is often done manually over one parameter at a time when there is no correlation of hyperparameters to be expected.

3.2.7 Adversarial Networks

A quite novel network layout called a Generative Adversarial Network (GAN) [51] was introduced to the DL community about 4 years ago. With this publication, two new concepts were introduced. The first concept is about using DNNs as functions that can generate random data according to some distribution. The main idea is here simply to feed random values as inputs to the network instead of some specific training data. A popular application is a task of mimicking a given unlabeled data distribution through the network function and its corresponding random input. This is called an unsupervised training task as opposed to supervised learning with the presence and usage of labels.

To train such a generator network one needs some kind of loss function that tells the network how good the currently generated data matches the distribution of the training set. Here is where the second key idea of GANs comes into play, an adversarial network. The adversary is basically a complex loss function that is able to say if the generated data looks *real* (e.g. similar to the training data) or *fake*. This network has to be trained as well, however, from the formulation of the goal one can already clearly see that this is a simple classification task. As a GAN variation is used in the first setup and an adversarial approach is followed in the second setup this section will introduce the GAN in more detail.

Figure 3.6 presents the basic setup of a GAN. The generator network receives a

random noise input following some normalized random distribution (e.g. uniform or standard normal distribution). This is called the latent vector or latent space P_z and is transformed by the generator into a different data distribution parameterized by the weights of the network. The adversary network is a simple discriminator and is trained on the generated data and the training data. The labels are artificially created depending on the data coming from the generated distribution or the training data (*fake* or *real* data). Training the discriminator with the usual classification loss (cross-entropy)

$$\mathcal{L}_D = -\sum_{i \in \text{real}} \ln D_i - \sum_{i \in \text{fake}} \ln(1 - D_i) \tag{3.11}$$

one can use the trained discriminator to formulate the loss for the generator. A theoretical approach to constructing a generator loss can be taken when viewing the process of the adversarial training as a Minimax game. The discriminator tries to determine *real* and *fake* data while the generator is trying to maximally confuse the discriminator. The value of confusion or discrimination can be measured with the discriminator loss which is automatically taken to be minimized by the discriminator. Therefore, a sound generator loss would be exactly the negative of this so that the generator actually tries to increase the discriminator loss. As the generator can only influence the *fake* dataset the sum over the real data in the loss is usually dropped as its derivative with respect to the generator weights is 0 leading to the generator loss

$$\mathcal{L}_G = \sum_{i \in \text{fake}} \ln(1 - D_i) = \sum_{z \sim P_z} \ln\left[1 - D(G(z))\right]. \tag{3.12}$$

In practice, this loss turns out to be not very useful because as soon as the discriminator has some idea of samples that it can safely label as *fake* data these samples receive an extremely low gradient making it impossible for the generator to learn. The first GAN publication [51] directly introduced an ad-hoc improvement of this loss

$$\mathcal{L}_G = -\sum_{z \sim P_z} \ln D(G(z)) \tag{3.13}$$

which makes training feasible.

The training procedure of this network is very similar to standard DNNs. The most important difference is that there are two networks whose weights need to be updated according to different loss functions. Effectively another hyperparameter n_D is introduced describing how many update steps are performed for the discriminator network until the generator does a single training step. This controls the relative power of the discriminator over the generator and is important as a strong discriminator just rejects all generated data whereas a weak discriminator might not provide helpful information at all. A discriminator that is on par with the generator does help to push the *fake* samples closer to the *real* training distribution.

Still, this first version of a GAN has some weak points. The first setup will actually introduce a modification of this first GAN and discuss these problems and how they improve in the modified version.

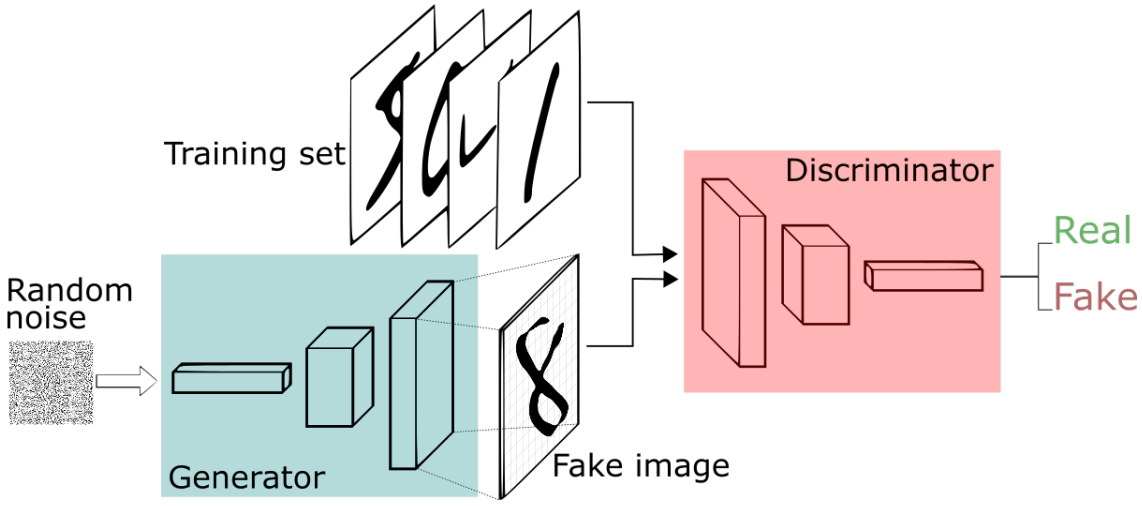

Figure 3.6: This diagram explains the basic principles of generative adversarial training with the example of generating images of handwritten digits. A random vector from the latent space is fed into the generator which produces a *fake* image. The discriminator is trained on the *real* training set and the images provided by the generator with the task to discriminate between these two classes. [52]

3.3 Software

As discussed in sec. 3.1.3, it is important to have a good software framework that automatically constructs the local gradients of functions which are then used for backpropagation. TensorFlow has become very popular for this task and is shortly introduced.

In physics, there is usually more data processing involved compared to other sciences using ML. To organize the DNN training as part of a bigger data processing workflow, the frameworks of Luigi and Law are extremely helpful.

3.3.1 TensorFlow

TensorFlow [53] is an open-source project started and developed by Google employees which aims at making training of DNNs easy and also efficient. It provides a framework that is able to construct symbolic gradients automatically. Additionally, it delivers an optimized interface between the software and the hardware (e.g. CPUs and GPUs). This allows users to efficiently use the provided hardware without explicitly writing code for it, therefore accelerating the development cycle of DNNs.

The framework implements tools users can use to construct so-called computation graphs. These tools are simple building blocks like matrix multiplication, addition, division. More complex functions do exist, however it is simple to use elementary operations to construct most of the more complex functions one needs. Each block internally encodes the computation used for the data flow through the network but also has knowledge of its local gradient function which is used for the backward

pass of the gradients. Additionally, some blocks provide estimated metadata telling TensorFlow how efficient this task can be run on GPUs or CPUs and what the corresponding optimized machine instructions for either of these two platforms are.

3.3.2 Luigi and Law

Luigi [54] is a python open-source software framework developed by employees from Spotify. This tool lets you build up a workflow by chaining smaller tasks. Therefore a task usually consists of a 'run' method telling the computer what work should actually be done. Additionally, there is a 'require' method that returns a list of task outputs that are required by this task. Finally, it has an 'output' method to inform Luigi which output files it should expect to be produced. This framework not only helps to keep the own code modularized and reusable but also automates dependency-resolving, therefore, running a complete analysis is as simple as just executing the last task in the chain. Luigi will then take care of running all tasks that are required while checking if the outputs already exist and consequently do not need to be run again.

Law [55] is an additional open-source project built upon Luigi and developed by Marcel Rieger explicitly for the purpose of physics analysis. Besides the very useful command line tool, it provides utilities and base tasks that make it simple to use external resources like remote file systems or computing resources.

4 Event Simulation with Generative Adversarial Networks

As discussed in sec. 2.1.4 for the CMS experiment, simulated events with a known ground truth are important for developing and optimizing discriminators in modern physics analyses.

Such a simulation of events in the CMS detector is computationally expensive and requires several seconds of computation time per event [25]. The future upgrade of the LHC becoming the High-Luminosity LHC (HL-LHC) and the resulting higher event yield will lead to a proportionally higher request of MC events for new analyses. Simultaneously, more CPU power needs to be spent on data selection and reconstruction resulting in an expected future computing resources requirement far greater than the anticipated increase through Moore's Law or new computing architectures [56]. Thus, a lot of research is done for finding solutions to this challenge for example by producing MC events in a more efficient way.

With the recent successes of GANs and CNNs first steps were taken by applying these methods and creating a DL version of a calorimetric shower simulation [57]. The publication presented promising results including speedups of up to 5 orders of magnitude compared to a traditional *GEANT4* simulation.

In this section, an improved version of the traditional GAN will be applied to the task of simulating not shower images but directly the high-level physics observables for an event. This investigation stands in stark contrast to most other GAN applications with respect to the type of data as it is very unusual to deploy GANs on non-image data. Probably one of the main reasons for this is the fact that image-like data with geometric structure is the only high-dimensional data humans can grasp and evaluate quite easily, therefore, being able to check and validate the approach of a generative model.

The high-level physics observables cannot be easily visualized for humans to make it obvious which events are physical and which are not. Additionally, it is much more important, that the exact data distribution is recreated instead of only producing a subset of events. Therefore, some metrics are introduced to evaluate the generative models of this approach in a comprehensible and quantitive way.

The following section will first present a generalized formulation of the task. Afterward, the Wasserstein-GAN (WGAN) will be presented and its way of functioning explained. Finally, three metrics are introduced to evaluate the trained models.

4.1 Generalization

The task is to produce a dataset consisting of entries (=events) which themselves consist of a fixed set of numerical features (=physics observables). These data entries shall follow a certain high-dimensional phase space distribution according to a given physics model. This is usually achieved by encoding the physics model in an algorithmic implementation (e.g. *GEANT4*) which takes some random seed value

and produces deterministically these events.

The goal is to translate the encoded physics model into a DNN which takes something similar to a random seed as input and produces the events with its observables as output. The MC simulation is used to produce a big enough amount of data indirectly encoding the target phase space distribution. This dataset is then used to train a GAN and therefore encoding the physics into the DNN.

For evaluating the trained model its generated data will be compared to the original training events using different statistical metrics.

4.2 Improved Wasserstein-GAN

First investigations producing MC simulated data using DL were performed using the originally proposed GAN (see sec. 3.2.7) without any modifications. Two main issues were immediately observed:

Training Instability It was important that none of the two involved networks could easily outperform the other. More often this happened to be the discriminator easily identifying unrealistic data as 'fake' in larger phase space regions and therefore providing the generator network with no helpful information on how to improve. This often leads to a stagnation of the training before the generator network could even come close to a good solution.

Mode Collapsing The generator network tends to produce very similar looking events and completely ignoring other phase space regions which are populated in the training data. This phenomenon is called mode collapsing and arises due to the fact that the generator loss is not effectively punishing this behavior.

Both problems were observed by many others in the ML community and got addressed by more recent publications. Two contributions lead to the very promising WGAN [58, 59] by making an important key observation. The GAN does not specifically incorporate a metric over the dimensions of the phase space leading to rather meaningless gradients for the generator network (see fig. 4.3). The following sections are going to introduce such a metric and translate it to a DL approach.

4.2.1 Wasserstein Distance

The Wasserstein distance is a measure between two N-dimensional probability distributions $P_{\boldsymbol{a}}$ and $P_{\boldsymbol{b}}$ and assesses their similarity. Another name for this distance is earth mover's distance better describing the way it behaves by imagining $P_{\boldsymbol{a}}$ as a distribution of earth heaps and $P_{\boldsymbol{b}}$ as targets or holes that need to be filled. There always exists an optimal transport plan according to which the work that needs to be done to move the earth to the holes is minimized (e.g. see fig. 4.1). In mathematical notation, such a transport plan γ is an element of all possible transport plans

between $P_{\boldsymbol{a}}$, $P_{\boldsymbol{b}}$ which can be written as

$$\gamma \in \Pi(P_{\boldsymbol{a}}, P_{\boldsymbol{b}}) \tag{4.1}$$

and is basically encoded as an $N+N$-dimensional probability distribution specifying the probability of taking the path from $\boldsymbol{a}$ to $\boldsymbol{b}$. This results in the notation

$$\mathbb{E}_{(\boldsymbol{a},\boldsymbol{b})\sim\gamma} \, ||\boldsymbol{a} - \boldsymbol{b}||_2 \tag{4.2}$$

describing the work that needs to be done for a given transport plan γ. In analogy to the earth heap interpretation, the expectation value is the earth mass moved and the vector norm is the distance that needs to be covered. The formal definition of the Wasserstein distance is then

$$D_W = \overbrace{\min_{\gamma\in\Pi(P_{\boldsymbol{a}},P_{\boldsymbol{b}})}}^{\text{optimal transport}} \overbrace{\mathbb{E}_{(\boldsymbol{a},\boldsymbol{b})\sim\gamma}}^{\text{mass}} \overbrace{||\boldsymbol{a} - \boldsymbol{b}||_2}^{\text{Euclidean distance}} \quad . \tag{4.3}$$

The Wasserstein distance is by this definition a proper distance measure between two probability distributions in the mathematical sense as it fulfills the four metric conditions, them being:

- Non-negativity: $D_W(P_{\boldsymbol{a}}, P_{\boldsymbol{b}}) \geq 0$
- Identity of indiscernibles: $D_W(P_{\boldsymbol{a}}, P_{\boldsymbol{b}}) = 0 \Leftrightarrow P_{\boldsymbol{a}} = P_{\boldsymbol{b}}$
- Symmetry: $D_W(P_{\boldsymbol{a}}, P_{\boldsymbol{b}}) = D_W(P_{\boldsymbol{b}}, P_{\boldsymbol{a}})$
- Triangle inequality: $D_W(P_{\boldsymbol{a}}, P_{\boldsymbol{c}}) \leq D_W(P_{\boldsymbol{a}}, P_{\boldsymbol{b}}) + D_W(P_{\boldsymbol{b}}, P_{\boldsymbol{c}})$

This encourages to think of the D_W as a distance. However, there might be some points about it that seem counter-intuitive. One example is the scaling of this distance. It is clear that the distance gets bigger with more local differences between two distributions. But having two distributions with the same D_W to some reference distribution does not mean that humans looking at some visual representation would necessarily classify them as being equally far away from the reference. A reason could be that one variable is much larger than the others and therefore dominating the numerical value of D_W while this dominating effect might not be visible in a visual representation where all variable axes were scaled. This leads again to the important point of normalizing data before processing it with DNNs.

The presented definition of D_W is very intractable because of the minimum computed over all transport plans. To actually make the WGAN work one needs the Kantorovich-Rubinstein duality.

$$D_W = \max_{C\in\mathrm{Lip}_1} -\mathbb{E}_{\boldsymbol{a}\sim P_{\boldsymbol{a}}} \, C(\boldsymbol{a}) + \mathbb{E}_{\boldsymbol{b}\sim P_{\boldsymbol{b}}} \, C(\boldsymbol{b}) \tag{4.4}$$

This formula makes use of the Lipschitz continuity

$$\mathrm{Lip}_k = \{f \in \text{continuous differentiable functions} \mid \forall \boldsymbol{x} \in \mathbb{R}^N : ||f'(\boldsymbol{x})||_2 \leq k\}. \tag{4.5}$$

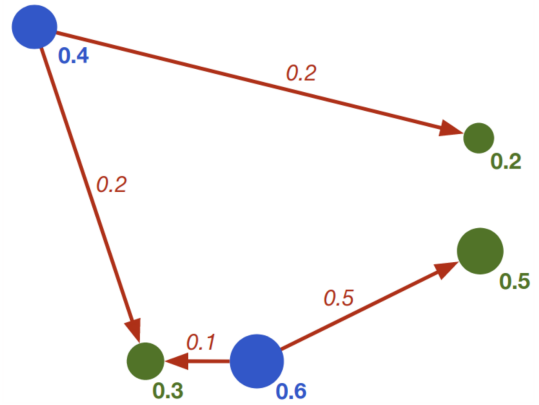

Figure 4.1: Optimal transport plan: The blue dots are the earth heaps and marked in green is the target distribution. The red arrows mark the optimal transport plan with numbers denoting the amount of mass moved along them. The work and therefore the Wasserstein distance is the sum over all arrow weights multiplied by their path length, respectively.

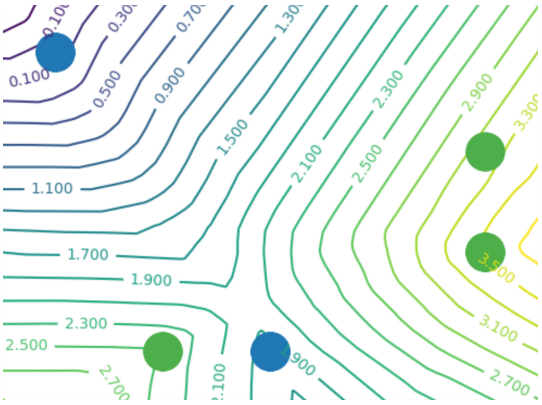

Figure 4.2: Demonstration of the (trained, close to optimal) critic function C over $\mathbb{R}^2$ having gradients bounded by 1 (Lip_1, visible through equidistant contour lines). The Wasserstein distance is $C(\text{green}) - C(\text{blue})$. Gradients of C should always be parallel to the optimal transport paths (red paths on the left) and perpendicular to the shown contour lines.

The duality states that the Wasserstein distance can be computed as a maximum over a set of functions with these functions being later on one of the networks called the critic (as opposed to the discriminator in the GAN setup). As an example of how this function C might look like it is visualized for the 2D case in fig. 4.2. It is clearly visible how this function provides the gradients to push the blue distribution exactly along the optimal transport path if used in a gradient descent approach.

4.2.2 Critic replaces Discriminator

The only thing that needs changing compared to the traditional GAN (sec. 3.2.7) to arrive at the WGAN is the loss function of the discriminator network effectively turning the discriminator into a critic network and the output of this network is the function C. The used loss is

$$\mathcal{L}_W = \mathbb{E}_{\boldsymbol{a}\sim P_a}\, C(\boldsymbol{a}) - \mathbb{E}_{\boldsymbol{b}\sim P_b}\, C(\boldsymbol{b}). \tag{4.6}$$

Because training minimizes this loss one obtains a critic network approximating the optimal critic function and the loss is converging towards the value of the negative

Wasserstein distance.

$$\mathcal{L}_W \rightarrow \min_{C \in \mathrm{Lip}_1} \mathbb{E}_{\boldsymbol{a} \sim P_{\boldsymbol{a}}} C(\boldsymbol{a}) - \mathbb{E}_{\boldsymbol{b} \sim P_{\boldsymbol{b}}} C(\boldsymbol{b}) \\ = - \max_{C \in \mathrm{Lip}_1} -\mathbb{E}_{\boldsymbol{a} \sim P_{\boldsymbol{a}}} C(\boldsymbol{a}) + \mathbb{E}_{\boldsymbol{b} \sim P_{\boldsymbol{b}}} C(\boldsymbol{b}) = -D_W \quad (4.7)$$

However, just training with the aforementioned loss does not guarantee that the resulting function will be 1-Lipschitz continuous. As opposed to the first WGAN proposal the Improved WGAN uses a gradient penalty term in the loss enforcing this condition instead of clipping network weights. This has multiple positive effects. First of all the performance has proven to be better in real-world applications using the gradient penalty. Another disadvantage of the weight clipping method is the non-trivial interpretation of the Wasserstein loss as it now converges not towards the actual Wasserstein distance but a scaled value[5] with this scale depending heavily on the network structure (number of layers, activation function, etc.).

The gradient penalty loss term is given originally by

$$\mathcal{L}_{GP} = \mathbb{E}_{\boldsymbol{x} \sim \overline{P_{\boldsymbol{a}} P_{\boldsymbol{b}}}} \left(||C'(\boldsymbol{x})||_2 - 1 \right)^2 \quad (4.8)$$

where an unconventional notation $\boldsymbol{x} \sim \overline{P_{\boldsymbol{a}} P_{\boldsymbol{b}}}$ was used to denote a uniform randomly drawn $\boldsymbol{x}$ from a random connecting line between an element from $P_{\boldsymbol{a}}$ and $P_{\boldsymbol{b}}$ or more formal $\boldsymbol{x} = \epsilon \boldsymbol{a} + (1 - \epsilon)\boldsymbol{b}$ with $\epsilon \sim U(0, 1)$, $\boldsymbol{a} \sim P_{\boldsymbol{a}}$, $\boldsymbol{b} \sim P_{\boldsymbol{b}}$. This loss is implemented by an element-wise linear combination of the two provided samples $\boldsymbol{a}$ and $\boldsymbol{b}$ and a randomly drawn ϵ for every sample.

The Lipschitz condition itself is not directly forcing the gradient to be exactly 1. The reason why this loss actually provides a stronger constraint than necessary is that in theory, the converged critic function has everywhere a gradient of exactly 1 except for a null set. Another possibility of a gradient penalty matching closer the Lipschitz condition would be

$$\mathcal{L}_{GP} = \mathbb{E}_{\boldsymbol{x} \in \overline{P_{\boldsymbol{a}} P_{\boldsymbol{b}}}} \left[\max(0, ||C'(\boldsymbol{x})||_2 - 1) \right]^2 \quad (4.9)$$

only penalizing gradients that are actually larger than 1. During experimentation, it was found that the second version of the gradient penalty behaved similarly to the first version at the beginning of the training. In the final stages, however, the second version had a better convergence behavior (less noise, same minimal loss value) compared to the loss term proposed by the authors of [59]. A possible explanation of this behavior would be that the generator produces already samples close to the real data and the critic is almost constant everywhere while also trying to have a gradient of 1 in the case of eq. 4.8. A function satisfying both conditions does not exist and therefore the critic is close to a constant function with a large amount of noise to provide the gradients of 1. In the case of eq. 4.9 the critic network can smoothly approach the constant function resulting in a more stable training and

[5] This is due to the fact that the weight clipping ensures Lipschitz continuity by bounding the gradient to some arbitrary value k instead of 1. This k is exactly the amount by which the value of the loss is scaled compared to D_W.

loss curve. For the following evaluations the gradient penalty from eq. 4.9 was used and the critic network was trained with the complete loss of $\mathcal{L}_C = \mathcal{L}_W + \kappa_{GP}\mathcal{L}_{GP}$ with κ_{GP} being a tunable hyperparameter.

Figure 4.3 visualizes how the concept of a critic mitigates the issues of a traditional GAN. Clearly, the training instability is resolved. The critic can be in a converged state and still give a meaningful gradient over the data phase space. In this regard, a fully trained discriminator fails. In fact, the training routine for the presented WGAN setup leverages this by training the critic much more often than the generator network to ensure a high quality of critic gradients. The mode collapsing is a bit harder to explain and there is some theoretical work published on this (e.g. [60]). The essential difference is that the critic is more globally informative. In fig. 4.3 the critic splits the fake blue distribution close to 0 by providing different gradients for the points on the left of this threshold compared to the points on the right. This is exactly the point at which the left blue distribution matches the size of the left green distribution. The right blue side can exactly fill up the right green peak. This kind of behavior makes it impossible to move all data points on to one single point. In contrast, the discriminator only evaluates locally the ratio between fake and real data without taking into account if there is enough generated data in the region or not. This can and in practice often does result in mode collapsing and this collapsed mode given by the generator is then playing a pointless cat-and-mouse game with the discriminator afterward.

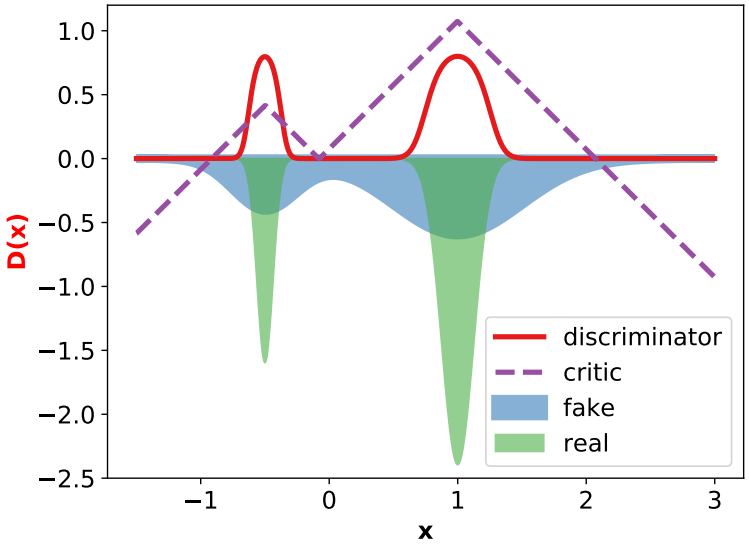

Figure 4.3: A toy example of generated (fake) data and training (real) data. The discriminator and critic are computed numerically by evaluating exactly the minimum of the corresponding losses (possible in this simple 1D case). The graph shows clearly the vanishing gradient of the discriminator for most of the fake samples. The critic provides everywhere a good description of how to improve the fake data distribution.

4.3 Training and Data Setup

Before looking at the results of this model, this section will give some information on the technical aspects of this setup like the specific network structure and training procedure that was used, the preprocessing of the data and the concrete datasets that were used.

4.3.1 Wasserstein-GAN Architecture and Training

As the used data has no structure that could be easily exploited with existing network layouts multiple dense layers were used. A hyperparameter search revealed that the size is not extremely critical for the performance. Both networks (generator and critic) became the same structure as the overall complexity is always limited by one of the networks (bottleneck). Each network consisted of 6 hidden layers à 288 nodes. Thus, for the updates to the first layer of the generator the gradient had to flow through 12 layers in the backpropagation step.

The latent space for the generator consisted of 128 standard normal distributed values. The dimensionality of the latent space had no significant impact on the performance.

A few common activation functions (see sec. 3.1.3) were compared with the ReLU which in the end performed significantly better than the others.

The critic was trained with the complete loss function presented in the previous section and a scaling of the gradient penalty loss of $\kappa_{GP} = 10$ (also robust with respect to the performance). The generator is trained on the critic output with the opposite sign compared to the critic loss:

$$\mathcal{L}_W = \mathbb{E}_{\boldsymbol{t}\sim P_{\text{train}}} C(\boldsymbol{t}) - \mathbb{E}_{\boldsymbol{g}\sim P_{\text{gen}}} C(\boldsymbol{g}) \tag{4.10}$$

$$\Rightarrow \quad \mathcal{L}_G = + \mathbb{E}_{\boldsymbol{z}\sim P_{\boldsymbol{z}}} C(G(\boldsymbol{z})) \tag{4.11}$$

As an optimizer, the Adam algorithm performed slightly better than the others with the default learning rate of 0.001 and a batch size of 256.

The critic was trained until convergence[6] before each training step of the generator.

All these hyperparameters where optimized on a validation set. As the number of hyperparameters is small and most of them did not significantly influence the performance with respect to a reasonable range of values the performance values are not reported on a separate test set. Instead, a resampling of the training and validation data was performed and the performance reported on this new validation set as the bias is to be expected much smaller than the fluctuations between individual trainings.

[6] The convergence criterion was met when 32 consecutive update steps were performed which did not improve the loss $\mathcal{L}_C$ on validation data.

4.3.2 Preprocessing and Shape Normalization

First experiments have shown that GANs, in general, are not very good in reproducing shapes of histograms, especially if these shapes have distinct features (e.g. the peaks for low and high CSV values). This gave the reason for taking another look at the normalization of data (see sec. 3.2.1). Instead of normalizing the position and the spread of the data one could directly perform some kind of shape normalization to match it exactly to a normal distribution. This was actually done for the upcoming task yielding much better results. The principle of how it works is depicted in fig. 4.4. The transform function $T_X(x)$ for a specific variable X is given by

$$T_X(x) = \Phi^{-1}(F_X(x)) \tag{4.12}$$

where $\Phi(x)$ is the cumulative distribution function (CDF) for the normal distribution and $F_X(x)$ is the CDF for the distribution of the variable X. The value $p = F_X(x)$ is mapping a value x to the corresponding percentile p which then, in turn, can be converted to a standard normal distributed value $T_X(X) = \Phi^{-1}(p)$ using the inverse normal CDF. Analogously the inverse of this transform is

$$T_X^{-1}(x) = F_X^{-1}(\Phi(x)). \tag{4.13}$$

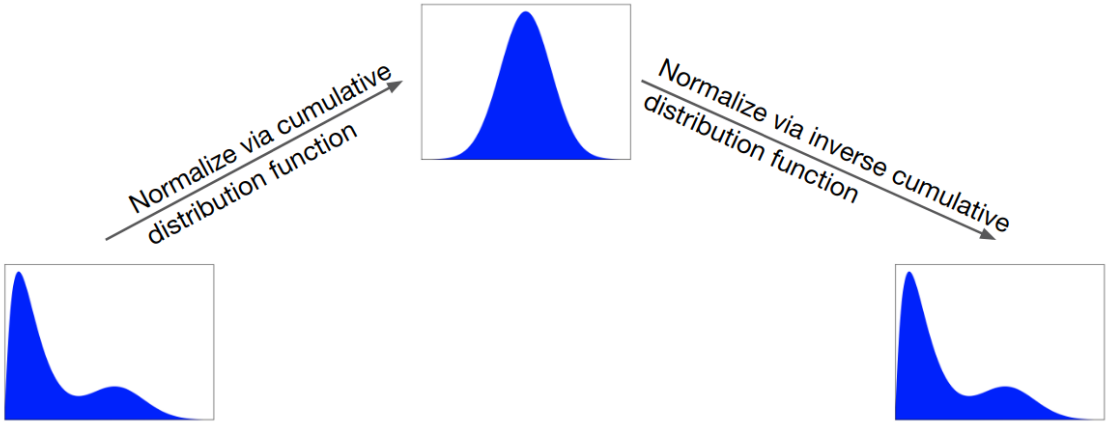

Figure 4.4: Process of shape normalization: *Left:* The transformation T_X uses the cumulative distribution function of X to transform the complete value distribution into a standard normal distribution. *Right:* The transformation T_X^{-1} transforms a standard normal distribution into the data distribution of X using the inverse cumulative distribution function of X.

An important remark about this method is that the function $F_X(x)$ is just given by data points and therefore represents a step function (see fig. 4.5). This has two issues. Using a step function in the process of transforming the data from a normal distribution back to the original one (T_X^{-1}) results in a distribution consisting only of previously existing values in X. The second issue is also regarding the back

transformation and data limits. It would be impossible to get larger values than the maximum or smaller values than the minimum observed in the data of X. Both issues are problematic with respect to the WGAN's goal of mimicking the given data distribution but not exactly copying it. However, two small adaptations to F_X can be made (shown in fig. 4.5). First, one simply interpolates between the steps instead of actually using the step shape. Secondly, at the minimum value and maximum value, the CDF is extended with a simple exponential function (three free parameters) fixed by the value (x, y removes 2 degrees of freedom) and the gradient of the first or last line of the interpolation (removes the last DOF).

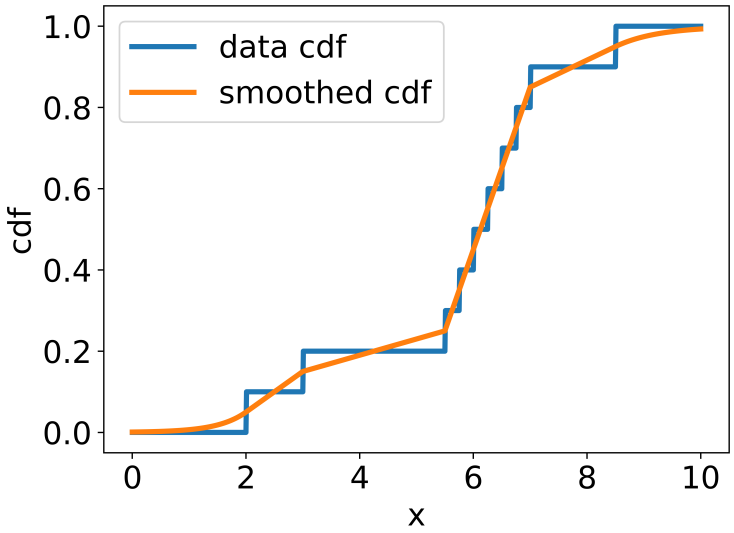

Figure 4.5: Modified cumulative distribution function of X. The linear interpolation and the exponential extensions on the left and the right ensure that the inverse cumulative distribution function is well-defined and can map onto any real number.

4.3.3 Datasets

To not only quantitatively (values of the metrics) but also qualitatively (comparison of different values of the same metric) assess and compare the generated data it is helpful for having some dataset that behaves worse than the WGAN-generated data and one that behaves better. The better dataset will be the training data itself. This is not really a solution to the formulated goals as copying the data is not viable, however, for assessing other aspects and qualities of data distributions the train data can just be split into two equal-sized sets to provide two different datasets with the same general phase space density. One of these two takes the role of being the training data while the other set takes the role of the idealized generated data. A very simple and therefore worse generation of data using the train data would be simply sampling from the variables independently without taking care of correlations between different observables. This is practically done by sampling from a standard

normal distribution and using the inverse transform from the previous section to get a continuously distributed variable value. This dataset then will match perfectly one-dimensional histograms, but every higher correlation will be completely destroyed which is why it should have in general a lower quality than WGAN-generated data. In the following, it will be called the decorrelated data.

The training datasets themselves were generated using *Pythia & Delphes* (see sec. 2.1.4). For the following results, 4 datasets were used in total. 2 of them had 32 low-level physics observables which were just the momentum four vectors of eight jets[7]. These jets were ordered according to their origin on the tree level generator output. The other two datasets consist of 26 high-level variables (see app. A for a list of these variables). From each of the high-level and low-level category there exist two datasets one being a signal dataset containing only events generated by the process of $gg \rightarrow t\bar{t}H \rightarrow t\bar{t}b\bar{b}$ and the other being the corresponding dominant background process dataset containing events generated by the process of $gg \rightarrow t\bar{t}b\bar{b}$. In both cases, it was required that the top quarks decayed semileptonic (one hadronic $t \rightarrow bW \rightarrow bqq'$, one leptonic $t \rightarrow bW \rightarrow bl\nu$). In the following section, it will be clearly stated at which point which of these four datasets are used using the terminology of high-level/low-level and $t\bar{t}H/t\bar{t}b\bar{b}$ (signal/background).

4.4 Metrics and Results

The goal of this section is to evaluate the WGAN-generated data compared to the two other benchmark datasets defined in the previous section. As this is one of the first investigations of a WGAN applied to non-image data the quality cannot be assessed in any similar way as has been done for image data used in almost any application of a WGAN. Before going into more details one can look at the 1-dimensional histograms over some of the variables for the three datasets (presented in fig. 4.6, 4.7). As can be seen even with the shape normalization the WGAN struggles a bit in reproducing the exact histogram distribution.

4.4.1 Wasserstein Distance as Metric

Section 4.2.1 already explained the behavior of the Wasserstein distance as a metric between two data distributions. The training of the WGAN already provides a critic network which can be used to estimate this measure. However, for the purpose of this evaluation, the computation of this measure should also be done on the decorrelated dataset as well as on two halves of the training data. An additional critic network was set up with the corresponding loss and same inner architecture as the one used in the WGAN. During the training, the value $-\mathcal{L}_W$ is a good approximation for the earth mover's distance which is shown as a function over the training time for the three datasets in fig. 4.8.

The value it converges towards should be a value close to the actual Wasserstein

[7] Bhad, Bj1, Bj2, Blep, Lj1, Lj2, Lep1, Nu1

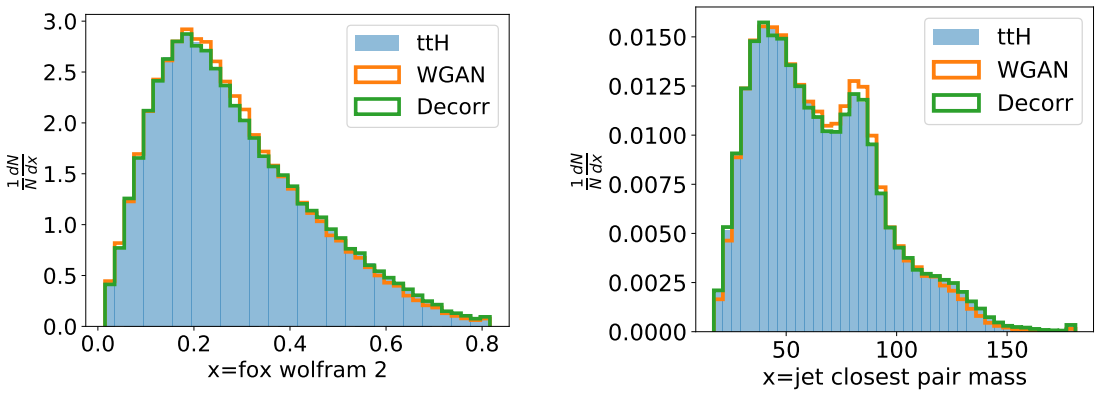

Figure 4.6: Variable histograms: The blue curve denotes the MC for the high-level $t\bar{t}H$ dataset. The yellow curve denotes the WGAN-generated data. The green curve is showing the histogram for the decorrelated dataset. As by construction, the decorrelated dataset fits nicely the distribution of the training data in blue. Left- and rightmost bins contain also overflow samples.

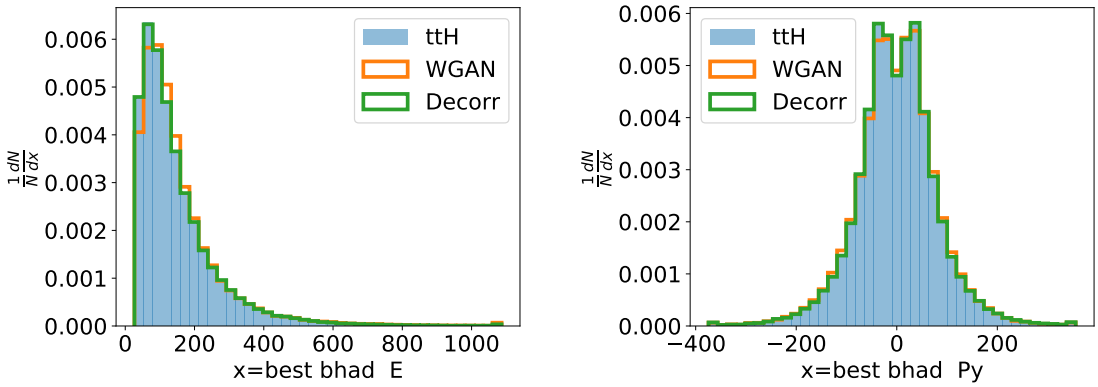

Figure 4.7: These plots show the same as fig. 4.6 but for the low-level $t\bar{t}H$ dataset instead of the high-level one.

distance between the ground truth $t\bar{t}H$ dataset and one of the three datasets to compare to. This metric already shows nicely the expected bad performance of the decorrelated dataset as it converges to a relatively high distance of 3, whereas the WGAN-generated data has a distance of approximately 0.3 to the $t\bar{t}H$ data. As expected when the MC data is split into two sets and compared using a critic an estimated distance of 0 is achieved meaning the critic can not successfully differentiate between one or the other. This is also a very strong check that there is no overfitting happening during training. This shows that the WGAN-generated dataset indeed looks similar to the training data and that the proposed way of computing this measure by training a simple critic network is suitable for this task.

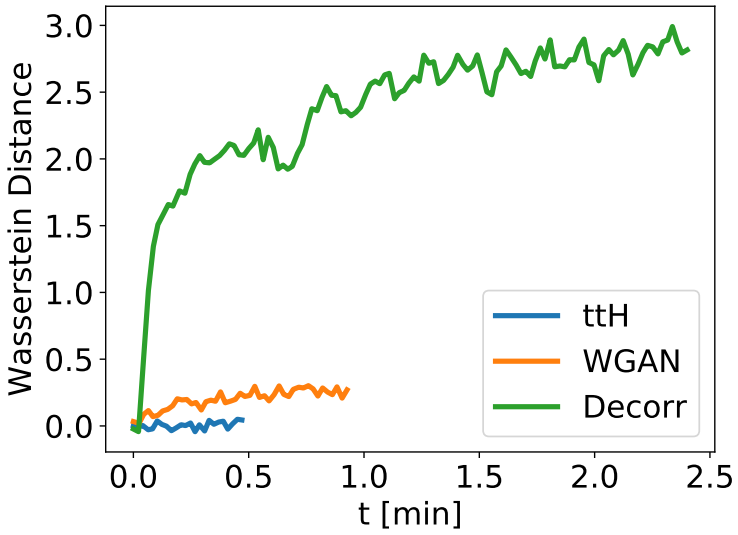

Figure 4.8: The plot displays the estimated Wasserstein distance $-\mathcal{L}_W$ during training of the critic between $t\bar{t}H$ data with high-level variables as ground truth and the labeled data as the data to compare to (looks similar for low-level). An automatic stopping criterion results in different runtimes. It is important that this runtime is low to ensure an efficient training of the whole WGAN where a critic train step is performed much more often than a generator train step.

4.4.2 Fisher Transformation

As the computation of the Wasserstein distance relies on DL it alone can not be a good metric. However, comparing and inspecting high-dimensional data is, in general, a very hard task and DL techniques next to other ML techniques are currently the only ones that can truly access high-dimensional correlations in a general way[8]. For providing at least one mathematically rigorous measure one can reduce the problem of comparing only 2-dimensional correlations. Picking two variables one can take a look at a 2-dimensional histogram, but this would be impractical to do so for all combinations of variables. A common measure to reduce this correlation information is to compute the correlation coefficient between two variables. It is then possible to plot all the correlation coefficients for all combinations of variables into a matrix which is shown for the different datasets in fig. 4.9.

As expected these plots show that the decorrelated data is indeed decorrelated and again the WGAN-generated data has a close resemblance to the training data. However, the logarithmic scale also reveals that especially in the low-level dataset the WGAN produces correlations where there should not be any.

To construct a meaningful and condensed value characterizing the difference in these correlation values one can use the Fisher transformation [61].

[8] Accessing high-dimensional correlations in a **non**-general way is easy and usually done by fitting a specific model parameterized by the variables spanning the high-dimensional space.

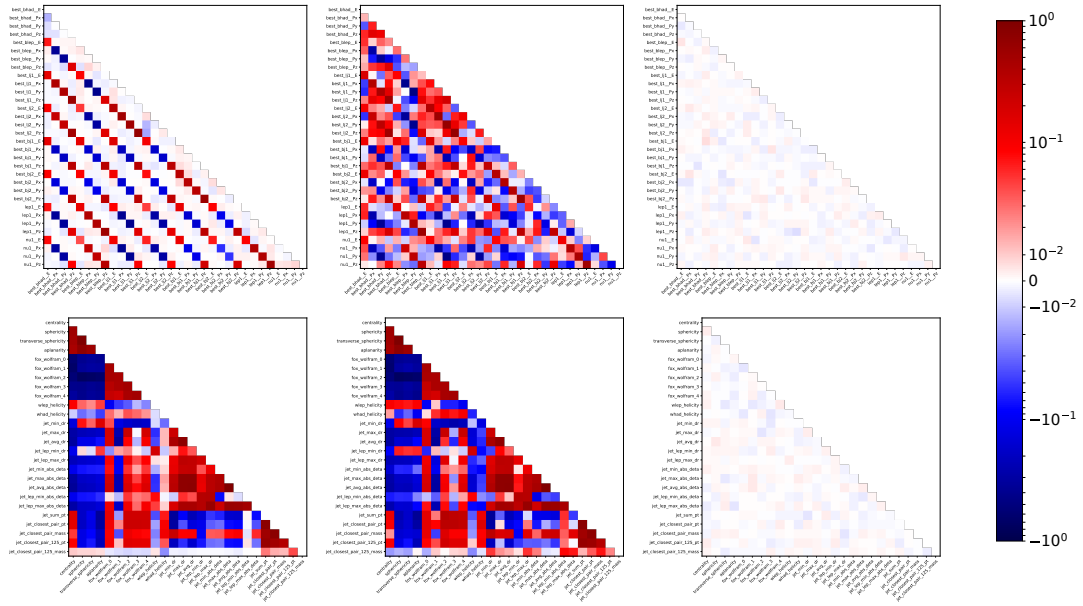

Figure 4.9: The different correlation coefficients are represented for the $t\bar{t}H$ dataset. The top row consists of evaluations on the low-level variables whereas the bottom row presents the high-level observables. From left to right: MC $t\bar{t}H$, WGAN-generated and decorrelated data.

The Fisher transformation is simply the arctanh of a correlation coefficient, but it has a very helpful feature. In contrast to the raw correlation coefficient r the Fisher transformation of this value $z = \text{arctanh}(r)$ is Gaussian distributed. Lets say we have computed r for the variables X and Y over a sample of size N. Now the variance of the variable z is approximately $\frac{1}{\sqrt{N-3}}$ and vanishes for large N as expected. For $N \to \infty$ the value r approaches the true correlation ρ, so the mean of z has to be $\text{arctanh}(\rho)$ and is in fact independent of N. If there exist two samples A and B of size N_A and N_B following supposedly the same probability distribution the value

$$\hat{z}_{AB} = \frac{z_A - z_B}{\sqrt{\frac{1}{N_A-3} + \frac{1}{N_B-3}}} \sim \mathcal{N}(\mu = 0, \sigma = 1) \tag{4.14}$$

should in conclusion be normal distributed with a mean of 0 and a variance of 1. With two datasets for comparison each containing 26 variables (or more) one has $\frac{25*26}{2}$ different $\hat{z}_{AB,XY}$ values which should all be normally distributed. With this much statistic, it makes sense to reduce this to one number by using the χ^2 measure which is able to show if the values reasonably follow a normal distribution. This results in a single value characterizing the difference in 2-dimensional correlations between two datasets A and B:

$$\chi^2(FT(A,B)) = \frac{\sum_{X,Y} \hat{z}^2_{AB,XY}}{\sum_{X,Y}} \tag{4.15}$$

If the two distributions A and B differ then the difference $z_A - z_B$ should not be distributed around zero but significantly deviate from that. This, in turn, would

$\chi^2(FT)$	MC	WGAN	decorrelated
26 high-level variables	≈ 1	≈ 50	$\approx 10,000$
32 low-level variables	≈ 1	≈ 40	$\approx 2,000 - 3,000$

Table 4.1: The correlation scores of the different datasets with respect to the training data (MC) for the $t\bar{t}H$ region.

blow up the χ^2 measure indicating clearly that the two distributions do not match well with respect to 2-dimensional correlations.

Figure 4.10 shows how this constructed value improves over the course of training. As expected the comparison of MC data to itself results in a value of the order 1. The other values are reported in tab. 4.1 for the $t\bar{t}H$ dataset. These values do not differ for $t\bar{t}b\bar{b}$ with respect to the overall noise with this noise also being the reason for the rough estimates in the table.

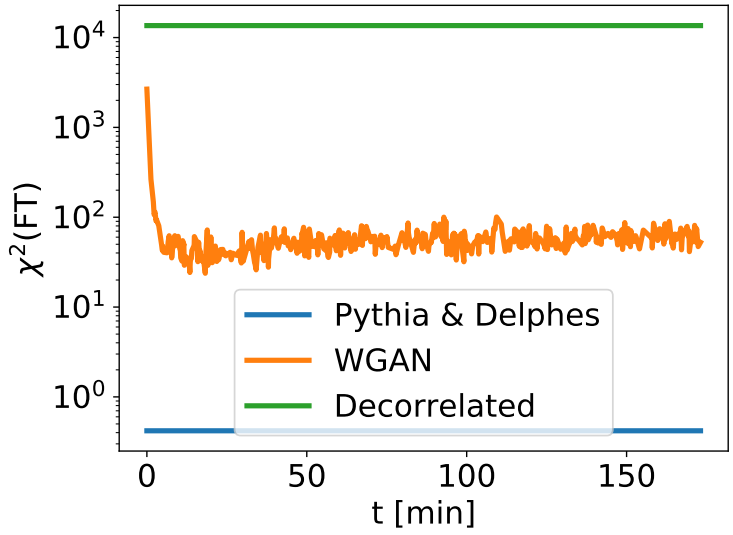

Figure 4.10: The improvement of the fisher transform score is clearly visible. As a reference, the values for the decorrelated data and the MC data have been added as horizontal lines. The used dataset was again the $t\bar{t}H$ high-level data with the others looking similar.

4.4.3 Classification Benchmark

The first two measures that were introduced are lightweight in terms of computation and can be tracked during training for having a fast feedback loop. However, it is hard to actually say if the generated WGAN data is good enough to be used for certain tasks. In fact, this definition of 'goodness' depends heavily on the task at hand. One task might only take the 1-dimensional histograms into account where

accuracy	MC	WGAN	decorrelated
26 high-level variables	64.0 %	61.1 %	62.3 %
32 low-level variables	86.0 %	69.9 %	58.6 %

Table 4.2: The reported validation accuracies of a classifier trained on the different datasets but validated always on the MC data.

a different application mostly considers highly correlated variables. For this kind of data generation, the ultimate and deciding metric will always be a value considering the actual task at hand. For this reason, it is interesting to mimic a simplified physics analysis. Most analyses for SM searches come down to discriminating signal from background events using nowadays mostly DNNs. In this section, such an analysis will be presented in an idealized way using the signal and background dataset and training a simple discriminator network[9] trained to differentiate between the two processes. This training happens three times, once for each dataset (MC, WGAN-generated, decorrelated). After the training the accuracy of the network will be reported on some validation data, this data being MC itself which was, of course, left out during the training in the MC dataset case.

An important note is that the decorrelated and WGAN-generated data were constructed separately for the $t\bar{t}H$ and $t\bar{t}b\bar{b}$ dataset. This means there were two WGANs trained and used for generation, one producing $t\bar{t}H$-like events and the other producing $t\bar{t}b\bar{b}$-like events.

The results of these trainings are shown in tab. 4.2. As expected, training a classifier on the data domain on which it is also evaluated results in the best accuracy. The reason why the accuracy on the low-level dataset is so much higher is that the used jets were ordered according to the event generator labels. The high-level variables are computed by algorithms that do not take such orderings into account as these do not exist in measured data, therefore the discriminating power is lower for the high-level dataset. It is interesting to see that in this case, the decorrelated dataset performs better than the WGAN data. The most likely explanation is that the high-level variables are already constructed in such a way that they are discriminating all by themselves while the correlations between the observables seem to contain very little information. The WGAN being able to adapt the distribution actually introduces slight differences to the one-dimensional histograms (as seen in the previous section) which in the case of the high-level variables seems to hurt more than the gain in potentially higher order variables correlations could be for the classification task. For the low-level dataset, these correlations make a difference and the one-dimensional histograms contain close to no information. However, the difference between the MC and the WGAN-generated data is large so that for this particular task one would not consider the WGAN to be a viable solution.

As we have seen, the WGAN is capable of generating data according to some training data distribution. The Wasserstein distance and the 2-dimensional correlation

[9] This discriminator has for simplicity again the same internal structure as the critic and the generator in the WGAN.

coefficients were evaluated to get better insights into the performance of the WGAN. Especially the latter one being a completely deterministic tool results in the confident confirmation that the WGAN is generating correlated data. The classification benchmark introduced a challenge for the WGAN. Even though the classifier was simple it showed that the data quality should always be measured with respect to the task the data is needed for. The classifier revealed that even though the WGAN is capturing correlations, it either is missing more complicated correlations or is just not very accurate, leading to a drop in the classification accuracy.

At the same time, the benchmark leads to some insights about the data itself. As the WGAN data can be regarded as a noisy version of the MC data distribution, we have seen that the classification on the high-level variables is more based on their individual discriminating features and not their correlated interplay, whereas it is the opposite with the 32 low-level variables. In both cases, it still seems to be very important that the lower dimensional correlations (one-dimensional histograms) match well to obtain a good performance.

This leads to the next section discussing the refinement of the MC-generated data itself by improving the shape-agreement of measured data and produced MC events.

5 Deep Scale Factors

The previous section has shown how difficult it can be to produce high-quality generated events and that much information is extracted out of the correlations between variables. Even the computationally expensive simulations using *GEANT4* are not in perfect agreement with the measured data. Because of this, there are usually different scale factors (SFs) and weights applied to the MC simulations mostly for correcting specific yields, e.g. the b-tagging efficiency. In more recent years analyses do not only use the binary b-tag given by the CSV algorithm and specific working points but started using the continuous CSV output directly. For this purpose, a second set of b-tagging SFs was developed specifically correcting the shape of the CSV output distribution. These SFs are computed by looking at a well-known and physically well-modeled phase space region in the data and matching the histogram shape of the corresponding MC events through the SFs. This fit was traditionally done only over a few variables with a coarse binning to avoid low statistics in the computation of the SFs. As argued, DL can be used to overcome this issue. The following section aims at presenting an alternative approach for computing shape-correcting SFs by using a specifically designed DL setup.

5.1 Generalization

The task can be abstracted to fitting a set of generated events to a measured dataset. In mathematical notation it can be written as follows:

Using the notion of phase space densities

$$\rho_A(\boldsymbol{x}) = \sum_{(w_i,\boldsymbol{x}_i)\in A} w_i \cdot \delta(\boldsymbol{x} - \boldsymbol{x}_i) \tag{5.1}$$

the goal is to produce a refined version of the generated (fake) data $\hat{\rho}_{\text{fake}} \approx \rho_{\text{fake}}$ that has every a similar density as the measured (real) data. This can be written as

$$\int_{\Delta V} \rho_{\text{real}}(\boldsymbol{x}) d^N\boldsymbol{x} = \int_{\Delta V} \hat{\rho}_{\text{fake}}(\boldsymbol{x}) d^N\boldsymbol{x} \quad \text{for all } \Delta V \tag{5.2}$$

where ΔV is some minimal resolution. This can be called a refinement step and there are two possible ways to approach this.

The first way to approach this would be to slightly change the event variables $\hat{\boldsymbol{x}}_i(\boldsymbol{x}_i) \approx \boldsymbol{x}_i$ of the generated data on an event-by-event basis leading to

$$\hat{\rho}_{\text{fake}}(\boldsymbol{x}) = \sum_{(w_i,\boldsymbol{x}_i)\in\text{fake}} w_i \cdot \delta(\boldsymbol{x} - \hat{\boldsymbol{x}}_i(\boldsymbol{x}_i)). \tag{5.3}$$

Without modern ML techniques this is very difficult to achieve, however, a slightly modified GAN approach to the one presented in the previous section could actually achieve such a task. Using a WGAN, this way of refining simulations was presented

in [62] refining data time-traces in the setting of the Pierre Auger experiment.

A second way of reaching the desired effect would be to slightly change the event weights $(\hat{w}_i, \boldsymbol{x}_i) = (SF(\boldsymbol{x}_i) \cdot w_i, \boldsymbol{x}_i)$ of the MC simulations to better fit the phase space distribution of the measurements. The introduced SFs would then be close to 1 and result in

$$\hat{\rho}_{\text{fake}}(\boldsymbol{x}) = \sum_{(w_i, \boldsymbol{x}_i) \in \text{fake}} SF(\boldsymbol{x}_i) \cdot w_i \cdot \delta(\boldsymbol{x} - \boldsymbol{x}_i). \tag{5.4}$$

This is exactly the way the traditional SFs work. However, to compute the SFs a discretization over a small number of bins was performed suffering from either a coarse binning (low number of bins) or low statistics inside the bins (high number of bins). A DL approach could overcome this issue.

Two main reasons lead to the decision of following the second approach. For the usage of such refined MC in analyses, it is important to have manageable uncertainties which can be propagated. The first method makes this potentially very hard. In contrast, SFs are already in use and the corresponding uncertainties have been investigated. They are mostly method independent. The other reason is that the second approach is in general much closer to the traditional SFs. Therefore, the here presented deep SFs can be easily cross-checked and compared against the traditional ones. Through the usage of DNNs for the SFs, it could even be possible to get physical insights into the current method which could lead to improvements in the old method without completely replacing it. The importance of the last argument is not to be underestimated. Looking at the SFs of the last years [63][10] one can see that they exhibit strong fluctuations that are hard to explain or motivate. Extending the idea of SFs to more variables and removing the binning completely can hopefully improve on this behavior.

5.2 DeepSF Architecture

The basic idea of the deep SFs is to let a DNN take jet variables as input and produce a corresponding SF. This SF needs to be multiplied with the regular MC weight to get the shape-corrected simulation. The training of this SF-Net is done via a loss function evaluating the difference between MC and data. This difference is characterized using a second DNN as an adversary similar to the GAN setup. This adversary is a discriminator trained with the softmax cross-entropy as loss function . As shown in sec. 3.1.2, the cross-entropy as loss function has the nice property of guiding the output to converge towards the ratio of the phase space densities between the two distributions at the given data point. In terms of these phase space distributions $\rho_{\text{real}}(\boldsymbol{x})$ and $\hat{\rho}_{\text{fake}}(\boldsymbol{x}) = SF(\boldsymbol{x}) \cdot \rho_{\text{fake}}(\boldsymbol{x})$, a trained discriminator fulfills approximately

$$D(\boldsymbol{x}) \approx \frac{\rho_{\text{real}}(\boldsymbol{x})}{\rho_{\text{real}}(\boldsymbol{x}) + \hat{\rho}_{\text{fake}}(\boldsymbol{x})}. \tag{5.5}$$

[10] CMS internal, therefore not shown in this thesis.

Ideally, the gradient for the SF should be (omitting the $\boldsymbol{x}$ dependence)

$$\frac{\partial \mathcal{L}}{\partial SF} \propto \operatorname{sgn}\left(\frac{\rho_{\text{fake}}}{\rho_{\text{real}}}\right)\left(SF \cdot \frac{\rho_{\text{fake}}}{\rho_{\text{real}}} - 1\right). \tag{5.6}$$

The sign function is not only rigorously correct but also important in the beginning of the training where the discriminator might estimate an overall negative phase space density. Without it, the training would diverge at that point. Writing the above in the equivalent form

$$\frac{\partial \mathcal{L}}{\partial SF} \propto \left|\frac{\rho_{\text{fake}}}{\rho_{\text{real}}}\right|\left(SF - \frac{\rho_{\text{real}}}{\rho_{\text{fake}}}\right) \tag{5.7}$$

leads to the loss function

$$\mathcal{L} = \left(SF - \frac{\rho_{\text{real}}}{\rho_{\text{fake}}}\right)^2 \cdot \left|\frac{\rho_{\text{fake}}}{\rho_{\text{real}}}\right|. \tag{5.8}$$

This derivation makes the assumption that no gradient regarding SF is propagated through $\frac{\rho_{\text{fake}}}{\rho_{\text{real}}}$ and its inverse. However, this loss cannot be implemented in this manner, as the ratio of the phase space densities can only be computed using the discriminator

$$\frac{\rho_{\text{fake}}}{\rho_{\text{real}}} = \frac{1-D}{D}\frac{1}{SF} \tag{5.9}$$

which clearly depends on the SF. This is solved by simply computing the above ratio and applying a gradient stop on it in the computation flow of the software.

With this loss function for the SF-Net, one has all the ingredients to implement the learning framework. It is schematically depicted in fig. 5.1.

At this point, one might ask themselves why there is no need for a Wasserstein metric in this setup, i.e., what is the exact difference between the two tasks. On first glance both seem to need an adversary comparing a *fake* and a *real* data distribution. As already pointed out in sec. 5.1, there are two ways to fit discrete data distributions: Either by changing the weights of single events or by changing the variables of the events effectively moving the events in the phase space. The discussion of the Wasserstein distance showed that the discriminator is not able to produce meaningful gradients for the moving of events as it mainly evaluates the local densities. The Wasserstein distance, on the other hand, did a good job on this, however, it fails to give a hint towards the local densities. With the SFs only being able to change the local phase space densities, the discriminator provides exactly the needed network output. The critic would not be able to provide meaningful information on how to adjust the event weight.

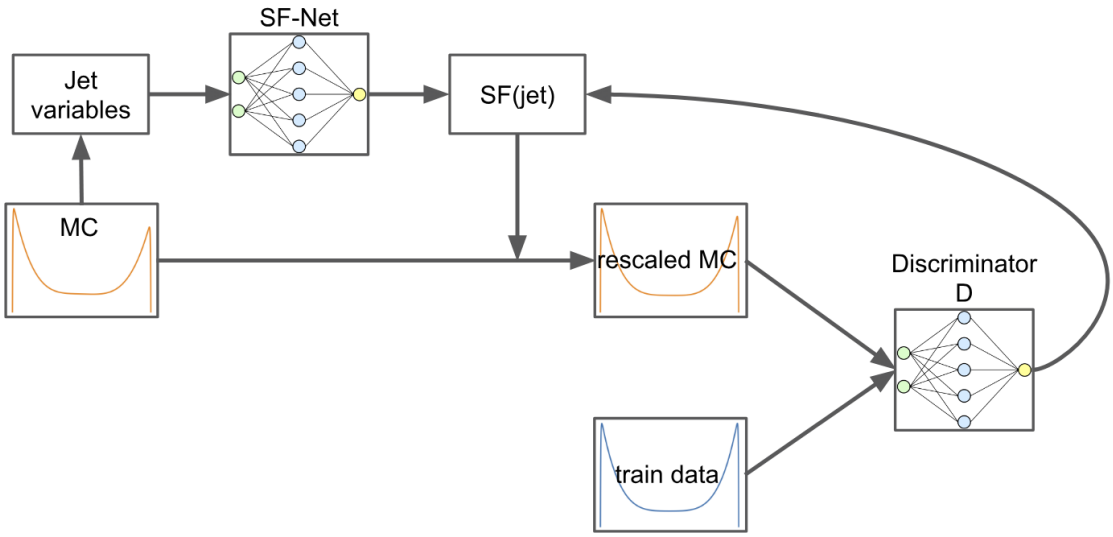

Figure 5.1: Diagram sketching roughly the learning setup for the deep SFs. The SF-Net takes jet variables of MC-generated events as input and produces the corresponding SFs. These SFs are applied to the MC events and result in the rescaled MC simulations. The discriminator is trained to differentiate the rescaled MC from the training data consisting effectively of the measured data and the subtracted MC contamination. The discriminator output D is then used in the loss function of the SF-Net to improve its output.

5.3 Traditional Scale Factors and Data Setup

This section will shortly introduce the traditional way of computing the b-tagging SFs as it was done in [63].

The traditional SFs essentially are computed in a heavy-flavor enriched region and a light-flavor enriched one. Additionally, they depend on three event parameters. For both flavor regions, a different binning (see app. B) is done in the other three variables p_T, η and CSV. In total there are 362 disjunctive bins. As there is no ground truth for the flavor accessible in data, the Tag and Probe method is applied. With corresponding selection cuts (see app. B) performed on the data one tries to isolate a region with a high purity of heavy flavor jets by focusing on $t\bar{t} \to l\nu l\nu b\bar{b}$ events. For the light flavor region, a cut is performed to achieve a pure Z+jets sample. Each of these datasets consists of exactly two jets (one of the cut criteria). By construction, these two have most of the time the same flavor type. The tag jet in the heavy flavor sample needs to pass the CSV medium working point. This results in a high confidence that the probe jet is also a heavy flavor jet even though the CSV algorithm might fail to indicate this. Analogously, the tag jet for light flavor has to fail the low CSV working point.

With this method, we have constructed two datasets of jets Data_{HF} and Data_{LF} with a high purity of the corresponding flavor, however, featuring a relatively wide

spectrum of CSV values. This exact same construction can be performed on MC to obtain $\mathrm{MC_{HF}}$ and $\mathrm{MC_{LF}}$.

As the b-tagging SFs should only be CSV-shape-changing, a yield normalization needs to be applied beforehand. The normalization was done on the presented datasets and separately for the heavy flavor and light flavor region and for every p_T and η bin. It introduces an additional trigger weight that is applied to the MC simulations.

There is still going to be some impurity in the constructed datasets regarding the flavor, which needs to be corrected if the SFs should depend on the true MC flavor. This is done by simply correcting the data yield using the MC dataset and splitting it into a pure and a contamination dataset. The pure MCs contain the generated flavor matching the region $\mathrm{MC_{HF,pure}}$, $\mathrm{MC_{LF,pure}}$ while the contamination dataset has the opposite flavor and is, therefore, estimating the contamination $\mathrm{MC_{HF,cont}}$, $\mathrm{MC_{LF,cont}}$. This enables us to compute the SFs:

$$\mathrm{HF}\ SF(\mathrm{CSV}, p_T, \eta) = \frac{\mathrm{Data_{HF}} - \mathrm{MC_{HF,cont}} \cdot \mathrm{LF}\ SF(\mathrm{CSV}, p_T, \eta)}{\mathrm{MC_{HF,pure}}} \tag{5.10}$$

$$\mathrm{LF}\ SF(\mathrm{CSV}, p_T, \eta) = \frac{\mathrm{Data_{LF}} - \mathrm{MC_{LF,cont}} \cdot \mathrm{HF}\ SF(\mathrm{CSV}, p_T, \eta)}{\mathrm{MC_{LF,pure}}} \tag{5.11}$$

As the expressions now depend on SFs from the other flavor region they are computed iteratively with their starting value being 1. Experience has shown that this tends to converge quickly.

$$\mathrm{HF}\ SF_i = \frac{\mathrm{Data_{HF}} - \mathrm{MC_{HF,cont}} \cdot \mathrm{LF}\ SF_{i-1}}{\mathrm{MC_{HF,pure}}} \tag{5.12}$$

$$\mathrm{LF}\ SF_i = \frac{\mathrm{Data_{LF}} - \mathrm{MC_{LF,cont}} \cdot \mathrm{HF}\ SF_{i-1}}{\mathrm{MC_{LF,pure}}} \tag{5.13}$$

Up to this point, everything will be done equivalently for the deep SFs. The DeepSF approach only substitutes the computation of this last equation, which is still done iteratively. The traditional SFs solve these equations by histogramming the data in the numerator (including the contamination) and the MC in the denominator. Taking bin-wise the ratio of these values results in a binned version of the SFs as it was desired. These are the traditional SFs. For the deep SFs, the nominator represents the *real* dataset and the denominator the *fake* dataset which needs to be refined by the SFs. The training will then deliver the SF-Net representing the functional dependence $SF_i(\mathrm{CSV}, p_T, \eta)$.

5.4 Training

As a general network architecture, a DenseNet layout was used for both the discriminator and the SF-Net. For both networks, the same complexity (number of layers

and nodes) was used to keep them balanced.

Without regularization, the discriminator network tends to overfit on the data as it needs to perform many more iterations than the SF-Net does. This leads to overall smaller SFs. In the overfitted case the discriminator predicts every simulated event as such by producing a low output. The SF-Net consequently reduces the SFs down 0.

The SF-Net does not need regularization as it trains on the whole dataset only a few times reducing the danger of overfitting significantly compared to the discriminator. Also, the labels are based on the discriminator output D which of course varies, therefore, leading to different labels even for the same events.

A good training setup with optimized hyperparameters was found using the python *skopt* module [50]. However, the search has shown that most of the hyperparameters did not affect the performance at all. The learning rate was the only parameter that had a significant impact. The following settings were used in the final network:

Network 10 layers with 128 nodes each and a leaky ReLU (α=0.01) as activation.

Regularization The L2 loss from sec. 3.2.4 was used with a scaling of $\lambda_{\mathrm{R}} = 3 \times 10^{-5}$.

Optimizer The Adam optimizer with the default learning rate of 0.001 produced the best results. Additionally, an exponentially decaying learning rate was scheduled with a decay parameter of $1 - 3 \times 10^{-5}$ per iteration.

Training loop Each loop fed a data batch with 64 jets into the network. Before every training iteration of the SF-Net, there were 8 iterations performed optimizing the discriminator.

To further improve stability there were in total 5 SF-Nets trained which only differ through the random initialization of the networks. The deep SFs analyzed in the upcoming section are the averages of these 5 SF-Net outputs.

5.5 Results

This section will present the resulting deep SFs through a few different representations. Where possible, a comparison is made with respect to the traditional SFs as described in sec. 5.3[11]. Additionally, the iteration of the SF computation is not relevant for the method itself. Thus, in order to reduce diverging effects of multiple iterations only the SFs from the first iteration are compared.

For the presented results the contamination was subtracted from the data. However, as this technical detail is not important for the method itself, the following discussion will use the word data for the reduced yield including the contamination.

First, a direct comparison will be presented where the deep SFs are produced with

[11] As the investigation was focused on the method and not on correctly producing the SFs some minor implementation details were left out if they did not contribute to the method working differently. The traditional SFs were, therefore, not taken as given, but computed for the specified data and leaving out the same minor implementation details so that only the methods in themselves are compared

exactly the same input variables as the ones that were used for the binning in the traditional method. Afterward, the deep SFs are extended to more input variables and the effects will be discussed.

5.5.1 Comparison to Traditional Scale Factors

This first comparison will show the results of the DeepSF approach applied to the input variables CSV, p_T, η. The used CSV algorithm is the DeepCSV which has four probability outputs for b, bb, c, and $udsg$. If not further specified the sum of b and bb is meant: $\mathrm{CSV} = \mathrm{CSV}_b + \mathrm{CSV}_{bb}$.

In fig. 5.2 and 5.3 histograms of different variables are presented. The gray shaded area is the distribution of the data which needs to be fitted by the scaled MC. The red histogram represents the MC-simulated distribution weighted with the traditional SFs. The green solid line represents the deep SF-weighted MCs.

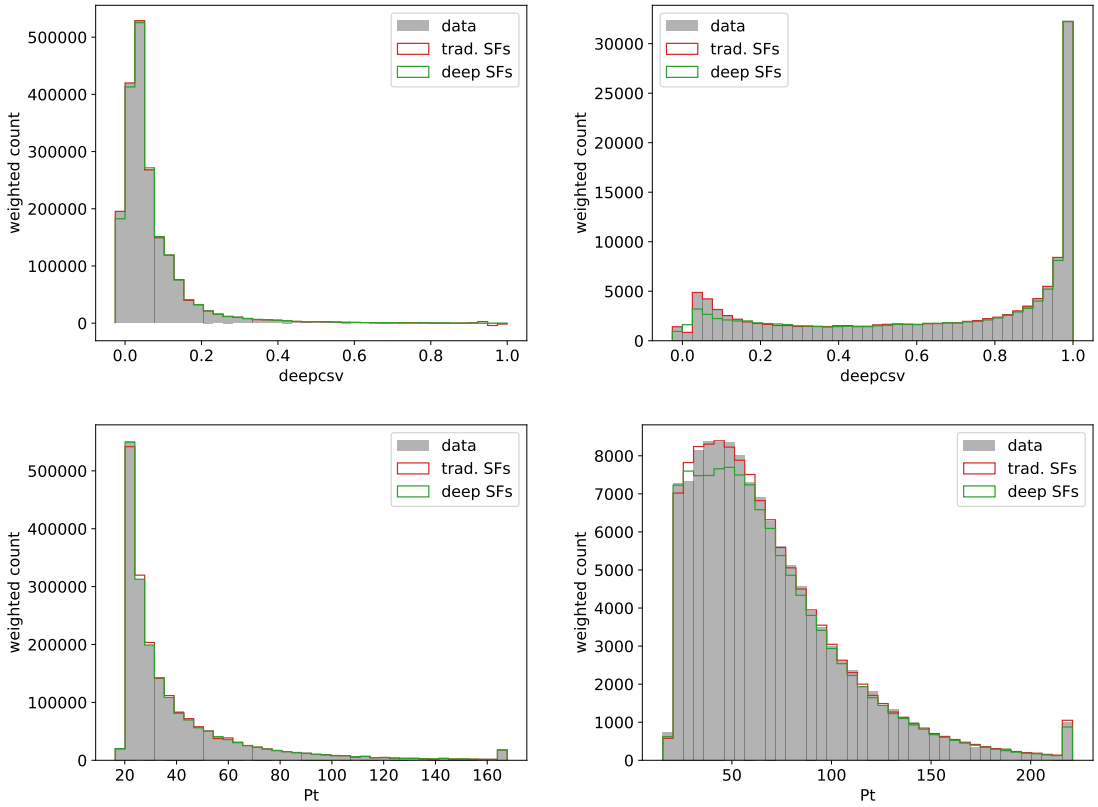

Figure 5.2: Agreement between data (gray shaded) and scaled MC using either the traditional SFs (red) or the deep SFs (green). *Left:* Light flavor sample. *Right:* Heavy flavor sample. *Top:* Histogram over CSV. *Bottom:* Histogram over p_T.

As one can see an overall good agreement exists between the traditionally scaled MC, the deep scaled MC, and the measured data. A positive first result is the improved simulation of the p_T and CSV_c values through the deep SFs in the light flavor region, although CSV_c was not even explicitly fitted against. The traditional

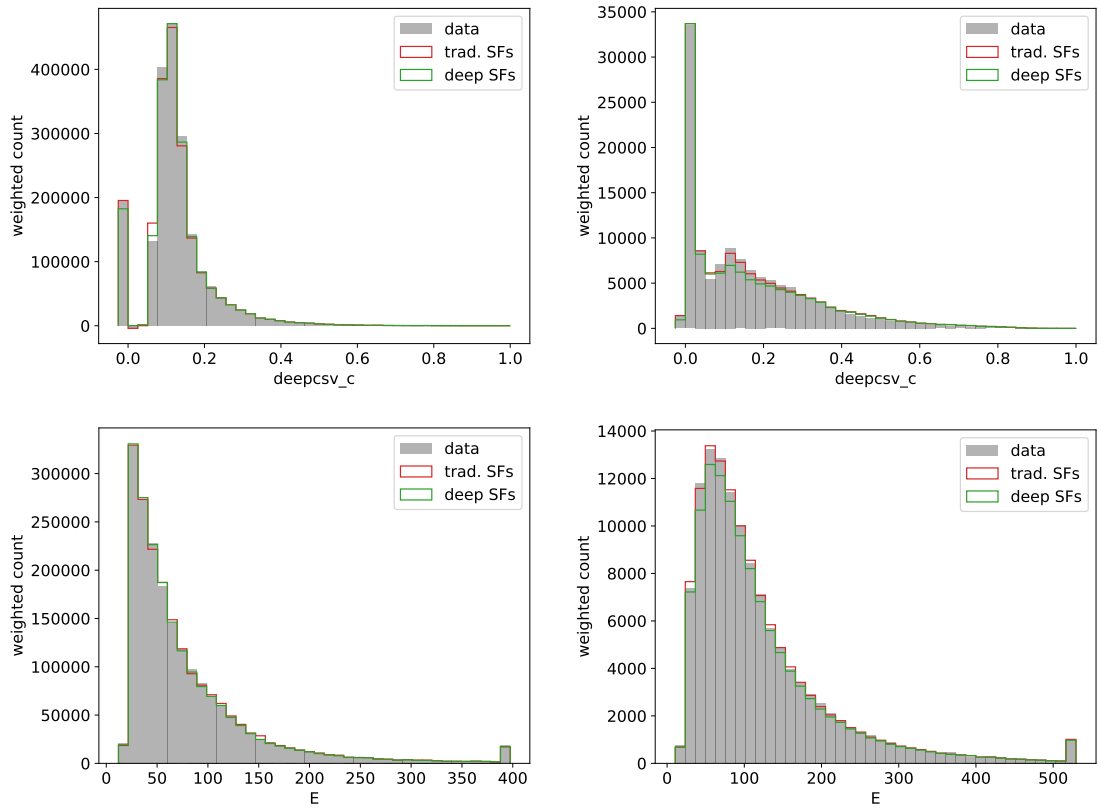

Figure 5.3: Agreement between data (gray shaded) and scaled MC using either the traditional SFs (red) or the deep SFs (green). *Left:* Light flavor sample. *Right:* Heavy flavor sample. *Top:* Histogram over CSV_c. *Bottom:* Histogram over energy E.

SFs suffered from a particularly low number of only 4 bins in the p_T range explaining the deficiency for that variable. As the deep SFs fit continuously across all variables the corresponding weighted simulation fits better the p_T distribution of the data.

In the heavy flavor region, the deep SFs suffer a little bit from the low statistics in the bin for CSV ≈ 0. The explanation for this poor behavior is given through the histogram of the SFs.

Fig. 5.4 shows the overall distribution of the SF values for the two methods. It reveals why the deep SFs do not work well in the heavy flavor region. The traditional SFs reveal a large amount of SFs smaller than 0 (overflow bin). The SF-Net is obviously not very capable of producing such discrete distributions based on small differences in the CSV input. Further investigations might improve on this behavior. The agreement for the light flavor region is acceptable.

A very common visualization of SFs is shown in fig. 5.5. The traditional SFs are plotted over the CSV output for a certain p_T and η bin. The deep SFs do not have a single value. To provide a meaningful comparison the events were binned according to p_T, η, and CSV and the statistics of the SFs inside this bin are visualized through the green bands. The deep SFs have a relatively wide distribution even after the

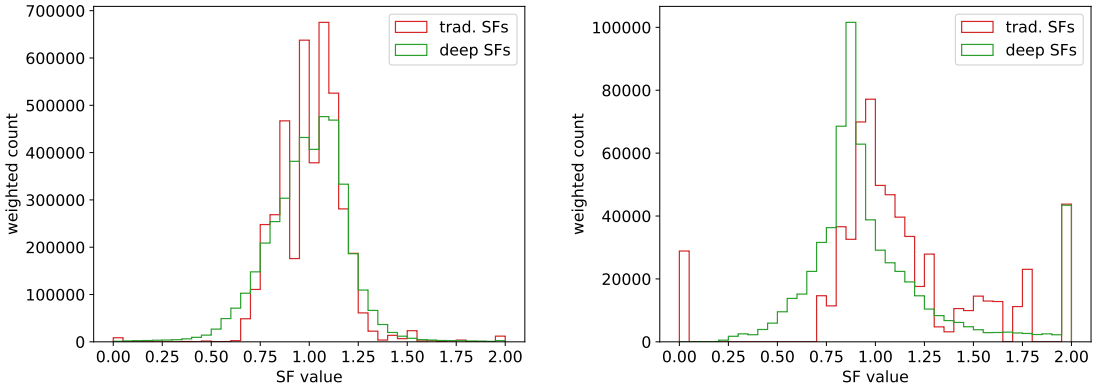

Figure 5.4: The histogram of SFs using the MC event weights to correctly attributed the importance of each SF. *Left:* Light flavor sample. *Right:* Heavy flavor sample.

binning in the three variables. This shows that the DeepSF approach really makes use of the freedom and not being contained on discrete bins.

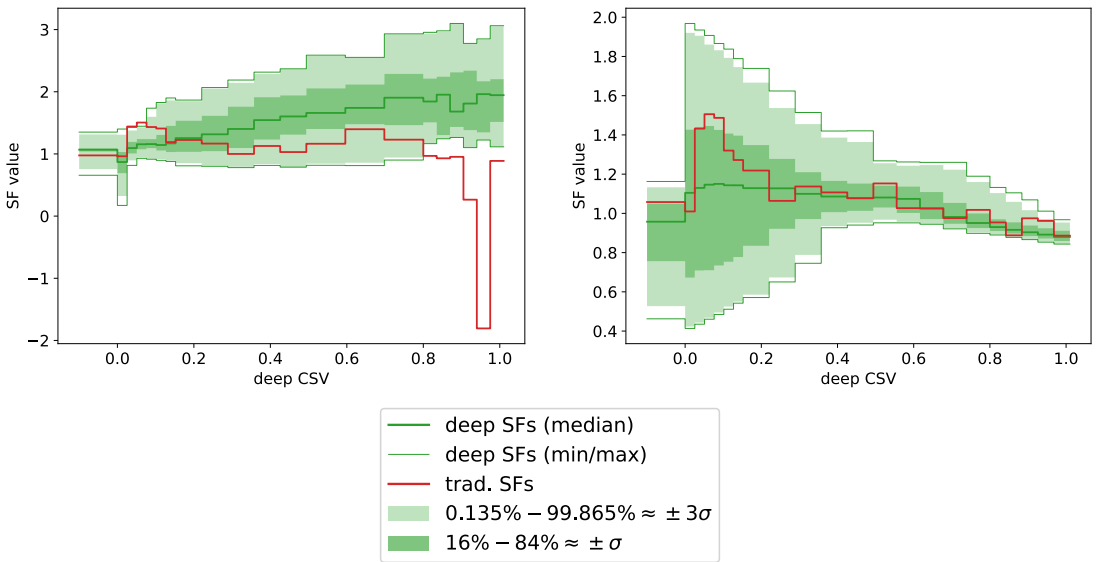

Figure 5.5: Visualization of SF shape: Traditional SFs are shown in red. The bin-wise distributions of the deep SFs are shown in green for different percentiles. *Left:* Light flavor sample in the bin $60\,\text{GeV} \leq p_T$, $|\eta| < 0.8$. *Right:* Heavy flavor sample in the bin $70\,\text{GeV} \leq p_T < 100\,\text{GeV}$, $|\eta| < 2.4$.

The deep SFs also allow investigations which have not been possible with the traditional approach. As the SF-Net is a continuous and differentiable mapping from jet variables to SFs, a potentially interesting question would be which variables strongly

influence the SFs. A measure that could quantify this might be

$$I_X = \frac{1}{\sum w_i} \sum_{(w_i, \boldsymbol{x}_i) \in \text{fake}} w_i \cdot \left| \frac{\partial SF(\boldsymbol{x}_i)}{\partial X} \right|^2 . \tag{5.14}$$

It is important for this measure that the data $\boldsymbol{x}_i$ is normalized. Otherwise, the higher scaling of variables would just scale the importance value of that variable proportionally.

This 'importance' measure for a variable X is shown in fig. 5.6. The CSV value has as expected the highest score on this measure.

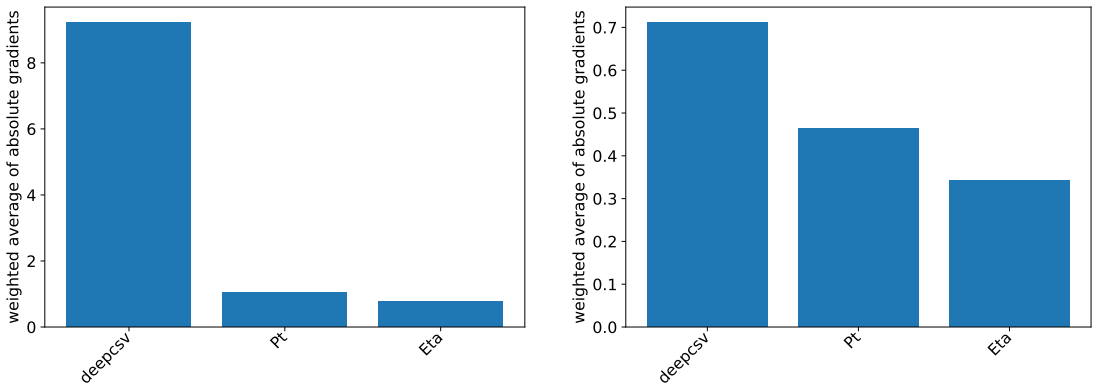

Figure 5.6: *Importance* value I_X of the different variables X as defined by eq. 5.14. *Left:* Light flavor sample. *Right:* Heavy flavor sample.

5.5.2 Extension to more Input Variables

One of the key advantages of the presented method is that it does not depend on the number of input variables. For the purpose of testing a first extension of these variables, the observables energy E and ϕ where included. Instead of the CSV value, all output values of the deep CSV were used independently: CSV_b, CSV_{bb}, CSV_c, CSV_{udsg}

Figure 5.7 displays again some variable histograms. As one can see the situation neither looks better or worse compared to fig. 5.3.

Also, the histograms of the SFs still match quite well (see fig. 5.8). As one would expect the tails of the distribution for the deep SFs got a bit longer as with more variables the SFs get more stronger constrained by the data and therefore showing stronger fluctuations.

The plots of the SFs in the binning of the CSV, p_T and η variables were omitted as they contained no additional insights and look very similar to the ones presented in fig. 5.5. However, the bar diagram with the importance score of the variables has now a few more candidates (see fig. 5.9): Surprisingly, a relatively high importance score was assigned to CSV_c for both, the light and heavy flavor case. Also, the

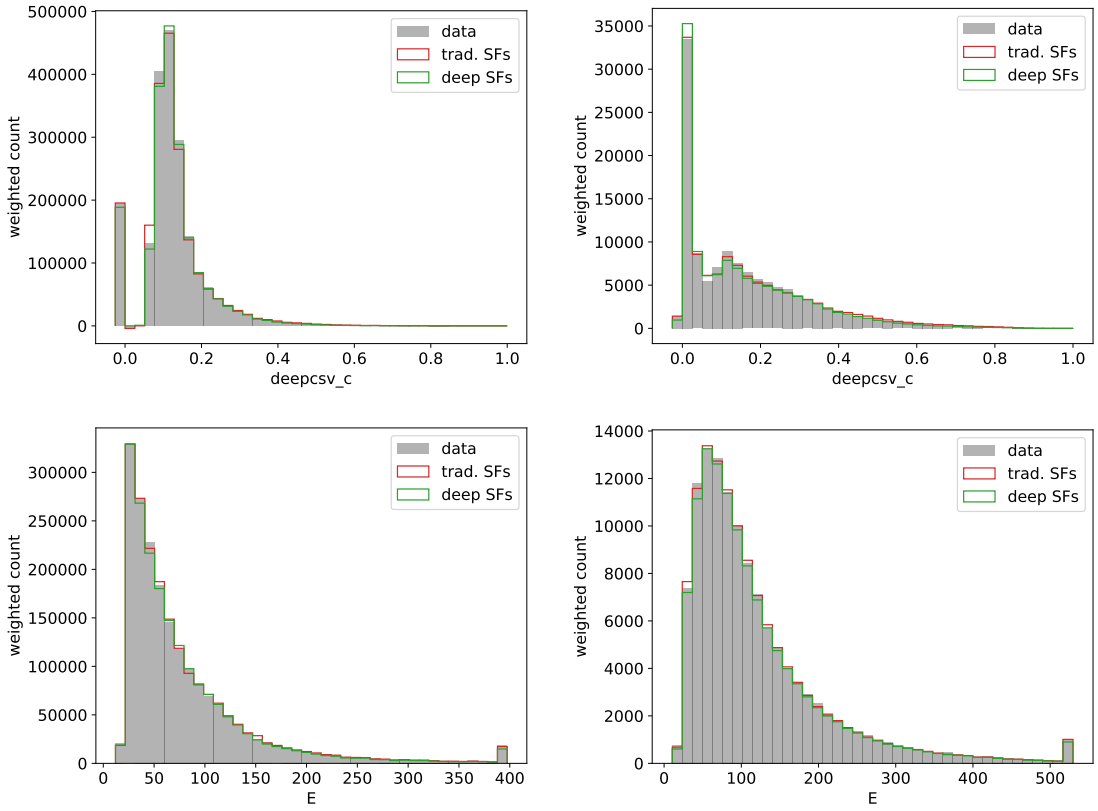

Figure 5.7: Agreement between data (gray shaded) and scaled MC using either the traditional SFs (red) or the deep SFs applied on 8 variables (green). *Left:* Light flavor sample. *Right:* Heavy flavor sample. *Top:* Histogram over CSV_c. *Bottom:* Histogram over energy E.

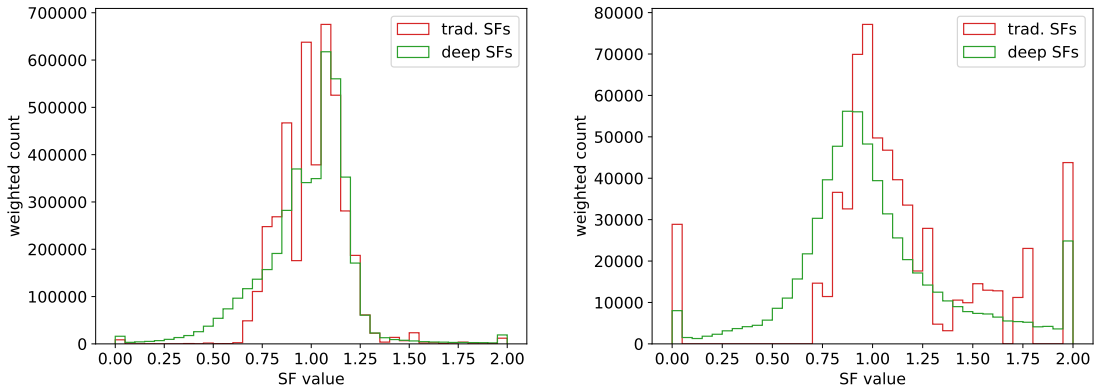

Figure 5.8: The histogram of SFs using the MC event weights to correctly attributed the importance of each SF. The deep SFs are based on 8 input variables. *Left:* Light flavor sample. *Right:* Heavy flavor sample.

energy has gotten a higher score than the traditionally used variable η which seems to be as unimportant as the variable ϕ which is completely decorrelated from every other variable and contains, therefore, no information at all.

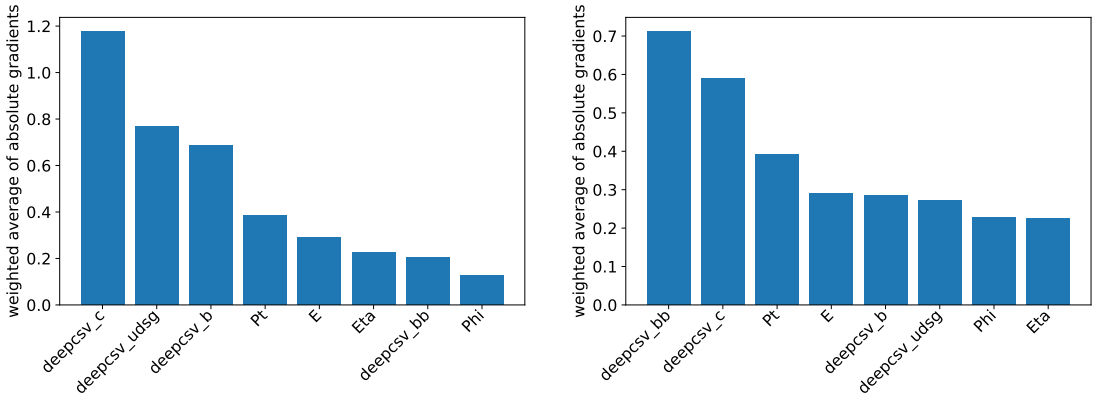

Figure 5.9: *Importance* value I_X of the different variables X as defined by eq. 5.14. *Left:* Light flavor sample. *Right:* Heavy flavor sample.

5.5.3 Summary and Outlook

Overall, this section has introduced a new method of computing SFs using DL techniques. In certain aspects, the produced deep SFs have a close resemblance to the traditional SFs, therefore, validating this method.

A real improvement going from 3 to 8 input variables could not be seen. However, this method is flexible and future modifications could provide interesting results. These future investigations could include:

- Introducing further variables that may be more relevant, for example, the input variables of the CSV (e.g., observables based on the secondary vertex).
- Using different input variables for the discriminator and the SF-Net. A possibility could be to let the discriminator only fit the CSV distribution while the SF-Net is computing SFs based on additional variables.
- Removing the iterative procedure by adding the generated flavor as an input variable to the SF-Net. The training of the networks for heavy flavor and light flavor would happen simultaneously and the traditional iterative procedure would be effectively incorporated into the training iterations.
- Adding uncertainties to the computation. Most of the uncertainties from [63] can be computed in a similar fashion for the deep SFs. As there is no binning involved as well as a reduced issue of low statistics it could be possible that the statistical error is smaller than the traditional counterpart.
- Retaining the old method and using the deep SFs to find possibly more important input variables for the traditional method. As a first result, it might be worth looking into how the traditional SFs would behave when they depend on CSV_c, p_T, and E as these were the most *important* variables as the second part of this section revealed.

In terms of a multi-dimensional fit with DNNs, this method worked remarkably well. Compared to the WGAN it seems that this method was more successful in

producing correct variable shapes. The presumed reason is that the generator in the WGAN has a much harder task to perform. In contrast, the SF-Net was tasked to refine an already well agreeing MC simulation through a single weight value only.

6 Fit of the observed Cosmic Ray Skymap

This section reflects on the development of a novel fit method published under the title 'Origins of Extragalactic Cosmic Ray Nuclei by Contracting Alignment Patterns induced in the Galactic Magnetic Field' [64] submitted to the journal Astroparticle Physics. In this thesis, the focus will be on the DL techniques and the high-dimensional fitting aspects.

This section will introduce a novel method that could potentially give new insights into the origin of UHECRs which is a very relevant still mostly unanswered research question.

As sec. 2.2 has motivated there do not exist a lot of cosmic objects that have the potential to produce UHECRs. However, the measured arrival directions are remarkably isotropic distributed. One important reason for this may be the deflection of the particles through the galactic magnetic field of our galaxy. Thus, the working hypothesis of this section is that the UHECRs arrive at the edge of our galaxy in a less isotropic pattern, meaning that a few of them are expected to originate from the same extragalactic source and therefore arriving from the same direction before entering the galactic magnetic field of the Milky Way. The corresponding null hypothesis is consequently an isotropic arrival of UHECRs.

Current research tries to investigate the possible scenarios by developing algorithms that essentially backtrack measured CRs through the galactic magnetic field and find clusters in the extragalactic arrival directions. These algorithms can then be evaluated on simulations of the different scenarios regarding their discriminating power between isotropic and anisotropic extragalactic arrival directions.

A particular problem with these approaches is, that there is very little information on the charge of the CRs which plays an important role when tracking particles through a galactic magnetic field. Thus, most models make simplifying assumptions that lead to a non-optimal performance.

An idealized algorithm would perform some global fit of the hidden physics variables (extragalactic source direction $\hat{s}$ and charge $\hat{Z}$) for each UHECR. This fit would take all measurements (energy E, arrival direction p, $X_{\max}$) with their respective uncertainties into account and produce a result that aims at clustering the $\hat{s}_i$ as much as the data reasonably allows ($i = 1 \dots N$ being the different CRs). This results effectively in a high-dimensional fit where the generated data is the prediction

$$\hat{p}_i = T \begin{pmatrix} \hat{s}_i \\ \hat{Z}_i \\ E_i \end{pmatrix}$$

which needs to fit the measured p_i resulting in an overall dimensionality[12] of $2N$. With N being in the order of 1000 this seems to be too large even for DNNs. However, an important difference compared to the previous sections is the fact that the data can actually be fitted exactly avoiding the detour using a DNN to encode

[12]The 3-dimensional unit vector p_i is spanning a 2-dimensional space.

the data distribution itself.

The following section will generalize the problem and point out the difficulties and problems that reason the use of the applied DL techniques. Afterward, the exact architecture of the fit will be discussed with a great focus on the loss construction. The different simulated scenarios will be presented in a separate chapter before showing the results.

6.1 Generalization

The task of reconstructing the hidden physics variables based on a model can be formally described as computing an inverse of a function. The physical model is a function with the hidden physics variables $\boldsymbol{p}$ being the input of this function T and the observed data the output $\hat{\boldsymbol{x}} = T(\boldsymbol{p})$. The task is to find the parameters $\boldsymbol{p}$ that produce predicted observations $\hat{\boldsymbol{x}} \approx \boldsymbol{x}$ which fit best with the measured data $\boldsymbol{x}$. A very common practice is to find the solution $\boldsymbol{p}$ via gradient descent over some loss, e.g. $\mathcal{L}(\boldsymbol{p}) = (\boldsymbol{x} - T(\boldsymbol{p}))^2$. For a low number of parameters, this is most of the time straight-forward, as gradients can be computed in an approximate fashion

$$\frac{\partial \mathcal{L}}{p_i} = \frac{\mathcal{L}(\boldsymbol{p} + \epsilon \boldsymbol{e}_i) - \mathcal{L}(\boldsymbol{p} - \epsilon \boldsymbol{e}_i)}{2\epsilon} \tag{6.1}$$

assuming a low computation time of T.

However, in the case of a large number of parameters and a non-trivial model T, this approach becomes highly inefficient as not only the computational cost for computing the gradients increase but also the required gradient descent steps that need to be taken increases with the size of the search space. Traditional approaches can overcome this hurdles by using knowledge and heuristics about T to artificially create some update step that brings one iteratively closer to the solution. One can expect that these updates are not always optimal and the development of such methods is elaborate and consumes a large amount of time.

This section presents a task for which it is feasible to encode the model function T inside a DNN approximating the model while providing an efficient way to actually compute the gradients of $\hat{\boldsymbol{x}}$ with respect to the parameters $\boldsymbol{p}$. This enables us to actually perform a traditional gradient descent.

6.2 Fit Architecture

The basic principles of the architecture will be introduced step-wise by investigating first a 1-dimensional model with a simple transformation T. Afterward, results for the sphere will be shown for the case of a simple relation for T and from there going to the full model using the JF12 parameterization (cf. sec. 2.2.3) for the galactic magnetic field transformation T.

In all three cases the dataset of UHECRs consists of N particles each having a recorded arrival direction p_i, energy E_i and maximum shower depth $X_{\max,i}$. For

each CR the fit has the parameters $\hat{s}_i$ for the source direction (direction outside of the Milky Way) and $\hat{Z}_i$ for the charge number. The hat variables will represent predicted quantities based on fit parameters and will always be real numbers as the method of gradient descent needs to work on a continuous parameter space. One such quantity is the rigidity $\hat{R}_i = \frac{E_i}{\hat{Z}_i e}$. The used transformation law encoding the physical model takes as input the rigidity and the source direction and returns the arrival direction on Earth $\hat{p}_i = T(\hat{s}_i, \hat{R}_i)$.

The three transformation laws in the increasing order of complexity are:

1. $$T(\hat{s}, \hat{R}) = \hat{s} + \frac{1}{\hat{R}} \tag{6.2}$$

   This is the one-dimensional toy case in which physical dimensions are omitted ($e = 1 \Rightarrow \hat{R} = \frac{E}{\hat{Z}}$). $\hat{s}$ and $\hat{Z}$ are on a scale of 0 to 1 while E is a value between 1 and 10 leading to $\frac{1}{\hat{R}} \in [0, 1]$.

2. $$T(\hat{s}, \hat{R}) = R_z\left(-2\frac{\mathrm{EV}}{\hat{R}}\right)\hat{s} \tag{6.3}$$

   $R_z(\theta)$ describes a rotation in 3 dimensions around the z-axis which is perpendicular to the galactic plane. Here we use already physical dimensions. A rigidity of 100 EV (proton with $E = 10^{20}$ eV) would lead to a deflection angle of -0.02 rad $\approx 1°$ which roughly compares to the scale of deflections for the galactic magnetic field.

3. $$T(\hat{s}, \hat{Z}, E) = JF12_{\mathrm{DNN}}(\hat{s}, \hat{Z}, E) \tag{6.4}$$

   In the following, we will refer to this as the full model. The JF12 parameterization of the galactic magnetic field is encoded in so-called lenses, a digital data structure that introduces a discrete mapping. It does not plainly depend on the source direction and rigidity but also on the exact energy and charge value. Because the lenses are discrete, they were used to create a large artificial dataset used for training a DNN. The trained network represents the JF12 model while being differentiable.

All of these transformations are continuously differentiable and the symbolic gradients can be easily computed using some appropriate software framework.

As pointed out in the introduction and the generalization of this task, the most important aspect of this approach is the engineering of the loss which is to be optimized by the fit. The loss should guide the fit in such a way, that the observed data is reconstructed through the choice of parameters while also clustering the source directions as much as possible. To that end, the loss contains two main components guiding the fit.

Distance Loss The distance loss penalizes large deviations of the predicted arrival directions $\hat{p}$ with respect to the measured data p. The loss term is therefore simply the distance norm (in both the 1- and 2-dimensional case):

$$D = \frac{1}{N} \sum_i ||p_i - \hat{p}_i||^2 \tag{6.5}$$

Cluster Loss The aim of this loss is to cluster source directions of CRs that are potentially close enough to originate from the same source. For the first investigations in one dimension, a simple k-Nearest-Neighbor (kNN) clustering was performed. For each source direction, one can compute the average direction of the k nearest neighbors

$$\langle \hat{s}_i \rangle = \frac{1}{k} \sum_{\hat{s}_j \in kNN(\hat{s}_i)} \hat{s}_j \tag{6.6}$$

and require that the distance to this 'local average' should be small (similar to the distance loss):

$$C_{1-dim} = \frac{1}{N} \sum_i ||\hat{s}_i - \langle \hat{s}_i \rangle||^2 \tag{6.7}$$

The disadvantage of this loss is the dependency on the overall number of UHECR N when fixing k which is unphysical. Therefore, a modified version was used for the spherical cases. Instead of selecting in a discrete manner a few potential clustering candidates, a weighted loss term for all inter-source distances was used:

$$C_{sphere} = \frac{1}{N} \sum_{i,j} \epsilon_{ij} ||\hat{s}_i - \hat{s}_j||^2 \tag{6.8}$$

The weights $\epsilon_{ij} = \epsilon(\hat{s}_i, \hat{s}_j) = \cos^{2\gamma} \angle(\hat{s}_i, \hat{s}_j) \in [0, 1]$ have provide a value of close to 1 for $\hat{s}_j$ in the vicinity of $\hat{s}_i$ and dropping to 0 for source directions further apart. As the transformation T provides a preferred deflection direction $\hat{g}_i = \frac{\partial T(\hat{s}_i, \hat{Z}_i, E_i)}{\partial \hat{s}_i}$ that can be easily computed (T is symbolic differentiable) the introduced weights ϵ are also depending on this deflection direction (introducing a slight asymmetry $\epsilon_{ij} \neq \epsilon_{ji}$). The function $\epsilon(\hat{s}_i, \hat{s}) = f(\hat{s})$ is 1 for $\hat{s} = \hat{s}_i$ and decreases over a range of 40° down to 0 along the deflection direction while decaying much faster perpendicular to it (on a scale of 4°). This is implemented using vector algebra and different values of $\gamma = 4.3, 470$ ($0.1 = \cos^{2 \cdot 4.3}(40°) = \cos^{2 \cdot 470}(4°)$) for the parallel, perpendicular component.

An important aspect of this loss term is the fact that even though ϵ_{ij} depends on the source direction parameters its value is regarded as constant with respect to the gradient descent. This leads to the gradient being exactly

$$\frac{\partial C_{sphere}}{\partial \hat{s}_i} = \frac{1}{N} \sum_j (\epsilon_{ij} + \epsilon_{ji})(\hat{s}_i - \hat{s}_j) \tag{6.9}$$

as it is expected to be. Without this technical detail, the fit would actually profit

from moving source directions apart from each other as this decreases ϵ and therefore the total loss to 0.

Charge Loss A third loss component was introduced to exploit the $X_{\max}$ measurement provided by the FD (cf. sec. 2.2.5). Given an atomic mass number A and an energy E, the observed $X_{\max}$ follows a broad, asymmetric probability distribution, called the Gumbel function $G(A, E)$ [65]. Because the distribution is so broad the information gain is quite small but existent as we will see. To successfully exploit this data it is important to not naively fit the charges $\hat{Z}_i \approx \frac{\hat{A}_i}{2}$ in a way such that the Gumbel function $G(\hat{A}_i, E_i)$ has its maximum at $X_{\max,i}$ as this would be a logical error regarding conditional probabilities. Instead, the decision was made to evaluate the χ^2_N-distribution of the measured $X_{\max,i}$ with respect to the fitted value $\hat{A}_i$ and the resulting Gumbel function $G(\hat{A}_i, E_i)$. The resulting loss is

$$Q = \left[\frac{\chi^2_N}{N} - 1\right]^2 = \left[\frac{1}{N}\sum_i \frac{\left(X_{\max,i} - \mathrm{E}[G(\hat{A}_i, E_i)]\right)^2}{\mathrm{Var}[G(\hat{A}_i, E_i)]} - 1\right]^2 \tag{6.10}$$

with $\mathrm{E}[\cdot]$ and $\mathrm{Var}[\cdot]$ being the expectation value and variance, respectively. It describes the notion that the χ^2_N distribution should have a mean value of N.

The resulting total loss used for the fit is the sum of the three components scaled through the use of 2 hyperparameters:

$$\mathcal{L} = D + \lambda_Q Q + \lambda_C C. \tag{6.11}$$

6.3 Fit Procedure

The fit of the parameters was done by implementing the complete workflow in TensorFlow [53]. The computation of the gradients and the optimization are built-in functionalities of TensorFlow and simplify the training a lot. In contrast to classical DL trainings, the parameters were not updated in a batch-wise manner but all at once in every iteration. This decision is not contradicting the usually positive training effects of small batches (cf. sec. 3.1.4) as in this case we do not train a neural network with many optimal solutions but fit physical parameters which will exhibit only a few local minima and all being close to the globally best solution (deflection is bounded). For the cluster loss C and the charge loss Q, it is actually necessary to evaluate each iteration over all data.

Performing a traditional non-stochastic gradient descent allows taking step-sizes of approximately the size of the gradient itself. An average learning rate was approximately 0.1. To increase the speed of convergence a handful of criteria were implemented to dynamically adapt (increase or decrease) the learning rate turning it as high as possible without destabilizing the convergence behavior.

When fitting a large number of arrival directions with the goal of clustering the source directions the charge somehow acts as a mediator between the clustering and

the arrival direction reconstruction. A source direction which the clustering loss pulls strongly into one direction needs to be compensated by a change in the charge so that the predicted arrival direction still keeps matching the data. As the charge loss Q is a very loose constraint on the charges this leads to negative ore extremely large predicted charges. To avoid these deviations artificial upper and lower bounds $Z_{\max}, Z_{\min}$ were introduced through an additional loss term

$$Q_b = \sum_i \max(\hat{Z}_i - Z_{\max}, 0)^2 + \max(Z_{\min} - \hat{Z}_i, 0)^2 \tag{6.12}$$

scaled by an additional hyperparameter λ_b. However, λ_b did not really need tuning as the goal was to force the charge bounds as a strong constraint, so $\lambda_b = 100 \gg 1$ was sufficient to effectively avoid charges outside of the bounds completely.

The other two hyperparameters $\lambda_Q = 0.1$ and $\lambda_C = 0.01$ were fixed through some simple experimentation and manual hyperparameter tuning. The target was that the set of parameters would produce a fit which converges on a significantly higher loss when performed on an isotropic source scenario compared to an anisotropic scenario (cf. next sections). The values worked well for both, the 1-dimensional and spherical scenarios.

For a gradient descent fit producing decent results a good initialization of the parameters is important. As these are depending on the scenario they will be highlighted in the upcoming sections.

6.4 Results

The first section visualizes and inspects the behavior of the fit for the 1-dimensional case. With the deepened understanding of the fit, the spherical scenario with a simple rotation as deflection will reveal the differences between the 1-dimensional and spherical geometry and the change of the cluster loss as introduced in sec. 6.2. The final scenario will be the highlight, as it represents a realistic scenario similar to data taken by the Pierre Auger Observatory.

Throughout this chapter, the charges will be shown as color-coded elements, while the size of the elements (bar or dot) represents the energy. The true source positions will be represented through star symbols. The variables Z and s represent the true charge and source position with the hat-marked variables being the corresponding fitted values.

6.4.1 1-dimensional Scenario

This toy scenario makes use of dimensionless variables $s \in [0, 1]$ and $E \in [1, 10]$. The charge $Z \in [0, 1], Z_{\min} = 0, Z_{\max} = 1$ is actually discretized into $Z = c/26$ for $c = 1, \ldots, 26$.

The first toy data scenarios consist of 10 CRs with the source direction, energy and charge being uniformly distributed. The shower depth values $X_{\max}$ were randomly

picked according to the Gumbel function $G(A = 2c, E)$ The observed arrival directions p_i are computed according to T from eq. 6.2. This leads to an isotropic scenario (s uniformly distributed) shown in fig. 6.1. The figure also contains the corresponding anisotropic scenario which differs only with respect to the s_i being all equal to a singular, randomly placed, source position.

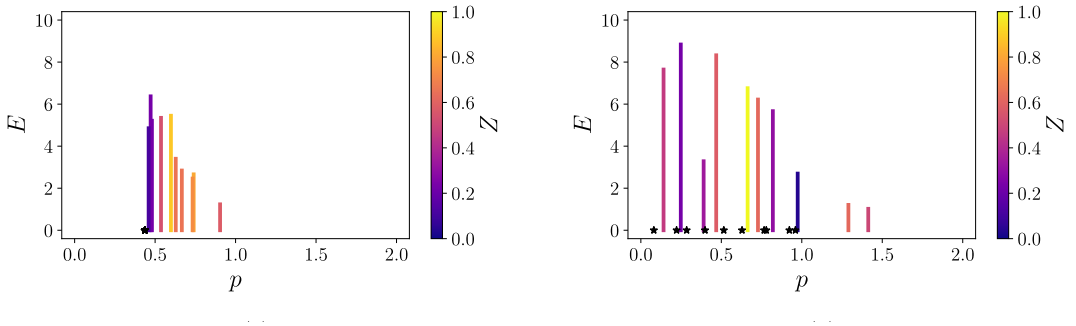

Figure 6.1: 1-dimensional toy scenario of ten CRs. Shown are the observed values $p_i = T(s_i, R_i)$ together with the energy and charge as bar height and color, respectively. The source positions s_i are shown by star-shaped markers. *Left:* Anisotropic scenario with 10 CRs originating from a single source. *Right:* Isotropic case with 10 randomly sampled source positions.

Even though the fit has no precise information on the charge Z, the pattern of the energies and arrival directions p differ greatly between the isotropic and anisotropic scenario, so we expect the fit to easily handle this case.

The initialization of the charges is set fix to 0.5, while the initial source directions are set by inverting the transformation T to retrieve $\hat{s}_{i,0} = p_i - \hat{Z}_i/E_i$. The number of neighbors that are considered for the clustering loss C is $k = 10$. Over the course of only 200 iterations, the fit is well converged and the change of the source directions visualized in fig. 6.2. In the case of anisotropy, the fit manages to find the source and corresponding charge values that are well able to explain a clustering of all 10 CRs. However, in the isotropic case clustering fails, as the particles cannot be reconstructed from a single source while obeying to the charge limits and the measured $X_{\max}$ distribution.

To express this desired behavior in a statistically significant way, this fit was performed for 100 anisotropic and 100 isotropic cases. The quantities $\Delta Z = \hat{Z} - Z$ and $\Delta s = \hat{s} - s$ are histogrammed over these 100 runs in fig. 6.3 and clearly show that the fitting of charges and source positions works in the anisotropic case while failing in the isotropic one.

The reason why the exact source direction can be reconstructed in the anisotropic case is actually not completely trivial. The reason lies in the number of CRs that get clustered. Through the hard boundaries on the charge in addition to the $X_{\max}$ constraint, there is for each CR a small interval of possible source directions. When multiple CRs belong to the same cluster then the source position of these particles can more precisely be reconstructed as it has to lay in the intersection of all the

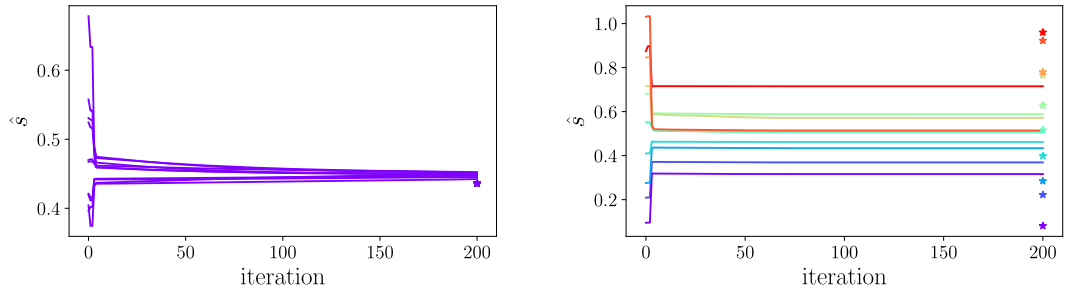

Figure 6.2: The fitted parameters $\hat{s}_i$ shown over the course of the fit. The stars on the left are the original source positions s_i. The color encodes which prediction belongs to which true source position. In both cases, the fit was performed with $k = 10$ neighbors in the cluster term. *Left:* The anisotropic case of 10 CRs. The fit successful reduces the pattern alignment into a single source. *Right:* The corresponding isotropic case. The cluster term is not able to bring the source positions in perfect agreement because of the limited charge values.

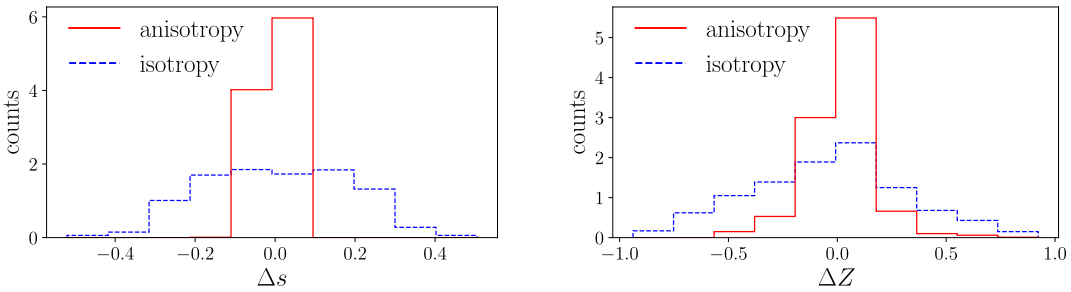

Figure 6.3: The error of the reconstructed source (*Left*) and the charge (*Right*) are shown for each CR. The solid red line represents an average of 100 anisotropic scenarios (10 CRs from a single source). The blue dashed histogram marks the average of 100 isotropic scenarios with 10 CRs each.

different intervals of each CR. This principle is visualized in fig. 6.4. This statement will also hold in the spherical scenarios: The more CRs are correctly clustered into a single original source, the more accurate the reconstruction of this position will be. Additionally, the charge gets also more constrained through the source and arrival directions leading to an improved charge reconstruction compared to information gained from $X_{\max}$.

The fully anisotropic case is of course extremely unrealistic. We actually want to quantize how well this method can discriminate a noisy, yet anisotropic scenario from an isotropic one. For this purpose, three scenarios with 100 CRs were constructed. One was again fully isotropic, while one was fully anisotropic, meaning all CRs originate from a single source. The third scenario contains 50 CRs originating from a single source overlayed with 50 isotropic CRs as noise. As a discriminating value, the converged loss was used. Fig. 6.5 shows histograms of these objective values

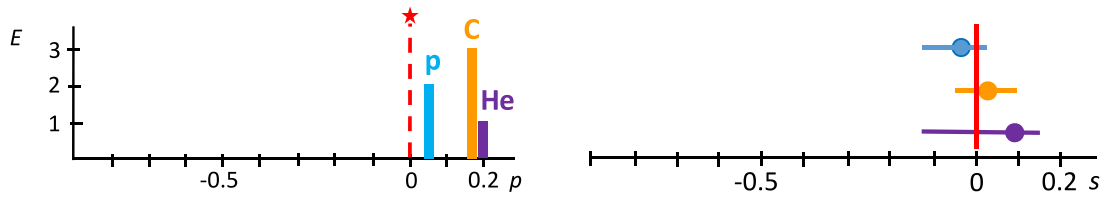

Figure 6.4: A simple example explaining how the reconstructed source position improves with the number of clustered CRs. *Left:* The source position as star and three particles with different charges and energies deflected according to T. *Right:* The dots mark the reconstructed source using the most probable charge obtained from $X_{\max}$. The horizontal extensions mark reconstructed source positions using a charge varied 1 sigma up- and downwards according to the Gumbel functions. Under the assumption that all the particles originated from the same source the different reconstructed estimates can be combined to get a more precise reconstruction of the true source position.

(each scenario was run 100 times), where the scale is normalized to the largest value, as the absolute value does not contain any information. Though still half of the CRs contain an anisotropic signal the loss values start already to significantly overlap. However, one has to consider that the pattern is almost unrecognizable for humans as the arrival directions of the 100 CRs are quite dense positioned in the 1-dimensional case. In the spherical scenario, this situation improves a lot as the deflection will still be along a 1-dimensional locally predominant direction while the CRs arrival directions actually distribute along a 2-dimensional surface.

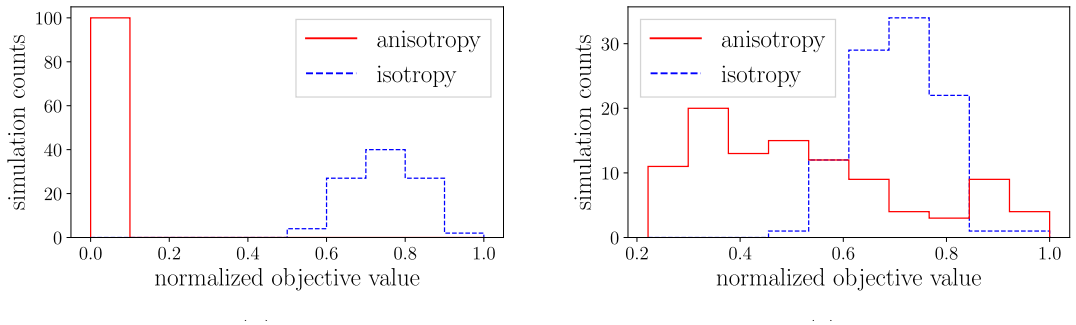

Figure 6.5: The final, converged loss values $\mathcal{L}$ from eq. 6.11 normalized to the largest observed value. For both, the anisotropic and isotropic case, 100 fits were performed to create the histogram. The blue dashed line represents the isotropic case with 100 CRs. The red solid histogram is the anisotropic case with (*Left*) 100 CRs originating from a single source (*Right*) 50 CRs originating from a single source and 50 isotropic CRs.

6.4.2 Scenario of Spherical Rotation

In this scenario the source directions are uniformly sampled from the unit sphere whereas the energy is uniformly sampled from the interval 40–100 EeV. The charges

$Z = 1, \ldots, 26$ are sampled from a uniform distribution between proton and iron. The applied transformation is eq. 6.3 and therefore rotates with angles between $\delta = 0.02 \ldots 1.3\,\text{rad} \approx 1 \ldots 74°$. Again, the corresponding $X_{\max}$ values are simulated using the Gumbel functions and the arrival directions computed with T from eq. 6.3.

In this setting, a number of 100 CRs was used with the fully isotropic case again simply randomly picking a source direction for every single CR. The anisotropic scenario consists of 10 sources each emitting 10 CRs. The anisotropic scenario is shown in fig. 6.6.

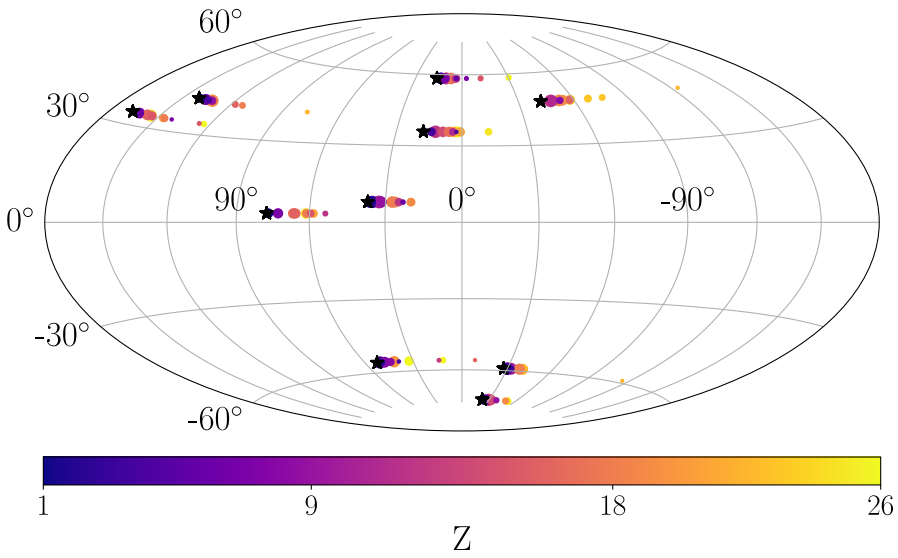

Figure 6.6: Map of arrival directions p_i based on the source positions (stars), their charge (color) and their energy (size) using the z-rotation T from eq. 6.3. In total there are 10 sources emitting each 10 CRs. The energies range between $E = 40 \ldots 100\,\text{EeV}$.

For the initialization of the parameters $\hat{Z}_i$ the Bayes' theorem can be applied to transform any prior probability distribution of $\hat{Z}_i$ in combination with $X_{\max,i}$ into a posterior charge distribution. The average of this posterior distribution was then taken as the charge initialization. The used prior was a uniform distribution $\hat{Z}_i = 1, \ldots, 5$. The reason for so low apriori charges is that in general CRs with a large rigidity and therefore low charge are less deflected and thus more informative concerning the reconstruction of sources. By assigning apriori low charges to all CRs ensures that we do not spoil any of the potential informative CRs through a high initial charge value.

The initialization of the $\hat{s}_i$ is simply p_i. Even though one could compute the inverse of the transformation T as was done for the 1-dimensional case, the initialization of $\hat{s}_i = p_i$ is much simpler and also applicable for the full scenario, where the inverse of T is not feasible anymore.

The resulting fit reveals almost perfect results. Figure 6.7 shows the fit of the

anisotropic and the isotropic case. Except for two outliers[13] all alignment patterns have been contracted to the right source position in the anisotropic case. The reason why this fit works so well is the modified clustering loss for the spherical case as it includes knowledge of the transformation. The clustering is performed over a wider range along the gradients of the transformation. In the case of a rotation around the z-axis, this gradient is exactly parallel to the latitudinal lines. The clustering along these lines happens over longer distances while alignment patterns that are parallel next to each other do not influence each other as their distance vector is perpendicular to the transformation's gradient. As expected the fit is not successful on the isotropic data. As the transformation only performs z-rotations the CRs are clustered in a vertical fashion as this is the component that cannot be contracted through the change of the charge.

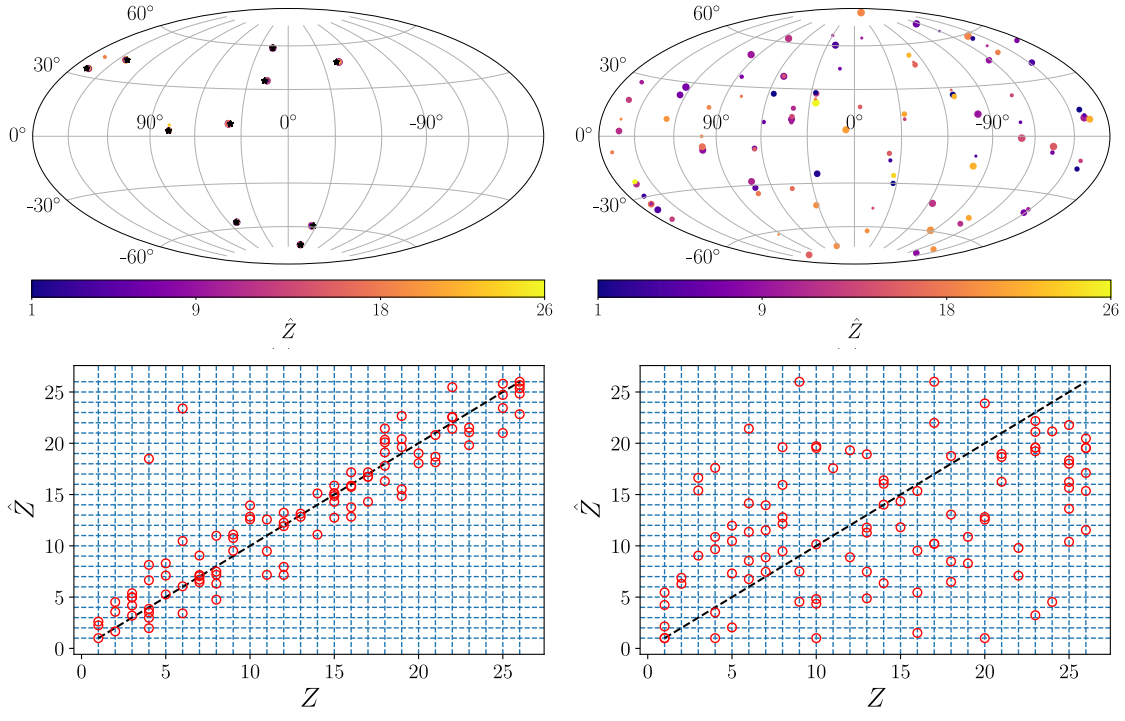

Figure 6.7: Fit results for the spherical scenario with T being a z-rotation. *Top:* The reconstructed source positions $\hat{s}_i$. Charges are denoted by color and the energy by the size. *Bottom:* The reconstructed charge values $\hat{Z}_i$ plotted against the true charge Z_i. The grid shows the discrete values of the true charge. *Left:* The anisotropic scenario consisting of 10 sources emitting 10 particles each. *Right:* The isotropic case with 100 CRs.

Again, it is also interesting to see in fig. 6.7 that the charge reconstruction works in the anisotropic case. In this plot, one can also clearly make out the two outliers. In the isotropic case, the charges are not really reconstructed. However, there is a slight correlation between the predicted label $\hat{Z}_i$ and true charge Z_i visible in the

[13] The outliers could not be fitted as they got pulled into the wrong cluster and converged into a local minimum. Local minima are of course always problematic for gradient descent.

plot because the charge loss Q actually provides some information on the charge.

In the 1-dimensional case, the converged loss value was used to discriminate between isotropic and anisotropic scenarios. In the 2-dimensional case, a quantity was used which is called top-hat counting [66]. The top-hat count of a CR is the number of neighbors (in the space of source directions $\hat{s}_i$) within a certain vicinity. For this analysis, this region was defined by a radial distance of 5°. The histogram in fig. 6.8 is constructed by taking the top-hat count value for each CR. As expected for the anisotropy almost all CRs have a top-hat count of 9–11, as by construction the original clusters had exactly 10 emitted particles. In contrast, the isotropic case shows very low top-hat counts not exceeding 4. This measure was extensively used in the evaluation of the upcoming full scenario.

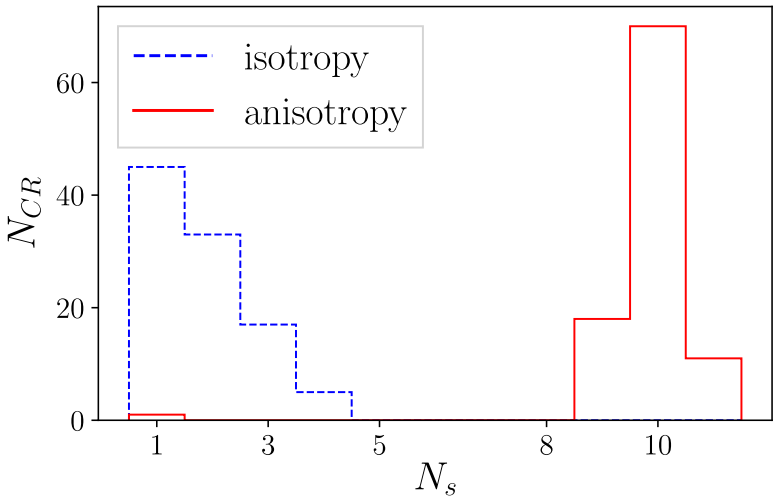

Figure 6.8: The red solid line is the histogram of top-hat counting from all CRs in the case of the anisotropic scenario (10 sources à 10 CRs). The blue dashed line is the result of a random isotropic scenario. The top-hat counting was performed over a region around $\hat{s}_i$ with a radial distance of 5°.

6.4.3 Full Galactic Magnetic Field Scenario

This scenario was constructed to provide a result that is close to reality. In total 1000 CRs are used which follow the energy spectrum from [67] above $E_{\text{min}} > 40\,\text{EeV}$. In the anisotropic scenario, 10% of the CRs came from signal sources. More explicitly, from the 1000 CRs 900 were left isotropic while the last 100 were split up into 4 sources each emitting 25 CRs. With 1000 CRs above 40 EeV this corresponds roughly to the amount of data taken by current experiments like the Pierre Auger Observatory. The charges were taken randomly from between proton and oxygen ($Z = 1, \ldots, 8$) and the Gumbel functions yield the corresponding X_{max} values. The simulation of the arrival directions p_i was performed using the regular JF12 simulation, instead of the approximated transformation T provided through a DNN.

A Gaussian smearing of

$$\sigma = 0.5 \frac{Z}{E/\mathrm{EeV}} \mathrm{rad} \tag{6.13}$$

was also applied to the arrival directions accounting for turbulent components in the galactic magnetic field.

A possible scenario is depicted in fig. 6.9 where the CRs stemming from the signal sources are gray shaded. To the left of the figure, the fit result is visualized where the isotropic background is gray. The signal CR patterns got well contracted with the clusters being also relatively close to the original source. Using the full magnetic model, however, shows also a new behavior. The earlier scenarios have shown that isotropic CRs get also clustered, however, these false clusters were not as strong as the caused by anisotropies. Additionally, the false clusters were equally distributed across the whole map. With the realistic magnetic field there exist extragalactic arrival directions that can be deflected to a large arrival area on Earth. These source directions are said to have a large transparency. On the other hand, there exist arrival directions with a low transparency which is the case when a large source direction space is mapped onto a small region in the Earth's sky. As the fit's goal is to cluster source directions while keeping the arrival direction intact, it moves a lot of the source directions into the regions of high transparency as this automatically clusters the directions while requiring only small adjustments in the exact source position and charge to reconstruct the arrival direction on Earth.

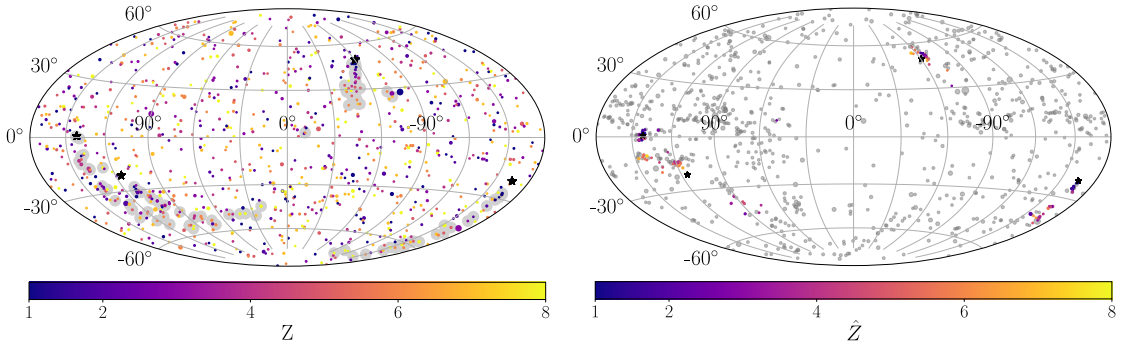

Figure 6.9: *Left:* The arrival directions p_i of 1000 CRs (900 isotropic, 4 sources à 25 CRs) in a realistic energy and charge scenario using the JF12 parameterization of the galactic magnetic field. An additional Gaussian spread was performed (see eq. 6.13). The gray shaded particles originated from the 4 sources (stars). *Right:* The reconstructed source directions $\hat{s}_i$ and charges $\hat{Z}_i$. The gray CRs are the isotropic background.

Because of the changing transparency of the magnetic field the top-hat counting would not give the right impression of the clustering when applying in the same way as it was done for the rotational transformation. To correct for this effect a map was constructed measuring the average top-hat count $\langle N_s^{\mathrm{iso}} \rangle$ in the case of an isotropic scenario with 1000 CRs. With this quantity it is possible to measure a relative top-hat counting $N_{\hat{s}_i} / \langle N_{\hat{s}_i}^{\mathrm{iso}} \rangle$ which is histogrammed for the presented anisotropic case and an isotropic one in fig. 6.10.

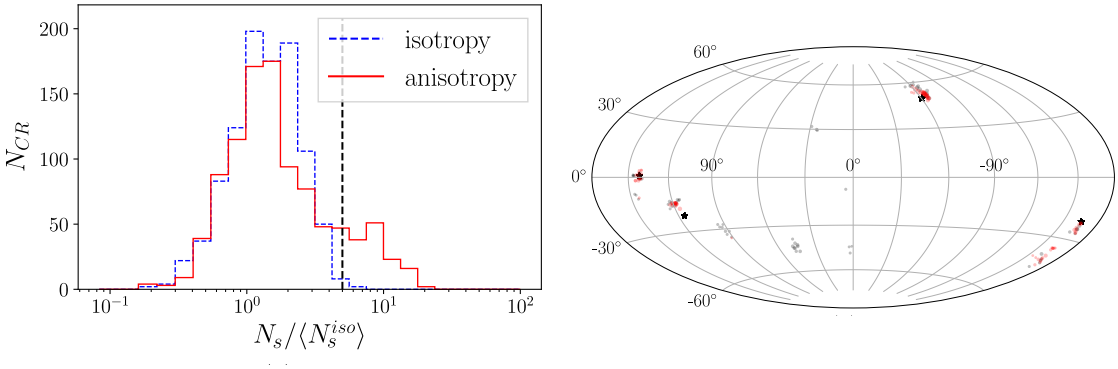

Figure 6.10: *Left:* The relative top-hat countings histogrammed over all CRs. The black dashed line marks the threshold value of 5 above which isotropic scenarios rarely produce any entries. The presented anisotropic scenario has a significant amount of CRs with a relative top-hat count larger than 5. *Right:* The arrival map filtered for all CRs with a relative top-hat count larger than 5. The size represents the relative top-hat counting. The red markers represent signal CRs whereas the gray symbols are isotropic background.

As expected, the anisotropic scenario has many CRs that have a relative top-hat count larger than 5 while the isotropy barely produces any CRs with $N_s/\langle N_s^{\text{iso}}\rangle > 5$. Based on this observation a threshold of 5 was used for further investigations of significant clustering. As an example, it is now possible to only visualize source positions in the fitted map that exceed the relative top-hat count of 5. This map is shown in fig. 6.10 with the size corresponding to the relative top-hat counting. The color indicates whether the CR originated from a source or from the isotropic background. Almost all signal CRs contribute to some significant clustering and are close to the true source position. Only very few clusters of background CRs are visible all still having a small relative top-hat count.

Finally, we want to evaluate the discrimination power of this method. For this purpose a single test value is needed. The used value is the count of significant CR clusterings measured through the number of CRs that have a relative top-hat counting exceeding the threshold 5:

$$N_{CR}(N_s/\langle N_s^{\text{iso}}\rangle > 5) = \left|\left\{ i \in \mathbb{N} : \frac{N_{\hat{s}_i}}{\langle N_{\hat{s}_i}^{\text{iso}}\rangle} > 5 \right\}\right| \tag{6.14}$$

The distribution of these test values is shown for 50 anisotropic scenarios (red curve) and 500 isotropic scenarios (blue-dotted curve) in fig. 6.11. The median test value for an anisotropic case is roughly around 93 which is close to the actual 10% signal CRs out of the 1000. From the 500 isotropic cases only one reported a test value larger than this median. Thus, the expected sensitivity in terms of a p-value is roughly $p = \approx 2 \times 10^{-3}$ for rejecting the null hypothesis of an isotropic origin. This corresponds to 2.9 standard deviations. The formulated expectation is of course heavily depending on the fraction of signal CRs being around 10%.

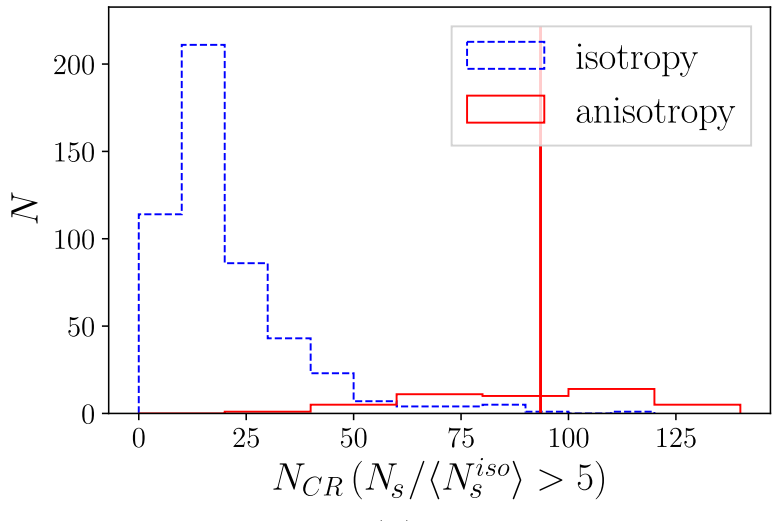

Figure 6.11: The test value (number of CRs with a relative top-hat counting larger than 5) evaluated for 50 anisotropic cases, similar to the one discussed in the previous plots, and 500 isotropic test cases. The median value for the anisotropic cases is around 93. Only 1 out of the 500 isotropic scenarios got a test value larger than that.

6.4.4 Summary

A novel fit method was introduced which reconstructed hidden physics variables $\hat{Z}_i$ and $\hat{s}_i$ by fitting the predicted observations with actual measurements. A plain mean squared error for the distance loss could be applied as the goal was to reproduce the exact dataset instead of the underlying distribution. This is the reason why the actual fitting part of the architecture is much simpler compared to the first two methods (see sec. 4, 5) introduced in this thesis.

Instead, one of the main challenges was to translate the physically motivated transformation law T into a differentiable structure. As it turns out, the transformation T given by the galactic magnetic field model JF12 can be very well captured by a DNN which was trained to reproduce the same results.

The loss function is the heart of this method as it directs the fit to produce the desired results. While the distance loss is straight-forward, the charge loss was carefully constructed to not put a too strong constraint on the parameters. The cluster loss took the deflection direction of the transformation T into account to cluster only CRs that have the potential to originate from the same extragalactic direction.

An important advantage of this method is its flexibility. One can demonstrate this by suggesting some future modifications and improvements:

- To include additional measurements, e.g. a new observable that constrains the charge better, one just needs to add an additional loss term through only a few lines of code.

- In the case of strongly varying uncertainties one could include weights in the distance loss to account for those.
- The cluster loss uses already the direction of the magnetic field to include only favorable cluster constellations. A possible next step would be to also take the predicted charges into account as the current value gives information on how far the source direction can be corrected in each direction through the charge.

Overall this approach has shown promising results with an expected sensitivity of $p = 0.2\%$ for anisotropic scenarios with 10% signal against isotropy. Additionally, the fitted source directions could give rough hints to the locations of the true sources.

7 Conclusion

This thesis presented three different methods that involved fitting between two multi-dimensional data distributions using modern Deep Learning (DL) techniques in the context of particle physics applications.

Traditionally, multi-dimensional data is evaluated and compared by extracting important features using expert-knowledge of the data. In contrast, the presented methods used gradient descent through backpropagation and Deep Neural Networks (DNNs). This thesis has shown that these tools are much more appropriate as they provide a general way of multi-dimensional fitting which is not biased against some human-engineered features. The gradient descent allows for an efficient navigation in large multi-dimensional space (cf. sec. 6) while the DNNs provide the ability to encode discrete data distributions in a continuous manner thus not suffering from low phase space densities caused by the many dimensions (cf. sec. 4, 5). One of the most important aspects of each method was the used loss function which guides the fit to the desired solution.

The first method, which trained a generator DNN to simulate Monte Carlo events, introduced the Wasserstein distance and how it can be used inside the loss function. As a well-defined metric between two data distributions, it was the main reason for an increase in performance and training stability compared to more common objectives.

Instead of generating data, the second method's goal was to produce sample weights for data events that would refine an already existing simulated dataset to match more closely a measured dataset. A novel and custom-designed loss function was presented which was very successful in producing the weights using a DNN output. The differentiable mapping of input events to output weights allowed the inspection of the functional dependence of the weights with respect to the different input variables. This has been impossible with the traditional method. Additionally, this method can safely be extended to more input dimensions making it an interesting alternative for future applications.

An important foundation for both of these methods was the so-called adversarial concept. This concept describes the usage of a second network which is trained simultaneously with the target DNN. The second network is called an adversary, as its output is used in the loss function of the target network. The usage of an adversary in both methods made it possible to use the discriminating power of a DNN as a training target guiding the target network towards a favorable solution.

The third method presented a novel fit in the context of reconstructing extragalactic source directions. The fit features a purpose-built loss function which directly encodes the underlying scientific question of anisotropy in the source directions. Additionally, the objective made also use of a non-differentiable physical model of the galactic magnetic field. As the objective needs to be differentiable, a simple yet very successful approach was carried out. The model of the field was encoded into a DNN through an ordinary DL training, not only making this fit possible but also showing overall promising results.

The usage of DL and more importantly, the construction of the loss function was an essential ingredient in all three cases. The development of the objective was done in a mathematically motivated top-down approach, meaning that the formulated goals instead of the technical possibilities were used as starting points. This leads to a more justified and creative use of the available tools and worked well for constructing task-specific objectives.

Overall, this thesis has shown that DNNs are a very appropriate tool for comparing and fitting multi-dimensional data. As DL offers a wide variety of techniques, a lot of tasks in the domain of multi-dimensional fits can be solved using similar strategies to the ones presented here. The usage of DNNs allows the methods to use more raw data and less human-engineered features resulting in an optimized extraction of the contained information. The most essential part of most DL setups is the loss function. With the many possibilities of constructing such an objective term within only a few lines of code, it is relatively simple to adjust it for different purposes leading to a fast development cycle.

A. High-level Variables

List of (in total 26) high-level variables that were used in sec. 4:

- Aplanarity
- Centrality
- Fox Wolfram 0–4 (5 variables)
- Jet Min/Max/Avg Abs Deta (3 variables)
- Jet Min/Max/Avg Dr (3 variables)
- Jet Closest Pair (125 Mass, 125 Pt, Mass, Pt) (4 variables)
- Jet Lep Min/Max Abs Deta (2 variables)
- Jet Lep Min/Max Dr (2 variables)
- Jet Sum Pt
- Sphericity
- Transverse Sphericity
- Whad Helicity
- Wlep Helicity

B. Bins and Selection Cuts for b-tagging SFs

	light flavor region	heavy flavor region
p_T[GeV]	20, 30, 40, 60, ∞	20, 30, 50, 70, 100, ∞
$\|\eta\|$	0, 0.8, 1.6, 2.4	0, 2.4
CSV	$-2.01, 0.0, 0.0254, 0.0508,$ 0.0762, 0.1016, 0.127, 0.1522, 0.2205, 0.2889, 0.3573, 0.4257, 0.4941, 0.5961, 0.6981, 0.8001, 0.835, 0.87, 0.905, 0.94, 0.975, 1.01	$-2.01, 0.0, 0.0254, 0.0508,$ 0.0762, 0.1016, 0.127, 0.1522, 0.2205, 0.2889, 0.3573, 0.4257, 0.4941, 0.5553, 0.6165, 0.6777, 0.7389, 0.8001, 0.842, 0.884, 0.926, 0.968, 1.01

Table B.1: Binning used for the traditional scale factors (SFs).

Selection Cuts for the Tag & Probe control region:
Heavy flavor enriched sample (dileptonic ttbar)

- 2 leptons ($ee/\mu\mu/e\mu$) with opposite charge
- exactly 2 jets

- $|m_{ll} - 91| > 10\,\mathrm{GeV/c^2}$ for $ee/\mu\mu$ events
- $E_T^{\mathrm{miss}} > 30\,\mathrm{GeV}$ for $ee/\mu\mu$ events
- tag jet must pass CSVv2M (medium) working point: CSVv2 > 0.800

Light flavor enriched sample

- 2 leptons ($ee/\mu\mu$) with opposite charge
- exactly 2 jets
- $E_T^{\mathrm{miss}} < 30\,\mathrm{GeV}$ for $ee/\mu\mu$ events
- $|m_{ll} - 91| < 10\,\mathrm{GeV/c^2}$
- tag jet must fail CSVv2L (low) working point: CSVv2 < 0.460

References

[1] A. M. Sirunyan, A. Tumasyan, *et al.*, "Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV", *Journal of Instrumentation*, 2018. DOI: 10.1088/1748-0221/13/05/P05011.

[2] J. Lin, M. Freytsis, I. Moult, and B. Nachman, "Boosting H to bb with Machine Learning", Jul. 2018. arXiv: 1807.10768.

[3] S. Chatrchyan, G. Hmayakyan, *et al.*, "The CMS experiment at the CERN LHC", *Journal of Instrumentation*, 2008. DOI: 10.1088/1748-0221/3/08/S08004.

[4] The Pierre Auger Collaboration, A. Aab, *et al.*, "The Pierre Auger Observatory Upgrade - Preliminary Design Report", Apr. 2016. DOI: 10.1051/epjconf/201713602003.

[5] C. Burgard and D. Noll, *Standard model of physics - TikZ example*, 2016. [Online]. Available: http://www.texample.net/tikz/examples/model-%20physics/.

[6] F. Capozzi, E. Di Valentino, *et al.*, "Global constraints on absolute neutrino masses and their ordering", *Physical Review D*, 2017. DOI: 10.1103/PhysRevD.95.096014.

[7] C. Patrignani, K. Agashe, *et al.*, "Review of particle physics", *Chinese Physics C*, 2016. DOI: 10.1088/1674-1137/40/10/100001.

[8] M. D. Schwartz, *Quantum Field Theory and the Standard Model Providing*. 2014. DOI: 10.1080/00107514.2014.970232.

[9] P. W. Higgs, "Broken symmetries and the masses of gauge bosons", *Physical Review Letters*, 1964. DOI: 10.1103/PhysRevLett.13.508.

[10] E. Mobs, *The CERN accelerator complex*, Jul. 2016. [Online]. Available: http://cds.cern.ch/record/2197559.

[11] L. Evans and P. Bryant, "LHC Machine", *Journal of Instrumentation*, 2008. DOI: 10.1088/1748-0221/3/08/S08001.

[12] *Interactive Slice of the CMS detector - CERN Document Server*, Retrieved 08 October 2018. [Online]. Available: https://cds.cern.ch/record/2205172.

[13] *Performance of the CMS Tracker Optical Links and Future Upgrade Using Bandwidth Efficient Digital Modulation*, Retrieved 08 October 2018. [Online]. Available: https://www.researchgate.net/figure/Cross-sectional-view-of-the-CMS-detector-11-The-HCAL-contains-and-measures-the_fig3_41217449.

[14] N. Parashar, "CMS Pixel Detector Upgrade", Oct. 2011. arXiv: 1110.2125.

[15] P. Adzic, R. Alemany-Fernandez, *et al.*, "Energy resolution of the barrel of the CMS Electromagnetic Calorimeter", *Journal of Instrumentation*, 2007. DOI: 10.1088/1748-0221/2/04/P04004.

[16] CMS Collaboration, *The CMS hadron calorimeter project - CERN Document Server*, 1997.

[17] V. I. Klyukhin, A. Ball, *et al.*, "Measurement of the CMS magnetic field", in *IEEE Transactions on Applied Superconductivity*, vol. 18, Oct. 2008, pp. 395–398. DOI: 10.1109/TASC.2008.921242. [Online]. Available: http://arxiv.org/abs/1110.0306%20http://dx.doi.org/10.1109/TASC.2008.921242.

[18] U. Bhawandeep, V. Khachatryan, *et al.*, "The CMS trigger system", *Journal of Instrumentation*, 2017. DOI: 10.1088/1748-0221/12/01/P01020.

[19] S. Chatrchyan, V. Khachatryan, *et al.*, "Description and performance of track and primary-vertex reconstruction with the CMS tracker", *Journal of Instrumentation*, 2014. DOI: 10.1088/1748-0221/9/10/P10009.

[20] S. Chatrchyan, V. Khachatryan, *et al.*, "Performance of CMS muon reconstruction in pp collision events at sqrt(s) = 7TeV", *Journal of Instrumentation*, 2012. DOI: 10.1088/1748-0221/7/10/P10002.

[21] S. Chatrchyan, V. Khachatryan, *et al.*, "Performance of CMS muon reconstruction in cosmic-ray events", *Journal of Instrumentation*, 2010. DOI: 10.1088/1748-0221/5/03/T03022.

[22] M. Cacciari, G. P. Salam, and G. Soyez, "The anti- k t jet clustering algorithm", *Journal of High Energy Physics*, 2008. DOI: 10.1088/1126-6708/2008/04/063.

[23] S. Chatrchyan, V. Khachatryan, *et al.*, "Identification of b-quark jets with the CMS experiment", *Journal of Instrumentation*, 2013. DOI: 10.1088/1748-0221/8/04/P04013.

[24] M. A. Dobbs, S. Frixione, *et al.*, "Les Houches Guidebook to Monte Carlo Generators for Hadron Collider Physics", Mar. 2004. arXiv: 0403045 [hep-ph].

[25] S. Abdullin, P. Azzi, *et al.*, "The Fast Simulation of the CMS Detector at LHC", *Journal of Physics: Conference Series*, vol. 331, no. 3, p. 032 049, Dec. 2011. DOI: 10.1088/1742-6596/331/3/032049.

[26] The Pierre Auger Collaboration, A. Aab, *et al.*, "Indication of anisotropy in arrival directions of ultra-high-energy cosmic rays through comparison to the flux pattern of extragalactic gamma-ray sources", Jan. 2018. DOI: 10.3847/2041-8213/aaa66d.

[27] A. Letessier-Selvon and T. Stanev, "Ultrahigh energy cosmic rays", *Reviews of Modern Physics*, 2011. DOI: 10.1103/RevModPhys.83.907.

[28] K. Greisen, "End to the cosmic-ray spectrum?", *Physical Review Letters*, vol. 16, no. 17, pp. 748–750, Apr. 1966. DOI: 10.1103/PhysRevLett.16.748.

[29] G. T. Zatsepin and V. A. Kuz'min, "Upper Limit of the Spectrum of Cosmic Rays", *Journal of Experimental and Theoretical Physics Letters*, vol. 4, p. 78, 1966.

[30] D. Kümpel, "Multivariate Search for a Directional Excess of EeV Photons with the Pierre Auger Observatory", PhD thesis, Bergische Universität Wuppertal, 2011.

[31] M. Opher, C. Prested, D. J. McComas, N. A. Schwadron, and J. F. Drake, "Probing the nature of the heliosheath with the neutral atom spectra measured by IBEX in the voyager 1 direction", *The Astrophysical Journal*, vol. 776, no. 2, p. L32, Oct. 2013. DOI: 10.1088/2041-8205/776/2/L32.

[32] R. Beck, "Magnetic Fields in Spiral Galaxies", Sep. 2015. DOI: 10.1007/s00159-015-0084-4.

[33] M. Prouza and R. Smida, "The Galactic magnetic field and propagation of ultra-high energy cosmic rays", Jul. 2003. DOI: 10.1051/0004-6361:20031281.

[34] R. Jansson and G. R. Farrar, "A new model of the galactic magnetic field", *Astrophysical Journal*, vol. 757, no. 1, Apr. 2012. DOI: 10.1088/0004-637X/757/1/14.

[35] M. S. Pshirkov, P. G. Tinyakov, P. P. Kronberg, and K. J. Newton-Mcgee, "Deriving the global structure of the galactic magnetic field from faraday rotation measures of extragalactic sources", *Astrophysical Journal*, vol. 738, no. 2, Mar. 2011. DOI: 10.1088/0004-637X/738/2/192.

[36] G. R. Farrar, N. Awal, D. Khurana, and M. Sutherland, "The galactic magnetic field and UHECR optics", *Proceedings of Science*, vol. 30-July-20, Aug. 2015. DOI: 10.22323/1.236.0560.

[37] B. Keilhauer, "Investigation of Atmospheric Effects on the Development of Extensive Air Showers and their Detection with the Pierre Auger Observatory", PhD thesis, Forschungszentrum Karlsruhe, 2004.

[38] J. Matthews, "A Heitler model of extensive air showers", *Astroparticle Physics*, vol. 22, no. 5-6, pp. 387–397, 2005. DOI: 10.1016/j.astropartphys.2004.09.003.

[39] The Pierre Auger Collaboration, A. Aab, *et al.*, *The Pierre Auger Observatory: Contributions to the 35th International Cosmic Ray Conference (ICRC 2017)*, 2017. arXiv: 1708.06592. [Online]. Available: http://arxiv.org/abs/1708.06592 (visited on 10/11/2018).

[40] J. Abraham, P. Abreu, *et al.*, "The fluorescence detector of the Pierre Auger Observatory", *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 620, no. 2-3, pp. 227–251, Aug. 2010. DOI: 10.1016/j.nima.2010.04.023.

[41] *A Gentle Introduction to Artificial Neural Networks*, Retrieved 29 September 2018, Sep. 2014. [Online]. Available: https://theclevermachine.wordpress.com/2014/09/11/a-gentle-introduction-to-artificial-neural-networks/.

[42] *Machine learning fundamentals (II): Neural networks*, Retrieved 06 October 2018, Dec. 2017. [Online]. Available: https://towardsdatascience.com/machine-learning-fundamentals-ii-neural-networks-f1e7b2cb3eef.

[43] Y. LeCun, B. Boser, *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Computation*, 1989. DOI: 10.1162/neco.1989.1.4.541.

[44] K. Hornik, "Approximation capabilities of multilayer feedforward networks", *Neural Networks*, vol. 4, no. 2, pp. 251–257, Jan. 1991. DOI: 10.1016/0893-6080(91)90009-T.

[45] *Back Propagation in Convolutional Neural Networks - Intuition and Code*, Retrieved 01 October 2018, Dec. 2017. [Online]. Available: https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199.

[46] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks", Tech. Rep., 2011. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification", Feb. 2015. arXiv: 1502.01852.

[48] D. P. Kingma and J. L. Ba, "Adam: a Method for Stochastic Optimization", *International Conference on Learning Representations 2015*, 2015. DOI: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503.

[49] *Overfitting, Wikipedia*, Retrieved 03 October 2018, Aug. 2018. [Online]. Available: https://en.wikipedia.org/wiki/Overfitting.

[50] T. Head, MechCoder, *et al.*, "Scikit-optimize/scikit-optimize: V0.5.2", Mar. 2018. DOI: 10.5281/ZENODO.1207017.

[51] I. Goodfellow, J. Pouget-Abadie, and M. Mirza, "Generative Adversarial Networks", *arXiv preprint arXiv: ...*, 2014. DOI: 10.1017/CBO9781139058452.

[52] *A Beginner's Guide to Generative Adversarial Networks*, Retrieved 03 October 2018. [Online]. Available: https://skymind.ai/wiki/generative-adversarial-network-gan.

[53] M. Abadi, A. Agarwal, *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", *None*, 2015. DOI: 10.1038/nn.3331.

[54] E. Bernhardsson, E. Freider, *et al.*, *Luigi*, 2012. [Online]. Available: https://github.com/spotify/luigi.

[55] M. Rieger, *Law - Luigi Analysis Workflow*, 2017. [Online]. Available: https://github.com/riga/order.

[56] N. Krammer and D. Liko, "Computing challenges of the CMS experiment", *Journal of Instrumentation*, vol. 12, no. 06, pp. C06039–C06039, Jun. 2017. DOI: 10.1088/1748-0221/12/06/C06039.

[57] M. Paganini, L. De Oliveira, and B. Nachman, "CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks", *Physical Review D*, 2018. DOI: 10.1103/PhysRevD.97.014021.

[58] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Network", in *International Conference on Machine Learning*, 2017. DOI: 10.2507/daaam.scibook.2010.27.

[59] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs", Mar. 2017. arXiv: 1704.00028.

[60] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On Convergence and Stability of GANs", May 2017. arXiv: 1705.07215.

[61] Fisher, "On the probable error of a coefficient of correlation deduced from a small sample", *Metron*, vol. 1, 1921.

[62] M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt, "Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks", *Computing and Software for Big Science*, vol. 2, no. 1, p. 4, Nov. 2018. DOI: 10.1007/s41781-018-0008-x.

[63] S. A. Zeid, J. Andrea, *et al.*, "Performance of b-tagging algorithms at 13 TeV", *CMS Analysis Note 2016/036, CMS*, 2016.

[64] M. Erdmann, L. Geiger, D. Schmidt, M. Urban, and M. Wirtz, "Origins of Extragalactic Cosmic Ray Nuclei by Contracting Alignment Patterns induced in the Galactic Magnetic Field", *Submitted to Astroparticle Physics*, Jul. 2018. arXiv: arXiv:1807.08734v1.

[65] M. De Domenico, M. Settimo, S. Riggi, and E. Bertin, "Reinterpreting the development of extensive air showers initiated by nuclei and photons", *Journal of Cosmology and Astroparticle Physics*, vol. 2013, no. 7, 2013. DOI: 10.1088/1475-7516/2013/07/050.

[66] D. E. Alexandreas, D. Berley, *et al.*, "Point source search techniques in ultra high energy gamma ray astronomy", *Nuclear Inst. and Methods in Physics Research, A*, vol. 328, no. 3, pp. 570–577, May 1993. DOI: 10.1016/0168-9002(93)90677-A.

[67] J. Abraham, P. Abreu, *et al.*, "Measurement of the energy spectrum of cosmic rays above 1018eV using the Pierre Auger Observatory", *Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics*, vol. 685, no. 4-5, pp. 239–246, Mar. 2010. DOI: 10.1016/j.physletb.2010.02.013.

Acronyms

AERA auger engineering radio array. 20

ALICE A Large Ion Collider Experiment. 9

ANN Artificial Neural Network. 23, 24, 26, 28

ATLAS A Toroidal LHC ApparatuS. 9

BDT Boosted Decision Tree. 23

CDF cumulative distribution function. 44, 45

CERN European Organization for Nuclear Research. 8

CMB cosmic microwave background. 15, 16

CMS Compact Muon Solenoid. 3, 8–11, 37

CNN Convolutional Neural Network. 23, 25, 37

CR cosmic ray. 15–17, 20, 67, 69, 70, 72–81

CSV combined secondary vertex. 12, 13

DL Deep Learning. 3, 23, 25, 26, 28, 31, 33, 37, 38, 48, 53, 54, 64, 67, 68, 71, 83, 84

DNN Deep Neural Network. 25–27, 29, 30, 32–35, 38, 39, 51, 54, 64, 67–69, 78, 81, 83, 84

EAS extensive air shower. 17, 19

ECAL electromagnetic calorimeter. 9, 10

FD fluorescence detector. 20, 21, 71

GAN Generative Adversarial Network. I, 33, 34, 37, 38, 40, 42, 44, 53, 54

GPU graphics processing unit. 23, 24

GRB gamma-ray burst. 14

HCAL hadronic calorimeter. 10

HEAT high elevation auger telescopes. 20

HL-LHC High-Luminosity Large Hadron Collider (LHC). 37

LEP Large Electron-Positron Collider. 8

LHC Large Hadron Collider. 3, 8, 9, 37, H

LHCb Large Hadron Collider beauty. 9

LINAC2 Linear Accelerator 2. 9

MC Monte Carlo. 13, 37, 38, 47, 49–54, 56, 57, 59–61, 63, 65

ML Machine Learning. 23, 31, 35, 38, 48, 53

MSE mean squared error. 26

NbTi Niobium Titanium. 10

$PbWO_4$ lead tungstate. 9, 10

PMT photomultiplier tube. 10, 20

PS Proton Synchrotron. 9

PSB Proton Synchrotron Booster. 9

QCD Quantum Chromodynamics. 5, 6

ReLU Rectified Linear Unit. 23, 28, 29, 43, 58

SD surface detector. 19–21

SF scale factor. 53–64, A

SM Standard Model of particle physics. 3–5, 13, 51

SPS Super Proton Synchrotron. 9

UHECR ultra-high-energy cosmic ray. 13–17, 19, 67, 68, 70

WGAN Wasserstein-GAN. 37–42, 45–48, 50–53, 64, 65

Danksagung

An dieser Stelle möchte mich für das schöne und spannende Jahr in der Arbeitsgruppe bedanken!

Ein besonderer Dank gilt Prof. Dr. Martin Erdmann für die intensive Betreuung und im Besonderen für die aufregende Möglichkeit bei der American Week nicht nur ein bisschen besser das Pierre-Auger-Experiment kennenlernen zu dürfen, sondern auch an einem spannenden Paper beteiligt zu sein.

Als stete Ansprechpartner dienten mir auch Yannik Rath und Marcel Rieger, vielen Dank dafür! Bei Dennis Noll und Benjamin Fischer bedanke ich mich für die entspannte und inspirierende Arbeitsatmosphäre in unserem Büro. Für das intensive Korrekturlesen danke ich Yannik Rath, Marcus Wirtz und Dennis Noll.

Nicht zuletzt möchte ich mich bei meiner Freundin für die Unterstützung und Motivation besonders in der Phase des Aufschreibens dieser Arbeit bedanken.