Friedrich-Alexander-Universität
Faculty of Sciences

Bachelor's thesis

11-11-2025

Julian van Laak

Investigating the agreement of Monte-Carlo simulations with KM3NeT data using Deep Learning techniques

Abstract

Neutrinos are elementary particles described in the framework of the Standard Model of Particle Physics. There are open questions and unknown aspects regarding the neutrino and its astrophysical sources which is why great effort is invested to resolve them. The KM3NeT/ARCA detector is a deep Sea neutrino telescope in the Mediterranean sea under construction with the aim of finding and observing high-energy neutrino sources. With this detector, light from the products of neutrino interactions can be detected and properties of the respective neutrino can be reconstructed. Monte Carlo simulations are created which can be compared to data and with this comparison the theoretical understanding of neutrinos and the processes leading to their detection can be verified. The cause of discrepancies between data and simulations can be investigated to resolve problems in the issues in the understanding and modeling of the detection. To investigate discrepancies between data and simulations a workflow has been established to train a Transformer model on the task of distinguishing whether an event originates from data or from muon simulations. Two models have been created. The first one consideres all signals which were assigned to an event and the second one consideres only the first signal on each PMT in an event to remove the possibility of after-pulses being the cause of the distinction since they are not simulated. Both models could successfully distinguish new sets of events, which have not been used in the training procedure, with high accuracies of 98.5% and 99.9%. It has been identified which features have the greatest influence on the models' decisions, namely the x- and y-position of the PMTs, and the so-called time over treshold, a proxy for the charge collected by a PMT. Lastly, the distributions of features for the signals from data events and from muon simulation events have been compared to find directly visible causes. It has been found that the x-position feature contains signals with values that are only present in data. This fraction makes up 25% of the data signals. It has been concluded that this characteristic is probably used by the models to distinguish events but there are hints for other, additional causes which are not so directly visible.

Zusammenfassung

Neutrinos sind elementare Teilchen, die mithilfe des Standardmodells der Teilchenphysik beschrieben werden können. Es gibt offene Fragen und unbekannte Aspekte in Bezug auf Neutrinos und ihre astrophyikalischen Quellen, weshalb große Anstrengungen unternommen werden, um diese zu klären. Der KM3NeT/ARCA Detektor ist ein Tiefsee-Neutrino-Teleskop im Mittelmeer, welches sich im Aufbau befindet, mit dem Ziel hochenergetische Neutrino Quellen zu finden und zu beobachten. Mit diesem Detektor kann Licht von den Produkten von Neutrino Interaktionen detektiert werden und Eigenschaften des jeweiligen Neutrinos rekonstruiert werden. Monte Carlo Simulationen werden programmiert, die mit den Daten verglichen werden können und mit diesem Vergleich kann das theoretische Verständnis der Neutrinos und den Prozessen die zu ihrer Detektion führen verifiziert werden. Die Ursache für Unterschiede zwischen Daten und Simulationen kann untersucht werden, um Probleme des Verständnisses und der Modellierung der Detektionsprozesse zu lösen. Um diese Diskrepanzen zu untersuchen wurde ein Arbeitsablauf eingeführt, der es ermöglicht einen Transformer für die Aufgabe, zwischen Events aus Daten und Events aus Myon Simulationen zu unterscheiden, zu trainieren. Zwei Modelle wurden erstellt. Das erste berücksichtigt alle Signale eines Events und das zweite berücksichtigt nur das erste Signal von einem PMT in einem Event, um die Möglichkeit auszuschließen, dass after-pulses die Ursache für die Unterscheidung sind, weil diese nicht simuliert werden. Beide Modelle konnten erfolgreich neue Events unterscheiden, welche nicht im Trainingsprozess benutzt wurden, mit hohen Genauigkeiten von 98.5% und 99.9%. Es wurde identifiziert, welche Features den größten Einfluss auf die Entscheidungen der Modelle haben, nämlich x- und y-Position der PMTs und die sogenannte time over threshold, die stellvertrend für die Ladung ist, die von einem PMT gesammelt wurde. Zuletzt wurden die Verteilungen der Features für die Signale der realen Daten Events und der der Simulationen verglichen, um direkt sichtbare Gründe zu finden. Es wurde gefunden, dass das x-Position Feature Signale beinhaltet, mit Werten, die nur in Daten vorkommen. Dieser Anteil beträgt 25% aller realen Daten Signale. Es wurde festgestellt, dass diese Charakteristik von den Modellen benutzt wird, aber es Hinweise auf andere, zusätzliche Ursachen gibt, die nicht direkt so sichtbar sind.

Contents

1 Neutrino physics

1.1 Fundamental properties

The neutrino was postulated in 1930 as an electrically neutral particle with spin $\frac{1}{2}$ while it was detected in 1956 for the first time. Today, its description is implemented in the Standard Model of Particle Physics. This model contains 12 fundamental particles with spin $\frac{1}{2}$ - the leptons and the quarks - all grouped in different families. Additionally, there are five bosons with integer-valued spins. They mediate the different forces like the strong force, the electromagnetic force, and the weak force. Neutrinos lack electrical and colour charge. Therefore, they can only interact via the weak interaction and gravity, although the latter can be neglected due to the small value of the neutrino mass. This has the effect that neutrinos interact very rarely, expressed by the cross section which is of the order of $\frac{\sigma}{E} \approx 10^{-42} \frac{\mathrm{m}^2}{\mathrm{GeV}}$. Consequently, there are two interactions that can occur: the charged current (CC) interactions and the neutral current (NC) interactions, both shown in figure 1. In the NC interactions the Z^0 boson is the force mediating

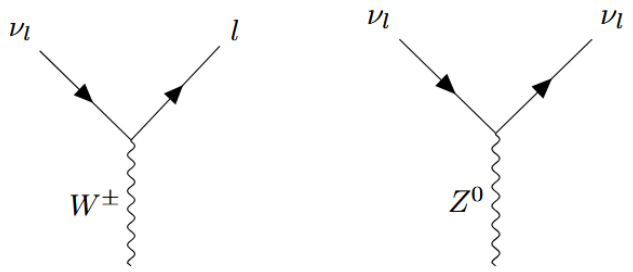


Figure 1: Feynman diagrams of CC and NC neutrino interaction vertices; $l \in \{e, \mu, \tau\}$. From [1]

particle and the outgoing particle is the neutrino with the same flavour. In the CC interaction the $\mathrm{W}^{\pm}$ bosons are exchanged and the outgoing particle is a lepton with the same flavour as the incoming neutrino. This is due to the conservation of lepton number and forbidden cross-generation couplings [1] [2]. The reason for the relatively small cross section of weak interactions - depending on the energy regime - are the large masses of the force mediating particles $W^{\pm}$ and Z^0 in contrast to the massless photon and gluon which are the mediators of the other two fundamental interactions in the Standard Model [3] [4].

1.2 Why neutrinos?

Even today there are still aspects which are unknown regarding the neutrino. Earlier expectations assumed neutrinos to be massless which is still implemented in the Standard Model. Nevertheless, the observation of neutrino oscillations provided an experimental evidence for the neutrino mass being larger than zero [3] [5]. The formalism of neutrino oscillations defines the neutrino flavour eigenstates $\nu_{e,\mu,\tau}$ as mixtures of mass eigenstates $\nu_{1,2,3}$. The relationship between the flavour eigenstates and the mass eigenstates is given by a 3×3 mixing matrix [3] [6]. So far there are only upper limits for the masses and no direct measurements, e.g. for the lightest neutrino mass the upper limit is a mass of 0.45 eV [2]. The thought-provoking aspect is the size of the mass: it is several

orders of magnitude smaller than other fermion masses in the Standard Model [3]. Due to the dependence of observable parameters like the oscillation probability solely on the squared mass differences, there is the open problem of the neutrino mass hierarchy which describes the possible arrangements of the masses depending on their values resulting in two possibilities: the normal and the inverted hierarchy [4]. Additionally there is the open question if neutrino and antineutrino are different particles or if they are just two different helicity states of the same particle where helicity is defined as the projection of spin onto the direction of motion [5] [2]. In conclusion there are certainly several features of the neutrino which make it an interesting particle to investigate and to search for physics which could help resolve some of the big open questions concerning the Standard Model.
The answers to these questions on the properties of the neutrinos inevitably play a role in astrophysical processes in which neutrinos participate. By investigating them, progress can not only be made in the field of fundamental particle physics but also in astrophysics by finding new sources of neutrinos and deepening the understanding of astrophysical topics, e.g. cosmic ray accelerators like supernovae and the early universe [7].

1.3 Neutrino sources

Neutrinos reaching the Earth originate from many different sources. A few origins will be explained in the following.
The best known source is the Sun. Due to the large number of fusion reactions occurring in the interior of the Sun, which are mainly the pp chain and the CNO cycle, numerous solar neutrinos are emitted and may reach the Earth [8].
An important type of neutrinos to consider are the geoneutrinos or also known as terrestial neutrinos. They are produced by radioactive decays of mainly ^{238}U, ^{232}Th and ^{40}K which occur naturally in the Earth interior and consist only of $\overset{(-)}{\nu}_e$. Due to their natural presence even in isolated places, that are protected from other disturbing particles, they play an important role in the detection process [1] [8].
Cosmic rays continuously hit Earth's atmosphere leading to interactions with the particles there and as a consequence secondary particles like mesons and muons are created. The charged component of cosmic rays is composed of nearly exclusively hadrons and a small part of electrons. Mesons like the positive or negative charged pions and kaons can decay into $\mu^{\pm}$ and $\overset{(-)}{\nu}_\mu$ respectively. The muon may then additionally decay into electron and anti electron neutrino where the same process just with particle and antiparticle swapped happens for the anti muon. The whole process can be written as

$$\pi^{\pm}, K^{\pm} \rightarrow \overset{(-)}{\nu}_\mu \, \mu^{\pm} \rightarrow \overset{(-)}{\nu}_\mu \, e^{\pm} \overset{(-)}{\nu}_e \overset{(-)}{\nu}_\mu \tag{1}$$

in a compact form while the lepton number and especially the family lepton numbers have to be conserved. These neutrinos are called atmospheric neutrinos while the muons are called atmospheric muons. Atmospheric neutrinos can reach detectors from above but additionally also from below travelling a long distance through the inner structure of Earth. By measuring them neutrino oscillation phenomena can be investigated [6] [8].
If a supernova occurs, which is either the thermonuclear explosion of white dwarfs in binary systems due to their mass limit or the core collapse of massive stars with masses larger than eight solar masses, a large amount of neutrinos is emitted. Roughly 99% of

the emitted energy in such a explosion is released via neutrinos [6].
Additionally, there are several other possible neutrino sources in space. Cosmological neutrinos whose creation would be traceable to a time period shortly after the Big Bang have energies in the μeV to meV range which is the reason why there is at the moment no direct method to detect them. When high-energetic cosmic rays interact with photons of the cosmic microwave background the so called cosmogenic neutrinos are created and their energies exceed PeV [1]. Furthermore astrophysical neutrinos can stem from sources like supernova remnants, Active Galactic Nuclei, or Gamma Ray Bursts. There, in the vicinity of these sources interactions of protons or nuclei with photons or gas occur which can lead to the production of neutrinos. These sources often not only produce neutrinos but also cosmic rays. The advantage of neutrinos is that they are not influenced by magnetic fields or other interactions which could change their travel path which is the case for cosmic rays [8].

1.4 Neutrino-matter interactions

There is more to the neutrino interaction than just one vertex from figure 1 but also a second vertex connected via the exchanged boson. The complete process occurs as interaction between the neutrino and a second particle where between them one of the $W^{\pm}$ and Z^0 bosons are exchanged - at least at the tree level of the process. A four-momentum exchange q occurs between the neutrino and the target particle which can be calculated as difference between incoming and outgoing four momentum. In principle, there are several possible target particles which could be taking part in such an interaction. When looking at neutrinos travelling in and around Earth, the main target particles are nucleons which consist of partons, and electrons while the cross sections of these interactions are dependent on the energy regime. For neutrino energies below 2 GeV the main contribution can be viewed as the scattering between neutrino and the nucleon as a whole whereas for higher energies starting at $5 - 10$ GeV the neutrino most dominantly interacts with the partons inside the nucleon [9]. For this thesis the ultra high energy regime in the TeV-PeV range is of particular interest and the reason for this will be explained in section 2.1. This so called deep inelastic scattering still being one of the main contributers in this energy regime can be written as follows:

$$\overset{(-)}{\nu}_l N \rightarrow l^{\pm} H \tag{2}$$

for the CC interaction and

$$\overset{(-)}{\nu}_l N \rightarrow \overset{(-)}{\nu}_l H \tag{3}$$

for the NC interaction where H is the hadronic final state [1] [9] [10]. Independently of which of the weak bosons is exchanged in the interaction, a hadronic shower of particles is always produced. Additionally, for higher energies the scattering on electrons gets more important and their cross sections are only a few orders of magnitude smaller than the ones of the nucleon interactions [9]. These interactions can be described by

$$\overset{(-)}{\nu}_l e^- \rightarrow \overset{(-)}{\nu}_l e^- \,. \tag{4}$$

For a small energy interval the cross sections of the processes $\bar{\nu}_e e^- \rightarrow \bar{\nu}_\mu \mu^-$ and $\bar{\nu}_e e^- \rightarrow \bar{\nu}_e e^-$ even surpass the cross sections of the nucleon interactions due to the creation of the W boson which is called the Glashow resonance [10] [9].

2 KM3NeT

2.1 Detector

The KM3NeT Collaboration is a research collaboration which currently builds two deep sea neutrino telescopes in the Mediterranean sea. Two different scientific goals are pursued which can be tackled with the two detectors respectively. These are finding and observing high-energy neutrino sources in the Universe and determining the neutrino mass hierarchy. The two detectors are named ARCA which is an abbreviation for Astroparticle Research with Cosmics in the Abyss and ORCA which stands for Oscillation Research with Cosmics in the Abyss. The location of ARCA is 100 km offshore from Porto Palo di Capo Passero in Sicily at a depth of 3500 m and ORCA's location is 40 km offshore from Toulon at a depth of 2450 m. These detectors consist of so called building blocks while two belong to ARCA and one belongs to ORCA. To each building block belong 115 Detection Units (DUs) which are a construct of two long ropes attached to the bottom of the sea and 18 Digital Optical Modules (DOMs) being installed between them. DOMs are the centrepieces of the detectors. They are glass spheres with a diameter of 43 cm and contain 31 photomultiplier tubes (PMTs) which will be explained in section 2.4 in more detail [1] [11]. A Detection Unit and a DOM are shown in figure 2. With them it is possible to detect the Cherenkov radiation emitted by

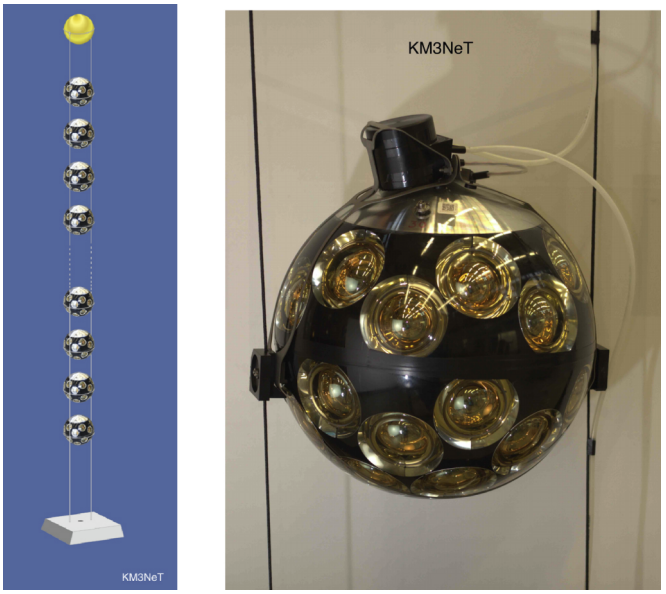


Figure 2: Detection Unit (left) and Digital Optical Module with cables and fixation (right). From [11]

products of neutrino interactions [11]. ARCA and ORCA differ in the spacing between DOMs and DUs due to their different areas of application. For ARCA the distance between DUs is 90 m and the distance between DOMs is 36 m, whereas for the more compact ORCA detector the DU distance is 20 m and the DOM distance is 9 m [9]. Additionally, the two detectors work best in different energy regimes, namely energies from GeV to a 100 GeV for ORCA and TeV to PeV for ARCA [9]. For this thesis ARCA data is used which is the reason why the review of energy dependent phenomena

is often limited to the TeV to PeV energy range. The detectors are currently still under construction which means that not all 230 or 115 DUs respectively are installed. At the moment there are about 50 DUs in ARCA installed and functional but the data used here is from a time period where 21 DUs were operational [12]. The instrumented volume of one ARCA block with all planned strings installed amounts to $0.48\,\mathrm{km}^3$ resulting in nearly $1\,\mathrm{km}^3$ for the two planned ARCA blocks [11]. The location and the sheer size of the detector give rise to several advantages to make good measurements. Due to the size many nucleons are available for neutrino interactions as described in section 1.4 increasing the probability of detectable interactions despite their small cross section and the large amount of water above and around the detector acts as a shielding against unwanted particles [13].

2.2 Detection techniques

2.2.1 Cherenkov radiation

Neutrino interactions produce particles which are often relativistic and may have a charge. For the detection of such events the Cherenkov radiation of these relativistic and charged particles can be utilised. This type of radiation is produced when a particle of the mentioned kind moves in a dielectric medium with speeds higher than the local phase velocity of light. This phenomenon is due to the created polarisation which is asymmetric along the trajectory. The angle under which the light is emitted is called the Cherenkov angle and is given by

$$\cos\theta_{\mathrm{C}} = \frac{1}{\beta n} \tag{5}$$

with n being the index of refraction of the corresponding medium and $\beta = \frac{v}{c}$ where v is the speed of the particle and c the speed of light in vacuum [1]. A qualitative explanation of this asymmetry is the interference of the emitted waves. As response to the moving, charged particle spherical waves are emitted moving away from the particle trajectory. For particle speeds smaller than the local speed of light there occurs no preferred constructive interference. For particle speeds exceeding the local speed of light the situation becomes different. There, parts interfere constructively and form a type of shock wave which has the shape of a cone with θ_{C} being the angle between particle track and the normal of the shock front [14]. For the case of using deep sea water as the medium which has $n_{\mathrm{water}} = 1.35$ the Cherenkov angle is $\theta_{\mathrm{C}} = 42^\circ$ for ultra-relativistic particles [1]. The choice of deep sea water is motivated by the type of the detector mentioned before which collects the data used in this thesis.
Mandatory for the emission of this radiation is the electrical charge of the particle being different from zero. As stated before neutrinos do not have a charge but fortunately in the CC interaction a charged lepton is emitted. If the energy of this particle is high enough it will emit Cherenkov light which in turn can be detected. The threshold kinetic energy at which emission is produced is given by

$$T_{\mathrm{thresh}} = \left(\frac{1}{\sqrt{1-n^{-2}}} - 1\right) mc^2 \tag{6}$$

where m is the mass of the particle. For sea water the prefactor is approximately equal to 0.49 [1].

The photons travelling in the sea water can in general be absorbed or scattered which leads to a decrease in the number of detectable photons. The major part of Cherenkov photons gets emitted with wavelengths of about $300 - 500\,\text{nm}$. Fortunately, the absorption of water is minimal at approximately these wavelengths, resulting in good conditions for detecting Cherenkov photons over distances [15]. For scattering the process can be divided into scattering on water molecules and scattering on larger particles, like small dust grains called Mie scattering. In the important wavelength range Mie scattering dominates which has the consequence that mostly scattering angles are small. The scattering length is approximately in the same value range as the absorption length in this wavelength region, but by introducing the effective scattering length which considers also the scattering angle one receives scattering lengths with an order of magnitude higher [1] [16]. Thus, while scattering and absorption are definitely not negligible effects in water, it is still well suited for measuring Cherenkov light.

2.2.2 Event signatures

By considering the products of the different neutrino interactions from section 1.4 one can draw conclusions on the type of the interaction and on the flavour of the corresponding neutrino. One way is to measure the Cherenkov radiation of the created particles where different emission patterns correspond to different events. In this case the particles which may emit Cherenkov radiation are the hadrons from the hadronic showers as well as other particles which stem from further interactions of the hadrons and the leptons in the CC interactions [9]. The different event signatures of neutrinos interacting with a nucleon are shown in figure 3. They are the dominant processes

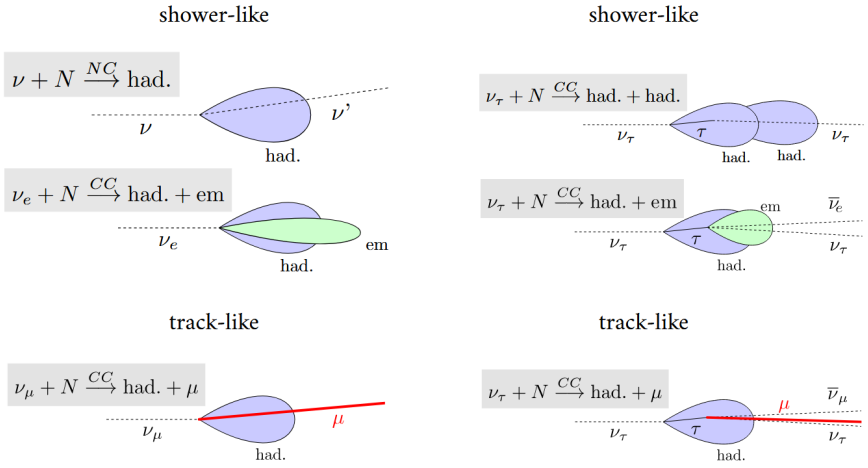


Figure 3: Schematics of different neutrino event topologies for interactions of neutrinos with nucleons. From [1]

in the for this thesis important energy regime except for the small energy interval corresponding to the Glashow resonance which was mentioned in section 1.4. One distinguishes between track-like which is characterised by the Cherenkov light emission from a single particle like a muon or tauon taking a straight path and shower-like characterised by a sequence of particle decays where the produced particles all may emit Cherenkov radiation depending on their energy. The fundamental example for the

shower-like signature is the NC interaction in figure 3 where only a hadronic shower is produced while the neutrino leaves the scene unrevealed. On the other hand, the primary example for the track-like signature is the NC interaction of a $\overset{(-)}{\nu}_\mu$ with a nucleon. Even though there is still a hadronic shower produced the created muon travels several meters depending on its energy through the medium. The other processes in figure 3 seem more complex. The electron (anti) neutrino interaction additionally entails an electromagnetic shower which consists of $e^\pm$ and photons. This is a consequence of the photons emitted via bremsstrahlung which can perform pair production keeping the recurring process running. In $\overset{(-)}{\nu}_\tau$ CC interactions the short lifetime of the produced $\tau^\pm$ - leading only to a short track - and its decay channels play a central role. The hadronic decay leads to a second hadronic shower whereas the decay in $\overset{(-)}{\nu}_e$ and $e^\pm$ leads to an electromagnetic shower. Additionally, the (anti) tauon can also decay into $\mu^\pm$ and neutrinos giving rise to a track signature [1] [9].

2.3 Background

2.3.1 Radioactivity, bioluminescent life and dark counts

The detectors are located deep in the Mediterranean sea which means there is no unwanted daylight hitting the sensors but still there are other sources of light occurring even in the depths of the sea. In the sea water a tiny fraction of ^{40}K naturally exists. As mentioned before ^{40}K is a radioactive isotope. It can decay via β-decay producing an electron among the decay products with possible energies exceeding the Cherenkov energy threshold. In the second decay channel electron capture occurs where an excited state of ^{40}Ar and an electron neutrino are produced. The excited state decays by emission of light which can in turn result in Compton scattering which may produce Cherenkov light emitting electrons as well [1] [9].
Another source of background light comes from bioluminescent living organisms which inhabit even the isolated environment where the neutrino telescopes are located. This type of light is called bioluminescence produced by chemical reactions [1].
Additionally, even if no photons are hitting the PMTs there still may occur a signal coming out of them. This is due to the small but non-zero chance of releases of electrons due to thermal excitation which are called dark counts [1] [9]. This is a fitting term because even if the photocathode is completely covered dark counts would still be measurable.

2.3.2 Atmospheric muons

In section 1.3 it was mentioned that atmospheric muons are produced as a result of cosmic ray interactions. Some muons do not decay and reach sea level. Factors enabling this phenomenon are their small energy loss while travelling through the atmosphere, a long lifetime and a small cross section. However, their journey does not necessarily end at sea level but they can travel up to $\sim 12\,\text{km}$ through water depending on their energy [13]. This results in a problem: the detection of neutrino interactions is partly based on measuring the Cherenkov light from muons created in these interactions but now it was established that there is the possibility of high energy muons with energies above 1 TeV travelling through the instrumented volume which are not coming from neutrino interactions but were rather produced in cosmic ray interactions. The rate

of these muons is by at least a factor of 10^4 larger than the rate of measured neutrino interactions. Atmospheric muons can only reach the detector from above because unlike neutrinos they can not travel through large parts of the Earth. Therefore taking advantage of this, a simple way of eliminating this contamination is by applying a cut on the reconstructed direction [1]. This means that only events with reconstructed directions which fulfil certain requirements are taken into account while the rest is ignored. The reconstructed direction and how it is obtained will be explained in section 2.5. Problems in the reconstruction algorithms can occur which may lead to interpreting atmospheric muons as upward travelling muons. Now it is difficult to assess if these events are misreconstructed or if they did originate from real events [1] [9].

2.4 From detection to data

The emitted Cherenkov light by a particle from a neutrino interaction as well as light from other light sources may be detected by the PMTs. These photons hit the cathode of the PMTs and - if they have enough energy - release electrons via the photoelectric effect. These electrons get accelerated through high voltages towards several dynodes where additional electrons get liberated at each dynode. Eventually an avalanche of electrons reaches the anode where the signal can be read out [17]. At this stage a threshold is used as a first filter. Only if an analogue signal from the PMT surpasses that threshold it is digitised and sent to shore. This digital version of the signal is called hit, and it contains the timestamp at which the leading edge of the signal crossed the threshold, the time that the signal remained over the threshold (ToT), and the PMT identifier. The hits from all the PMTs are transmitted to the shore station, where they get processed. This approach is motivated by the "all-data-to-shore" principle. The timestream of hits is sliced into time bins of 100 ms duration called time slices. They are processed by a computing algorithm called the Data Filter. The task of the Data Filter is to save sets of hits that contain information of interest, while discarding hits produced by background sources. Each set of physically-meaningful hits is saved as an event. This process is carried out in multiple stages described in [11], but the general concept is to find sets of causally connected hits compatible with the propagation of high energy particles through or nearby the detector [11] [9] [18].

2.5 Reconstruction

The information obtained from a neutrino event is only the ToT, arrival times and locations of the PMTs. The information that one needs in order to extract scientific conclusions from the data for the proposed goals is the direction of the neutrino which is responsible for the interaction and its energy. The closest one can get at the moment is determining the energy and direction of the products from the neutrino detection and additionally reconstructing these parameters for the neutrino itself [9]. This can be done by reconstructing these parameters where reconstructing means fitting a model to data while the important model here is the theoretical understanding of Cherenkov radiation emitted as a cone. Depending on the detector and event signature (track or shower) different reconstruction algorithms are used. For the reconstruction of muon tracks the following chain of algorithms is used

JMuonPrefit → JMuonSimplex → JMuonGandalf → JMuonStart → JMuonEnergy

while the workflow needs an event as an input. The first algorithm assumes many track directions, and based on these assumptions it draws conclusions on position and time of the muon for each hypothetical direction. The best suited directions determined by a fit quality parameter are saved and passed forward to the next stage [9] [19]. To improve speeds the JMuonSimplex algorithm is used which executes an intermediate direction fit. This is realised by comparing the arrival times of all hits from the Cherenkov photons at the detectors in a specific radius with the expected arrival times [9]. With the JMuonGandalf algorithm the final direction estimate is determined. It takes the previously selected fits and scans the surroundings. Here, probability density functions take several parameters into account like e.g. quantum efficiencies, angular acceptances, and light emission phenomena like radiative energy losses or emission by delta-rays, and with these functions it is possible to determine a track which best fits the signal [19]. A detailed description of this algorithm can be found in [19]. The JMuonStart algorithm determines the start point of the track by back-projecting the emitted light onto the newly reconstructed track. The last algorithm determines the reconstructed energy by minimizing a cost function which is dependent on this energy [9] [19]. A root file is produced by the last step in this algorithm chain [20]. Shower events are constructed differently and this process is more complicated [9]. Due to the minor role of shower events in this thesis the overview on shower reconstruction is not considered, but the concept of reconstructing important quantities from the obtained information is the same.

2.6 Simulations

A time period of data taking and the associated data is summarised with the term run. For every run corresponding Monte Carlo simulations are produced. There are two types of simulations: one type simulates neutrinos and their products in the detector resulting from interactions and the other type simulates atmospheric muons. Monte Carlo methods describe the process of producing a selected number of outputs following a chosen probability density function. By comparing these simulations to real data one receives feedback on the current understanding of the theoretical physics models [9]. The simulation chain is a chain of algorithms and its structure depends on which type of events to simulate similar as before. The concept of this chain - also often referred to as pipeline - can be summarised by several major steps: the generation of the particles and tracking their path, followed by the simulation of the emitted Cherenkov light, then applying the detector response to the Cherenkov light as well as trigger algorithms which are identical to the ones used for real data and lastly the reconstruction chain which was discussed before is applied, although with an additional algorithm included directly before the reconstruction chain [9] [19]. In the generation step neutrinos within a certain volume depending on the flavour and thus the event signature are created. This volume is distinguished by allowing the possibility of detection of created particles from neutrino interactions. These neutrinos all have specific energies which can be chosen. Now the question arises how many neutrinos should have which energy. The distribution is created by a energy spectrum of the form $\mathrm{d}N/\mathrm{d}E \propto E^{-\gamma}$ which does not necessarily match with real physical energy spectra. This is done to save computing resources because a physical energy spectrum has an excess of particles in the lower energy regimes and a shortness for high energies. By applying weights to every event one

can create a connection between the spectrum used for the simulation and the physical spectrum. This weight using approach is also used for the simulation of atmospheric muons [9].
Simulations are produced in the same data structure as the experimental data. This means that exactly like the experimental data there are a certain number of events in a run, which have many parameters assigned to them. As before, each event contains a certain number of hits. This same composition allows direct comparisons between data and simulations. For example, a specific parameter like the reconstructed energy can be compared between events of data and simulations or values from hits, such as the ToT, can be compared. Simulations are in particular useful because they reflect the theoretical understanding. They include the current understanding of the KM3NeT detectors, as well as the physical models influencing, for example, the interaction processes, the propagation and detection of signals, and backgrounds. This theoretical understanding can be tested by comparing the experimental data to the simulations. If deviations between them occur, an attempt can be made to find the cause. Therefore, with this method of comparing it is possible to improve the theoretical understanding [9].

3 Machine Learning

3.1 What is machine learning?

Due to the increasingly large amount of data in many fields, automating data analysis becomes more and more important. This automatisation is a crucial aspect of machine learning. To achieve this goal a machine learning model has the ability to detect patterns hidden in the data and based on them take decisions or predict future outcomes. The more popular form of machine learning, which is also used in this thesis, is called supervised learning. This can be summarised as the development of a model that provides a statistical approximation of a complicated function which may depend on a large number of parameters [21]. The model is applied to data and then returns predictions for specific features or properties which is a crucial aspect of this type of learning because the data on which the model gains its experience needs to be labelled. This means that the actual value for this quantity, which should be predicted by the model, needs to be known for the training data and needs to be compared to the model's prediction in order for it to learn in this approach. This requirement can sometimes lead to problems because in general data is not always labelled [22]. Fortunately, the data used in this thesis has an intrinsic property, namely whether it stems from a simulation or whether it is experimental data measured with the detector.
A first example to demonstrate the concept of machine learning is the classification of handwritten digits. One could think of a very complex function which takes the grey scale values of every pixel of the image containing the handwritten digit as an input and gives the correct number as an output. For humans recognising handwritten digits is most of the time quite simple while we also receive an input through our eyes and are able to give an output, but how should one tackle this task with a computer? Writing a program, which is static, fixed, and does not allow a flexible change in its structure and values based on the training inputs, seems not suited to achieve the desired goal. This fixed program can work on already classified images, but would have no useful applications for unknown datasets on which it has not been programmed, essentially predicting the already known. To get a real benefit the idea is to classify new and unknown datasets correctly. Here, the learning term comes into play. The approach is now to design a computer program, that is flexible and can change its structure and parameters based on its experience from previous datasets [21] [23] [24].
In summary, the model aims to estimate an unknown function f which fulfils

$$\boldsymbol{y} = f(\boldsymbol{x}) \tag{7}$$

where $\boldsymbol{x}$ is a vector of inputs and $\boldsymbol{y}$ a vector of outputs. For the previous example a possible output vector could have an entry for each number and the value of this entry gives a probability. The principle of training used in the supervised approach is comparing the outputs of the model, which were created by performing complex operations on the input parameters, with the real values one wants to reproduce. Based on this feedback the parameters of the model can be changed [21]. Complex patterns which have been learned in the training then can be used in new datasets.

3.2 Neural networks

The most important and best know class of machine learning models are the so called feedforward neural networks, also called multilayer perceptrons. In figure 4 a small neural network is shown as an example. The number of hidden layers in this example amounts to two. It is in principle in control of the coder and their computational resources how many hidden layers are used. The model consists of neurons and the

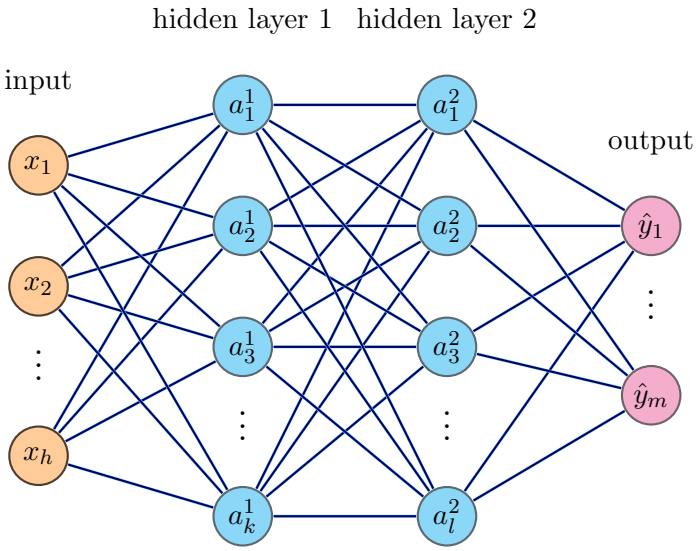


Figure 4: Feedforward neural network with two hidden layers. Adapted from [25] [26]

connections between them. The terms neuron and neural were established because this structure was influenced by structures in neuroscience. Essentially, each neuron is a function $f : \mathbb{R}^n \to \mathbb{R}$ giving a number as an result. The input layer is a special case where the values of the neurons are just the input values given to the model. A neuron in the hidden or output layer is a function of all the neurons in the previous layer - at least for a fully connected neural network. A weight w_i is assigned to each connection between two neurons and additionally a bias value b is added to every neuron which is not in the input layer. Altogether this gives the following calculation procedure for a neuron j which is connected to n neurons with values x_i from the previous layer:

$$z_j = \sum_{i=1}^{n} w_{ij} x_i + b_j \,. \tag{8}$$

This function is linear, and concatenations of linear functions are still linear. Arranged in this way the result would be a linear model regardless of the number of hidden layers. Thus, to approximate nonlinear functions the model needs nonlinear steps in its computations. This is realised by so called activation functions which are applied on every output from equation 8 individually. A common example of an activation function is the rectified linear unit abbreviated with ReLU and given by

$$f_{\mathrm{ReLU}}(x) = \max(0, x) \,. \tag{9}$$

Lastly, after computing the neurons going from layer to layer the output is given by the neurons in the output layer. Depending on the problem to be tackled, many neurons or

only a single neuron can be used. To simplify, the values of the neurons in each layer can be summarised as entries in a vector, and the corresponding weights can be written in a matrix.
In conclusion, the model acting as one big function is given by a sequential execution of other functions where each function represents a layer in the network. Now, the reason for naming the model "feedforward" becomes clear. The computations follow through the network layer after layer and output values are not given back to the model [23] [22] [27].
Firstly, the model needs to be trained on a set of data. For each set of inputs in the training data the desired outputs are known and can now be compared with the output of the model. By changing the weights and biases the output of the model can vary. This reflects the flexible structure of the model: the only defined instruction is that the output closely matches the desired values, but no direct instructions on how to choose the weights and biases is given [23].

3.3 Loss functions

The structure of the program is in need of a quantitative classification on how well the produced output $\hat{\boldsymbol{y}}$ matches the desired output $\boldsymbol{y}$. This is done by choosing a loss function $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$. Many different loss functions can be used depending on the chosen problem. An example is the well known squared error loss function which is given by

$$L_i = \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\|^2 \tag{10}$$

where the sizes of vectors $\hat{\boldsymbol{y}}_i$ and $\boldsymbol{y}_i$, which are the predicted and true labels from one single training example i, depend on the number of output neurons. In the training many training examples are used which results in the mean squared error loss function given by

$$L = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\|^2 . \tag{11}$$

In this thesis a classification problem is investigated and therefore another, better suited loss function is used [26]. It is called the Binary Cross Entropy loss and is given by

$$L_i = y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) . \tag{12}$$

for a single training example [28] [29]. Due to the fact that only one value as a prediction and true label will be used, the vector notation can be neglected.

3.4 Minimisation with Gradients

To train and improve the performance of the model, one has to minimize the loss function depending on the weights and biases of the model summarised by the vector $\boldsymbol{\theta} = (w_{11}^1, ..., w_{lm}^N, b_1^1, ..., b_m^N)$. This procedure is complicated due to the high dimensionality of the parameter space. Finding the global minimum is nearly impossible, thus the goal is to find a local minimum which is "good enough". To achieve this goal the gradient is used. It always points into the direction of steepest rise, thus the negative gradient points into the searched direction. The parameters will then be updated many times

by a small amount, eventually making small steps to a local minimum in the high dimensional space. The repeated change of the parameters can be written as

$$\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} - \eta \boldsymbol{\nabla}_{\theta} L \tag{13}$$

where η is the learning rate and L the loss function averaged over the whole dataset. The learning rate needs to be chosen manually and has a great effect on the training performance. If it is too low, it could lead to being stuck in a local minimum being worse than other local minima, which would otherwise be accessible. If it is too high, it could be difficult to gradually converge into the minimum [26] [27]. In practise the learning rate is often varied while training [23].
Computing equation 13 is very resource heavy. This can be solved by using Stochastic Gradient Descent (SGD). The big dataset is separated in many small subsets called batches which contain several, randomly chosen examples. Then, the average is calculated in this smaller subset and the parameters are updated [26]. A larger batchsize normally results in a better adjustment of the parameters but more computing resources are needed [27].
Another alternative will be used in this thesis which is called adaptive moment estimation (ADAM) which works similar to SGD. This method helps to accelerate the learning process by remembering previous gradient calculations and considering them in the calculation process. They are scaled by several parameters which exponentially decay resulting in a suppression for gradients which are several steps behind the current step [26].

3.5 Backpropagation

It is now clear that computing the gradient is necessary to achieve learning. Getting an analytical expression seems feasible but the numerical application would use many resources. The so called backpropagation algorithm is an algorithm which works very efficiently and therefore keeps the computational expenses low. The calculation is based on the chain rule from calculus. While it is directly possible to obtain an expression for the derivative of the cost function with respect to any parameter of the model by going backwards starting at the loss function and working a way through the model, looking at the repeating structure of the model gives a hint that some expressions reappear several times [23].
A simplified example [30], on which backpropagation can be demonstrated well, while also being applicable to general structures, is shown in figure 5. This neural network has in principle four layers but only the last two are of interest for this example. In the

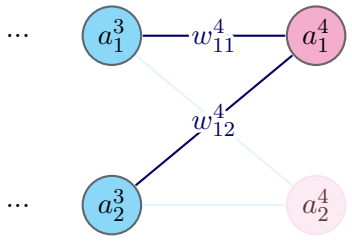


Figure 5: Last two layers of exemplary network with four layers. Adapted from [25] [30]

backpropagation computation only one training example and therefore the not averaged

loss function from equation 10 is considered. By using equation 8 the initial z_j^l are calculated, where l corresponds to the l-th layer in the network, and this is followed by applying an activation function to get a_j^l. To calculate the derivative with respect to w_{11}^4 as an example the chain rule needs to be applied following the different computation procedures through the last layer of the network:

$$\frac{\partial L_i}{\partial w_{11}^4} = \frac{\partial L_i}{\partial a_1^4} \frac{\partial a_1^4}{\partial z_1^4} \frac{\partial z_1^4}{\partial w_{11}^4} \,. \tag{14}$$

Two derivatives can be calculated directly which gives

$$\frac{\partial L_i}{\partial w_{11}^4} = 2(\hat{a}_1^4 - a_1^4) \cdot \frac{\partial a_1^4}{\partial z_1^4} \cdot a_1^3 \,. \tag{15}$$

The connection between a_1^4 and z_1^4 is only based on applying the activation function. Depending on the network different activation functions $f_{\text{activ}}(x)$ can be used. In this example the sigmoid function is used which is defined as

$$f_{\text{sigmoid}}(x) = \frac{1}{1+e^{-x}} \,. \tag{16}$$

The derivatives now become

$$\frac{\partial L_i}{\partial w_{11}^4} = 2(\hat{a}_1^4 - a_1^4) \cdot a_1^4(1 - a_1^4) \cdot a_1^3 \,, \tag{17}$$

and the expression can be simplified with the definition

$$\delta_1^4 \coloneqq \frac{\partial L_i}{\partial z_1^4} = 2(\hat{a}_1^4 - a_1^4) \cdot a_1^4(1 - a_1^4) \,. \tag{18}$$

With this definition all derivatives with respect to the parameters of the network from the previous layer which directly influence the last upper neuron can be calculated which gives

$$\frac{\partial L_i}{\partial w_{11}^4} = \delta_1^4 a_1^3 \,, \quad \frac{\partial L_i}{\partial w_{12}^4} = \delta_1^4 a_2^3 \quad \text{and} \quad \frac{\partial L_i}{\partial b_1^4} = \delta_1^4 \,. \tag{19}$$

This illustrative scheme can be generalised and applied to any neuron giving the rule

$$\frac{\partial L_i}{\partial w_{mn}^l} = \delta_j^l a_k^{l-1} \quad \text{and} \quad \frac{\partial L_i}{\partial b_j^l} = \delta_j^l \,. \tag{20}$$

Additionally, the δ_j^l can be calculated by going backwards through the network starting at the output layer with the recursive formula

$$\delta_j^l = \sum_k \delta_k^{l+1} w_{kj}^{l+1} a_j^l (1 - a_j^l) \,. \tag{21}$$

With this technique the derivative of the cost function with respect to any model parameter can be calculated in an efficient way [30]. By combining the backpropagation procedure with gradient descent the model can now be trained.

3.6 The original Transformer

The Transformer is a special machine learning model which relies on the so called attention mechanism. It was introduced in the context of language translations by Google researches in 2017 [31]. This new technique was a great milestone in machine learning and is well known because it is used in the field of language processing and therefore in systems like ChatGPT [26].

3.6.1 Input, Embedding and Positional Encoding

The input given to the Transformer consists of several vectors arranged as a sequence. These vectors can for example represent parts of written text, where to each word a unique vector is assigned, while in practise subwords are used. The number of these vectors is called sequence length which depends on the length of the text given to the model. There are also other examples which can be given to the Transformer but all possible inputs need to have in common that they can be represented by unique vectors. For example if a text passage consisting of ten text snippets is given as an input, the sequence length would be ten and it would consist of ten vectors each representing a snippet [32]. Each vector in the sequence can be written either as a row or a column in a matrix where here the convention of rows will be used [26] [32].
The first step of the transformer procedure is called embedding. Here, the input vectors in the matrix get transformed by a linear transformation into the embedding space which has a different dimension, resulting in a new matrix with new vectors $\boldsymbol{x}_i \in \mathbb{R}^d$ as elements. Each vector, which was assigned to an input snippet beforehand - for example a subword - now has a different dimension d, while the number of vectors stayed the same. The parameters of this embedding matrix, which is used for the transformation, are learnable parameters whereas the embedding dimension d is a fixed value and needs to be chosen [26] [31] [33].
Positional encoding is used to attach additional information to the input element concerning its position in the sequence. This is done by adding a vector with dimension d to the embedding vector. This additional vector has a function

$$\mathrm{PE}_{\mathrm{pos},2i} = \sin\left(\frac{\mathrm{pos}}{10000^{2i/d}}\right) \tag{22}$$

as components which have an even index $2i$ and

$$\mathrm{PE}_{\mathrm{pos},2i+1} = \cos\left(\frac{\mathrm{pos}}{10000^{2i/d}}\right) \tag{23}$$

for components with an odd index $2i + 1$ while pos is the position of the respective input element in the sequence. There are also other choices on how to implement such information [31].

3.6.2 The Attention mechanism

With the attention mechanism one can find connections between different input elements. This is done by computing three new vectors which one receives by multiplying the embedding vectors with three matrices with learnable parameters respectively. The different objects are called query vectors

$$\boldsymbol{q}_i = \boldsymbol{x}_i U_{\mathrm{Q}}\,, \tag{24}$$

key vectors

$$\boldsymbol{k}_i = \boldsymbol{x}_i U_{\mathrm{K}}\,, \tag{25}$$

and value vectors

$$\boldsymbol{v}_i = \boldsymbol{x}_i U_{\mathrm{V}}\,, \tag{26}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the embedding vector for input element i and all three learnable matrices have the dimension $\mathbb{R}^{d\times d}$. In reality all of these vectors are put as rows into matrices Q, K, V and X all with dimension $\mathbb{R}^{s\times d}$ where s is the sequence length, resulting in matrix multiplications

$$Q = XU_{\mathrm{Q}}\,, \quad K = XU_{\mathrm{K}}\,, \quad \text{and} \quad V = XU_{\mathrm{V}}\,. \tag{27}$$

Now the dot product between all query vectors and all key vectors is computed and can be written in the short notation

$$\frac{QK^T}{\sqrt{d}}\,, \tag{28}$$

while the dot product is scaled with the square root of the embedding dimension. This product now has the dimension $\mathbb{R}^{s\times s}$. Then, the softmax function which is defined as

$$f_{\mathrm{softmax}}(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}} \tag{29}$$

is applied row-wise, turning the sequence of numbers into a probability distribution. The scaling by $1/\sqrt{d}$ is done to prevent giving large values, which may arise for large embedding dimensions, to the softmax function. Lastly, the result is multiplied by the value matrix, giving in sum

$$\mathrm{Attention}(Q, K, V) = f_{\mathrm{softmax}}\left(\frac{QK^T}{\sqrt{d}}\right) V\,. \tag{30}$$

Before applying the softmax function masks can be applied to ignore certain values. These mask would set values in the QK^T matrix to $-\infty$ which has the consequence that the applied softmax function gives zero at this entry [26] [31] [33].
In the Transformer, instead of a single attention procedure as above, a Multi-Head attention is performed where multiple attention computations are executed in parallel. Both principles are shown in figure 6. The keys, values and queries now do not have d as their dimension but rather $\frac{d}{h}$ with h the number of parallel attention layers. This is realised by linearly projecting the Q, K and V matrices into subspaces with learnable matrices $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times d/h}$ where i is the respective attention layer. This gives

$$\mathrm{Head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V). \tag{31}$$

In the end the different attention heads are concatenated

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{Head}_1, ..., \mathrm{Head}_i)W^O \tag{32}$$

with $W^O \in \mathbb{R}^{d\times d}$ an additional parameter matrix [31]. This method can get more information out of the input compared to the single attention mechanism because the result of the dot product of key and query in each attention layer still has the same dimension $\mathbb{R}^{s\times s}$ as before, resulting in h different investigations while still regarding the whole sequence [33].

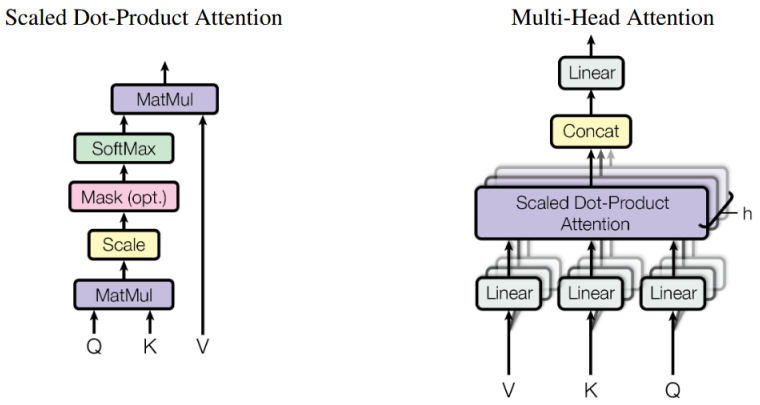


Figure 6: Scaled Dot-Product Attention and Multi-Head Attention. From [31]

3.6.3 Feed-Forward Network

The Transformer architecture consists of a second element, namely a fully connected feed-forward network. Here, the sequence elements have no option to influence each other in contrast to the attention mechanism. Part of the feed-forward network structure are two linear transformations while the output of the first passes through a ReLU activation function from equation 9. Together this yields

$$\mathrm{FFN}(\boldsymbol{x}_i) = \max(0, \boldsymbol{x}_i W_1 + \boldsymbol{b}_1) W_2 + \boldsymbol{b}_2 \tag{33}$$

while the output has the same dimension d as the input. Here it is also possible to arrange the different vectors $\boldsymbol{x}_i$ in a matrix but the feed-forward network is applied separately to every vector [31].

3.6.4 Assembling Transformer layers

The outputs, which are matrices of dimension $s \times d$, of each of the two sublayers are added to the initial input which was given to the respective sublayer beforehand and additionally layer normalization is applied. This procedure can be viewed as adding the corresponding modified vector containing new information to its initial input vector in the sequence, arranged in an efficient way. One building block of the Transformer consists of applying Multi-Head attention followed by the feed-forward network. Together, this gives the procedure for one whole block by first computing

$$X' = \mathrm{LayerNorm}(X + \mathrm{MultiHeadAttention}(X)) \tag{34}$$

and afterwards

$$X'' = \mathrm{LayerNorm}(X' + \mathrm{FFN}(X'))\,. \tag{35}$$

Typically, multiple of these blocks are used [31].

3.6.5 Output

After the computations the output vectors of the sequence in the matrix still have the same embedding dimension. In the context of language processing these vectors are

transformed again into the input space where input elements like subwords in this context are represented by unique vectors. The transformation is carried out by a learnable transformation matrix, which is essentially the counterpart to the embedding in the beginning. Additionally, a softmax function is applied to get a probability distribution for a sequence element [26] [31]. In this thesis a classification task is investigated so the used Transformer follows a different procedure in the last step to generate the desired output.

4 Preprocessing

4.1 Datastructure

The data from the detector and from the simulation processes is stored in so called offline files which are root files, a datastorage concept established by CERN. These files were copied from a computing centre in Lyon to the local computing cluster in Erlangen. Every file corresponds to a certain run with a certain duration which typically amounts to several hours. In principle, the root files can be read out with the km3io software package which can be found in [34]. The read out and processing of these files is not the fastest, as we will see later. The files contain - among other specifications - a certain number of events collected in this specific run. There are many parameters associated to each event such as energy, vertex position and direction. An undetermined number of reconstruction solutions called tracks is stored for each event, which leads to a data structure with variable dimensions depending on which event is considered. These subarrays with variable lengths have the effect that the data read out with the km3io package is stored in so called *Awkward* arrays [35] and not in the user-friendly and well known *NumPy* arrays [36]. Typically, the best reconstruction result or - loosely said the best fitting track - is stored at the first place in the corresponding array [37].
Furthermore, there is additional information about the hits stored. These parameters consist of positions which describe the coordinates of PMTs, directions which give information about the orientation of PMTs, the time of arrival, and the ToT [38]. An important distinction can now be made: event- and hit-level features. This distinction is important for later stages because the Transformer receives hit-/pulse-level features as an input but its output is an event-level feature.

4.2 Data conversion

In this thesis the Transformer model from [39] will be used and adapted. This model is based on the GraphNet framework [40] which realises the application of Deep Learning in the context of neutrino telescopes. GraphNet in turn uses PyTorch [41] and PyTorch Lightning [42] in the background. Two file types are implemented in the GraphNet workflow: Parquet and SQLite [40]. For the Transformer architecture the SQLite type was chosen. A noticeable downside of these files is that they are not compressed, which would not be the case for Parquet files, resulting in a fast access to data [38]. Therefore, the first step is to convert offline files to the SQLite format. GraphNet already entails a workflow which consists of three major steps, namely reading, extracting and writing. There, the most important step is the extracting because the task of the reader is only to access the offline file and hand the data over to the extractor, and the writer mainly makes use of the `to_sql` function which is a method of a *pandas* dataframe [43] in which the extracted data is stored. The event-level features are saved in a table called `truth` and the hit-level features in a table called `pulse_map`. In the next section the important methods of the provided extractors will be described and the added modifications.

4.2.1 The Extractor and its modifications

There are two different types of extractors used. One extracts the hit-level information and another one the event-level information while both use the km3io package. The

extracted hit-level features are shown in table 1. Some of them will be important for the input of the Transformer. They are then arranged in a dataframe, where one row

features							
t	tot	pos_x	pos_y	pos_z	dir_x	dir_y	dir_z
Used as an input feature							

features		
trig	dom_id	channel_id
Not used as an input feature		

Table 1: Extracted hit-level features grouped into two different categories indicating whether or not the feature is used as an input feature for the Transformer

contains one hit. Each hit contains a value for every feature and is assigned to an event by a column containing the event numbers. With this event number one can distinguish hits from different events.
The following is now focused on the extractor of the event-level features. Here, the km3io package is also used to read in the values and to find the best fitting track for every event, thus only one track value of variables like energy, position, and direction is assigned to each event. In this step a large constraint was found in the code, where files with no reconstruction of the first event are discarded. This encouraged thoughts on why the first event holds a certain role and decides over the fate of all other events in the file. Nevertheless, this constraint has the consequence that the columns of the reconstructed parameters of some files are completely filled with padding values after the conversion, where the padding value is a value which is inserted if the corresponding parameter does not exist or does not fulfil certain criteria. This value, which is set to the number 999.0, is also used if the original data in the offline files contained NaN or None values.
Each event also contains a unique identifier already mentioned before to distinguish it from all other events. This number is computed by adding strings of the DAQ run identifier, which is a special property of the whole file, the attribute `frame_index` and the `trigger_counter`, whose values both depend on the considered event, all together and inserting a string containing a zero in between them. In addition a weight is assigned to each event. For data events the weight amounts to one, but for simulation events it is different and has to be considered.
Later, cuts which filter out noise events will be applied. Some parameters needed for these cuts are not implemented in the extractor. Two of them are parameters from the JGandalf algorithm. The parameter `JGANDALF_BETA0_RAD` describes the uncertainty on each reconstructed track and the parameter `JGANDALF_NUMBER_OF_HITS` indicates the number of hits from the JGandalf algorithm [44]. Additionally, the parameter `JSTART_LENGTH_METRES` is needed, which is the measure for the "distance between projected positions on the track of optical modules for which the response does not conform with background" [44]. All of these parameters are stored in the file and can be accessed with an attribute called `fitinf`. The situation in accessing the right values is more complicated as, for example, for the energy because additionally to event and track indices there is a new index which is assigned to each fitparameter in the `fitinf` container. For the two parameters from JGandalf this index is 3 and 0, and for the JStart parameter it amounts to 10. By trying to access these parameters, it was found

that not all fitparameters were available for every event. Therefore, the innermost dimension of the `fitinf` object, which is the dimension to index the desired parameter, needs to be filled so that it at least contains a `None` or `NaN` value at the desired place. Otherwise an error occurs when one tries to access a value which does not exist. Then, one can take the chosen parameter and save it for every event and for every track. From this array the first track corresponding to the best track is taken for every event, and this object can now be converted to a numpy array because of its simple structure. The result is now similar to the other parameters which are saved: one - and the best fitting - value for every event. The same procedure needs to be done for the parameter `JMuonEnergy` which is part of a reconstruction object. This parameter indicates whether the energy reconstruction was successful [37]. Another parameter called `rec_type` was also added, which describes the quality of reconstruction tracks. This procedure turned out to be less complicated because the parameter is organized analogous to parameters like energy.
The weights of the simulations mentioned in chapter 2.6 are also included in the extraction process. After the conversion they can be easily accessed by the `w_total` parameter. Their values differ depending on the simulation type. For the neutrino simulation events almost all weights, except less than 0.05%, have values which are smaller than 0.1. For the muon simulation events all weights have values close to one with a maximal deviation of 0.3.
Additionally, a parameter was added which is not needed for applying cuts but rather for the training process of the Transformer later. The parameter is named `is_mc_score`, and its value is one if the corresponding file stems from a simulation and it is zero if the file contains real, measured data. This value is assigned to each event in the file. In later sections it is often referred to as Monte Carlo score or short MC score.
Not only two tables are saved but also a third one. This additional table is called `trigg_pulse_map`. The only difference between this table and the `pulse_map` table is that hits with a `trig` value of zero are filtered out. In this way only triggered hits are considered which are hits evaluated to be caused by neutrino or muon interactions [38].

4.2.2 *Snakemake* workflow

With the help of the *snakemake* [45] workflow provided by Lukas Hennig the conversion can be easily realised. In *snakemake* rules can be defined which will be executed. These rules take a file as an input which is in our case the offline file and produce an output file which would be a file in the SQLite format. Log files, which describe errors in detail if problems occur in the conversion process, and resource requirements can be defined. The actual task is also defined which is executing the data converter from GraphNet equipped with the chosen reader, writer, and extractors.
The execution will be sent out as a job via slurm [46]. Slurm is a system, that manages the available computational resources of computing clusters. It gives access to these resources to the users based on their needs, and prevents conflicts by introducing queues, which has the downside, that sometimes jobs will be pending for a while before they start to run [46]. For scheduling resource demands like the maximal duration of the job computation and the required memory need to be defined. These values vary depending on the type of the file and its size. The minimal and maximal values for file sizes before and after the conversion, the duration of the conversion, and the requested memory

are shown in table 2. The requested memory is relatively high, because the extraction

File type	data	neutrino	muon
Offline file size [GB]	3.0 - 4.5	1.0 - 1.2	2.1 - 3.1
SQLite file size [GB]	1.2 - 1.9	1.3 - 1.7	1.2 - 1.8
Duration [min]	13 - 19.5	6 - 9	11.5 - 17
Requested memory [GB]	69.75 - 108.5	23.25 - 31.0	62.0 - 100.75

Table 2: Minimal and maximal values for file sizes, requested memory and duration for the three different file types of the used files

process is not optimised. Therefore, to save resources one has to carefully evaluate the requested memory for the process for each file size and file type differently. Otherwise, if the memory demand is set too low, out of memory errors will occur. This made the conversion process complicated and tedious. In the used slurm system memory gets distributed by the requested number of cpu cores. Every core unlocks 7.75 GB of memory which explains the precise memory values in table 2. Each of them is a multiple of the memory per core. Thus, for these processes there is a small amount of memory not utilised, which is smaller than 7.75 GB, but by requesting one core less the process crashes eventually when the limit is reached.

4.3 Cuts

Firstly, after the conversion finished and before analysing the data in the SQLite files the padded events need to be removed. The padding value in the extractor was set to 999.0 which makes it in principle easy to filter out such events, but this value comes with the risk of discarding events which own this value naturally. This can be resolved by checking multiple reconstructed values for each event and not, for example, just filtering the reconstructed energy and drawing conclusion only on the outcome of this single parameter. By applying this filter on 100 runs with data, muon simulation, and neutrino simulation files respectively 78% of data events, 58% of MC muon events, and 56% of MC neutrino events were filtered out. The mentioned "if statement" in the extractor is mainly responsible for this large fraction. The run numbers which have been used are $14300 - 14402$ while runs 309, 324, and 325 were not available.
To filter out noise and badly reconstructed events the so called antinoise cut can be applied. It is defined in table 3. Most parameters used here and their definitions have been mentioned before in chapter 4.2.1 where their extraction process was described. The parameter `jmuon_E` corresponds to the reconstructed energy in GeV for each event, the parameter `jmuon_lik` defines the likelihood, and the parameter `dir_z` is the z direction of the event which is given by [37]

$$\cos(\texttt{jmuon\_zenith}) = -\texttt{dir\_z}\,. \tag{36}$$

`jmuon_zenith` is the reconstructed zenith angle in radians, and it was used because it was easily accessible in the truth table of the converted SQLite file. Strictly speaking, these parameters are the values for the best tracks which were assigned to the events as discussed before. The parameters are suitable for removing noise because many of them act as parameters indicating quality in reconstructing steps. Noise events typically cause problems in reconstructing processes which is then reflected in these quality parameters.

Parameters and their requirements	Individual Influence on data runs
`rec_type` $== 4000$	0%
`JMuonEnergy` $==$ `True`	0.5%
`jmuon_E` > 10	9.2%
`jmuon_lik` > 50	20.4%
`JSTART_LENGTH_METRES` > 0	1.4%
`JGANDALF_BETA0_RAD` > 0	0.1%
`dir_z` > -0.1	1.0%
`JGANDALF_NUMBER_OF_HITS` > 20	1.7%

Table 3: The parameters used for the antinoise cut and their corresponding requirements. Individual influence on data events computed by applying condition individually on dataset of 100 data files. Definitions from [47]

By applying the conditions each individually on, for instance, the set of 100 data files, which were previously filtered by removing padded events, their influence can be determined. The measure

$$\text{Influence} = \frac{\#\text{ discarded events due to condition}}{\#\text{ all events}} \tag{37}$$

which is the fraction of filtered events compared to all events is also shown in table 3 for each condition. Of course there may be overlaps when applying conditions together. It can be seen that the `rec_type` parameter does not make a difference because all events where the conditions would not be fulfilled were padded events which have been discarded before. The definition is still kept in the cut for safety reasons. The received statistics do not necessarily hold true for every dataset but by using 100 files the statistics should be in general meaningful.

5 Data and Monte Carlo comparison

The cut procedure from the previous chapter was applied on the 100 data and simulation runs. This number was used to get better statistics. By using another parameter, called `pdgid`, the neutrino flavour can be attached to each event for the neutrino simulations. There is an official Monte Carlo particle number scheme released by the Particle Data Group [48] and incorporated in the dataformat of KM3NeT. Electron neutrinos have a value of 12, muon neutrinos a value of 14, and tauon neutrinos a value of 16. For the identification of antiparticles the same number is used but with an additional minus sign.
An important parameter of the events from the truth table is the reconstructed energy which is shown in figure 7, where data and simulations are compared. Here, the antinoise cut is already applied. Additionally, weights have been applied to the simulations. With

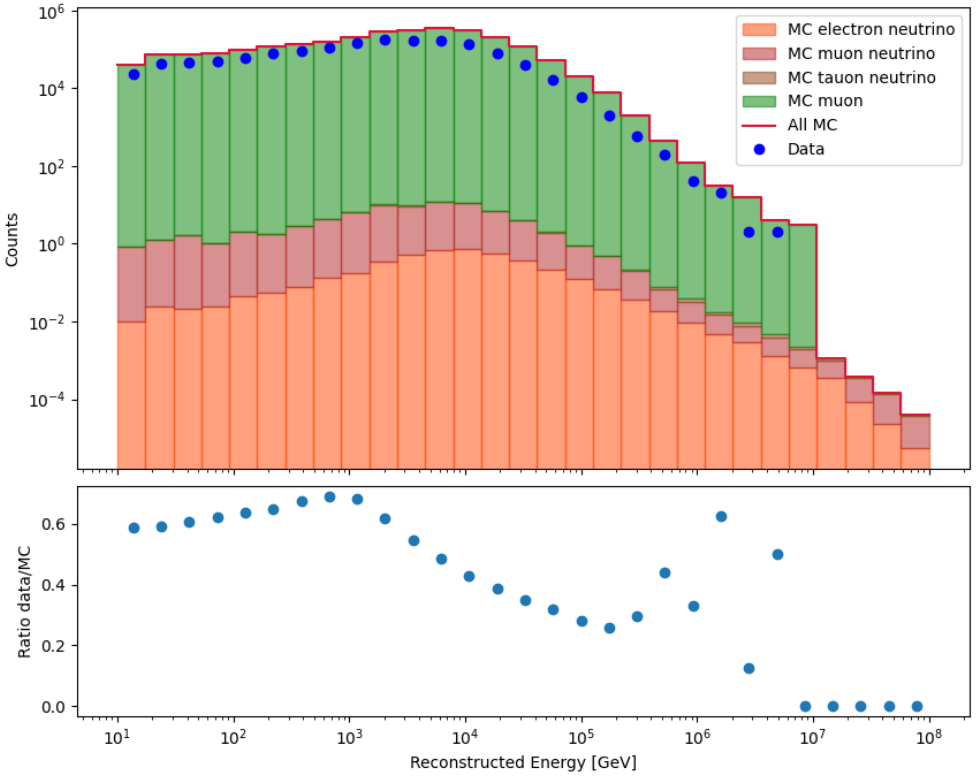


Figure 7: Distribution of reconstructed Energy and ratio for 100 data, muon, and neutrino files. Layout inspired by [47]

this plot one can verify the correctness of the conversion process from root files to SQLite databases by comparing the distribution to that in [47]. It was not possible to reproduce these plots with similar values because here many more events are present while using less runs. Additionally, a maximal discrepancy in the ratios of 25% between data and simulation would be expected but here a larger difference is found, thus there is a reason beyond noise [49]. Attempts were made to resolve this issue by investigating the distributions of some parameters and the extraction processes more closely. In the last stages of the work it has been found that a convention different to equation 36

Type	Total number of events
Data	1435522.0
Muon	2639378.7
Neutrino	79.4

Table 4: Total number of events from figure 7 for the three different event types

has been used in the extractor, where no minus sign is used to calculate and save the `jmuon_zenith` parameter in the SQLite database. Therefore, when using this in the SQLite file accessible parameter to apply the antinoise cut with definitions from table 3 and equation 36 another value range of `dir_z` has been considered than originally expected. This explains the discrepancy of the distributions in figure 7 and in [47]. Originally, the cut would remove most atmospheric muons by only considering a certain region as described in section 2.3.2 but now the opposite is achieved. Therefore, the large amount of detected events is plausible. The observations and conclusions in the following sections still hold true because fundamental deviations in the muon simulations from real data, which are to be investigated should remain similar regardless of the direction region, which is considered [49].
In figure 7 it can be seen that there is a large difference in counts between muon simulations and neutrino simulations. For these particular runs the total number of events is shown in table 4. This discrepancy led to the decision of not using neutrino simulation runs for the training of the Transformer because in comparison the number of neutrinos is so small it would probably be ignored by the model.
From figure 7 and from table 4 it can be seen that there is a discrepancy between simulations and data. In an ideal scenario the ratio between data and Monte Carlo simulations would be 1 which is far away from our value which amounts to 0.54. Now, the idea was to investigate the root of the data Monte Carlo discrepancy in general further. This is realised by training a Transformer on the task of discriminating between data and Monte Carlo simulation which will be in this case muon simulations as mentioned earlier. Then, by analysing a successful model the reference points, which influence the "decision" of the model, can be studied.
The antinoise cut will be applied to all datasets used in the following sections before the other methods described in them will be applied.

6 The Transformer and its training

In the following the Transformer architecture from [39] is used. To explain its structure and integration in the machine learning workflow the code from [39] will often be used as well as the GraphNet documentation [40]. The utilised functions often rely on PyTorch and its documentation was also used [41].

6.1 Some further preprocessing

6.1.1 Dataloader and Dataset

For the training process and later the evaluation a so called dataloader is needed. This ensures quick and easy access to the data for the model and a well defined organisation of the data. In the repository there is a predefined function which assembles a training and validation dataloader for a given SQLite file. The idea is that for the dataset a predefined percentage of the data is used for the training of the model, where the loss is calculated and the parameters get adjusted, and the remaining data is used as a validation set, where only the loss is calculated but no parameter adjustments are performed. This separate iteration allows to monitor if the model overfits and based on that permits interfering. Here, overfitting means learning the training set by heart and not the underlying patterns. If the model overfits it performs badly on the validation set whereas if it learns meaningful patterns not only the training loss but also the validation loss should decrease in general. Here, the distribution was chosen to be $75 : 25$, thus 75% of the events in the whole dataset are assigned to the training set and 25% to the validation set. Another important term which will be later used is the epoch. This is defined as giving the whole dataset to the model which then trains on the whole training set and evaluates the whole validation set both for one pass. Thus, each event is presented to the model exactly one time.
For the dataloader the input features have to be selected while our choice was already mentioned in table 1 and the important truth-level information which is in our case the previously defined Monte Carlo score. The batch size needs to be selected which was chosen to be 64. Here, one has to find a balance between computational resources which rise for higher batch sizes and performance of the model as stated before. Each batch then contains 64 events distributed randomly except the last batch because in general the number of all events in the dataset will not be divisible by 64. Additionally, weights for the loss function can be assigned where this weight is 1 for data events and the corresponding weight from the simulations for the muon events.
In GraphNet a graph definition has to be specified which is also given to the dataloader. This definition specifies the processing and arrangement of the data before it is given to the model. For the graph definition the KM3NeTHitsSequence class was used. In this setting a detector has to be specified where the already implemented ORCA class was modified and renamed to an ARCA21 class. Unlike in the other classes where there is an option to shift and rescale the input features nothing comparable was implemented for the ARCA class and the input features were left unchanged. A node definition needs to be chosen. For this study, the NodesAsPulses class was chosen, where each hit, which is a row in the pulse-table, is represented as a node.
Internally PyTorch is used to create a dataset from the SQLite file which is then handed to the dataloader. One batch in the dataloader now contains the specified amount of

events and their respective hits. It works similar to a dictionary in Python where one can access single parameters with their corresponding strings. The parameters are given in a PyTorch tensor object. This is the structure which is mainly used in the processing of the model.

6.1.2 Data selection

For the data, which is used for training the model - referred to as training data in the following section while this dataset includes in principle also the validation set - 11 data files and 11 muon simulation files have been selected. They stem from the 100 files discussed earlier. Together the number of events from these files is in the order of 10^5 which was selected to balance out the need of computational resources and having a large enough dataset for the Transformer. If the dataset is too small the Transformer may overfit and may not be able to extract the desired, hidden patterns. Files with low numbers of padded events have been chosen. Due to the architecture of the dataloader the SQLite files for each run have to be concatenated to one SQLite file which contains all events and additionally the cuts need to be applied. Here, the procedure is as follows: each SQLite file was loaded separately in a Python script where the cuts have been applied. Now the individual hits are also considered in the further procedure, so hits belonging to filtered events need to be removed by comparing the unique event numbers. Furthermore, a selection based on how many hits an event contains is made that will be described in the following chapter. Then, new smaller SQLite files with cutted events are created and all of them are concatenated. Performing cuts first and concatenation afterwards saved working memory but caused higher main memory usage. This trade-off was found to be better suited for this task. The resulting SQLite file with a size of 17 GB now contains all remaining events regardless of file type.

6.1.3 Sequence lengths

As stated in chapter 3.6 a sequence is given to the Transformer. Here, a sequence consists of all the hits assigned to an event. This number varies for different events but in the Transformer architecture explained later in 6.2 a maximal sequence length needs to be chosen. Therefore, a possibility would be to find the event with the longest sequence length in the dataset set and select it as the maximal length. However, there are multiple events in a batch and the event with the longest sequence determines the sequence length in that batch. The sequences of other events are filled up with zeros until all events in the batch have the same sequence length [38]. Thus, batches containing events with relatively long sequences would use many computational resources and have a significant influence on all other events in the batch. As a consequence a maximum sequence length was chosen where events with longer sequences are discarded. The distributions of the events according to their sequence length is shown in figure 8 for the 11 data runs and in figure 9 for the 11 muon simulation runs. For both distributions the 99-th percentile was computed and the resulting range is highlighted in the distributions. This gives a maximum sequence length of 314 for data and 343 for muon events, and based on this a filter for the training dataset was performed. Due to the favourable distribution most events can be included while having sequence lengths which are not too large to require great amounts of computing resources. All in all with the previously discussed cuts applied this gives a total number of events of around $4.4 \cdot 10^5$ for each

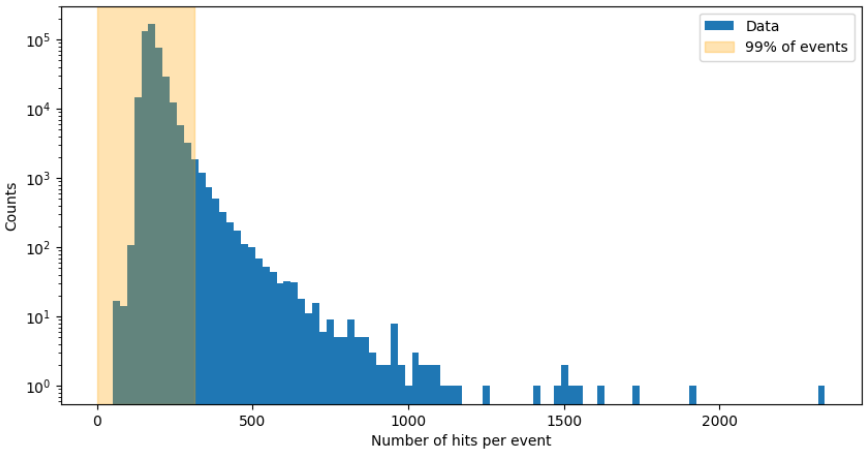


Figure 8: Distribution of events according to their sequence length/Number of hits per event for the 11 data runs used for the training

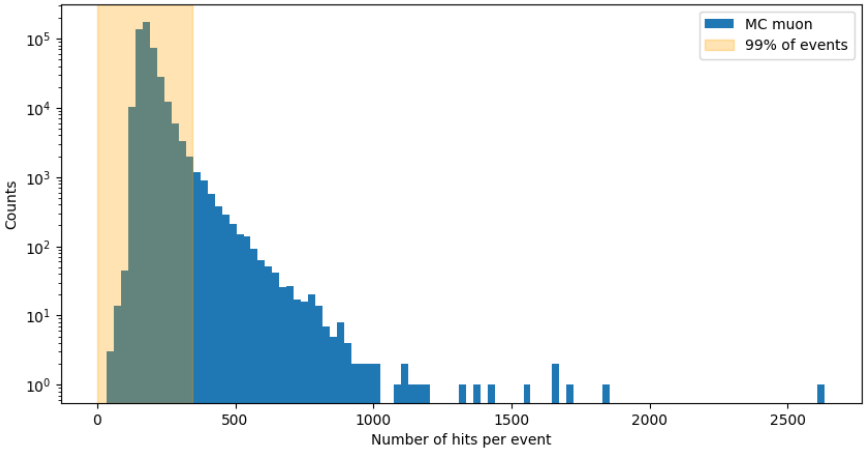


Figure 9: Distribution of events according to their sequence length/Number of hits per event for the 11 muon simulation runs used for the training

file type respectively. For the first training which will be described later in chapter 6.4 the $\sim$ 2400 muon events which have larger sequence lengths than 314 but smaller or equal lengths than 343 were not discarded. This should have been avoided because in principle this could have lead to a bias. While a direct check was not carried out, actions described later will be taken so that this potential bias does not distort the analysis. For the second training this problem was fixed and events regardless of their file type have the same maximal permitted sequence length.

6.2 Transformer architecture

The architecture of the model is based on the structure described in Chapter 3.6. When the model is initiated for the first time several parameters need to be chosen which have an influence on the structure. For instance, there is the option to chose multiple layers or a single linear layer for the embedding. Here, only a single layer was chosen. The

embedding dimension was selected as 128. The number of features is eight as discussed before and the positional encoding from chapter 3.6.1 was enabled. The default setting of eight attention heads was kept.
The first step in the so called forward-pass is converting the batch which has two dimensions, namely the total number of hits in the batch and the features, to a sequence with three dimensions, namely the batch size, the sequence length, and the features. Here, shorter events are padded with zeros as mentioned earlier. After this step the embedding is realised by a linear transformation implemented with a simple PyTorch module. Now, the passed object has three dimensions: batch size, sequence length, and embedding dimension. A so called class token is initialised as an additional, learnable parameter and with suitable dimensions, so that it can be assigned to each event. If enabled the positional encoding is added to the sequence and then the class token and the sequence are concatenated to a single object. Depending on the choice a number of encoder blocks without considering the class token and then a number of encoder blocks with considering the class token is carried out. The default setting of eight and four for the amounts of these two block types were not changed. The encoder block is the centrepiece of the Transformer. It consists of the Multi-Head attention process and the feed-forward network as well as layer normalisation. The activation function in the feed-forward network is the so called GELU function, unlike in chapter 3.6.3 where ReLU was used. For the attention and the feed-forward network a technique called dropout is used. This has the effect that some elements are set to zero randomly with a predefined probability where the default value is 20%, increasing the robustness of the model and reinforcing learning patterns rather than memorising [27]. All these different components are implemented with the corresponding modules from PyTorch. At the end the updated sequence is returned.

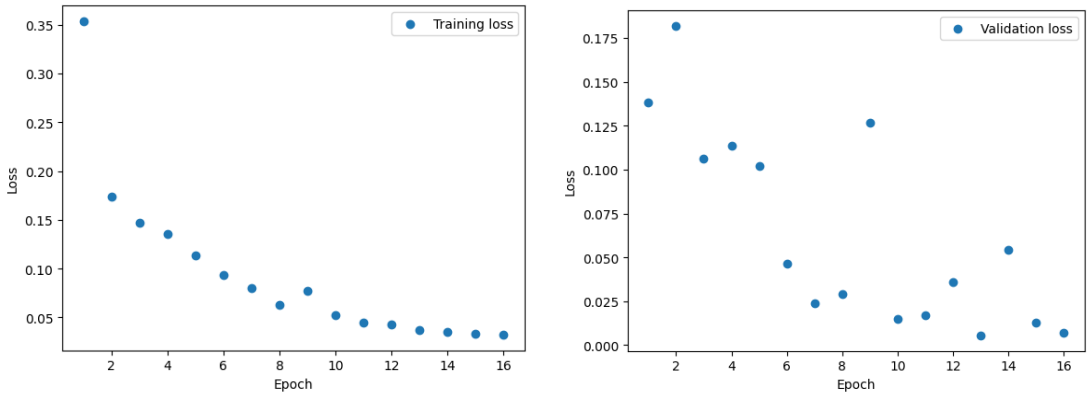
6.3 Model building

In GraphNet a so called Standard Model can be assembled which connects all parts and allows to use many built in functions. For this a task needs to be specified which processes the output of the Transformer. In our case the Binary Classification Task was used to predict whether an event is simulated or originates from real data. The chosen loss function is the Binary Cross Entropy Loss from equation 12 and the weight for the loss function can be used. Due to the cuts the muon weights mentioned in section 4.2.1 have now only a maximal deviation of 0.03 from one. Therefore, these weights will not have a large effect but were still used because of the simple implementation. The task module has the assignment to transform the output of the model according to the desired output and to calculate losses. Within this module, the output of the Transformer is transformed to a single number per event via a learnable transformation and then the sigmoid function is applied to get an output in the interval $[0, 1]$. Then, the Standard Model can be initialised which combines the graph definition, the Transformer model, the task, and the optimizer, for which ADAM was chosen. In total, the Transformer model has 1.6 million trainable parameters and the task, which is used to transform the output to the final result, has 129 learnable parameters.
The Standard Model is in turn built from other classes which inherit eventually the PyTorch lightning module. This has the effect that the process of training can be automated in a very efficient and user-friendly way. Eventually, training the model can

be performed by merely executing the `StandardModel.fit` function.

6.4 First training

For the first training the maximum sequence length was 343 as mentioned before and the maximal amount of epochs was chosen to be 20. The default, constant learning rate of 0.01 was chosen. The so called early stopping option is used, which stops the training after a certain number of epochs has passed, in which the validation loss remained the same or even deteriorated. With this technique one automatically gets the model, which performs best on the validation set, from all undergone epochs because the last setting does not have to be necessarily the best model. The epochs for the early stopping option were chosen to be 3. In the training set there were 10389 batches and the validation set consists of 3464 batches. The combined set contains around $8.8 \cdot 10^5$ events and around $1.6 \cdot 10^8$ hits in total. The model trained for 16 epochs, while the best model was found on epoch 13. This is the model referred to in the following sections as the model from the first training or also called the first model. The training and validation loss over the epochs are shown in figure 10. The training loss appears to be relatively



(a) Training loss (b) Validation loss

Figure 10: Training and validation loss for the first training

stable and is declining in nearly each epoch compared to the previous one, whereas the validation loss fluctuates more. Additionally, the model's predictions for the validation set are shown in figure 11. The validation set contains about $1.1 \cdot 10^5$ events per type, and hence serves as a good representation to test the model. To analyse the models performance a metric like accuracy can be used. It is defined as

$$\text{Accuracy} = \frac{\#\text{ correctly classified events}}{\#\text{ all events}}. \tag{38}$$

It has to be chosen what a correctly classified event is. For now, the rather strict definition of

$$c_{\text{strict}}(p,t) = \begin{cases} \text{correct } (1) & |t-p| < 0.05 \\ \text{false } (0) & |t-p| \geq 0.05 \end{cases} \tag{39}$$

is used where p is the prediction and t the actual target. With this definition the model has an accuracy of 98.5% when applied to the validation set. It is important to note

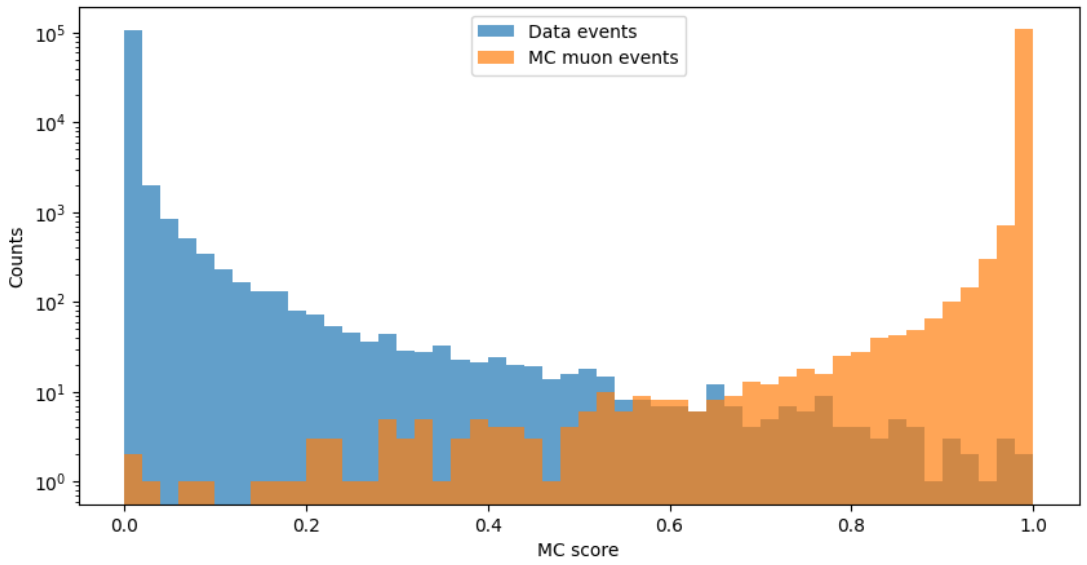


Figure 11: Prediction of validation dataset with best model from first training

that the model's parameters were not adjusted by considering the validation set - except choosing the best performing model - and the model therefore learnt underlying patterns and did not simply memorise the dataset by overfitting. Thus, the model has the ability to successfully distinguish most events from an unknown dataset. Later, the model will be analysed on a second, new set - the test set - while the focus will be mainly distributed on the following second model.

6.5 Second training

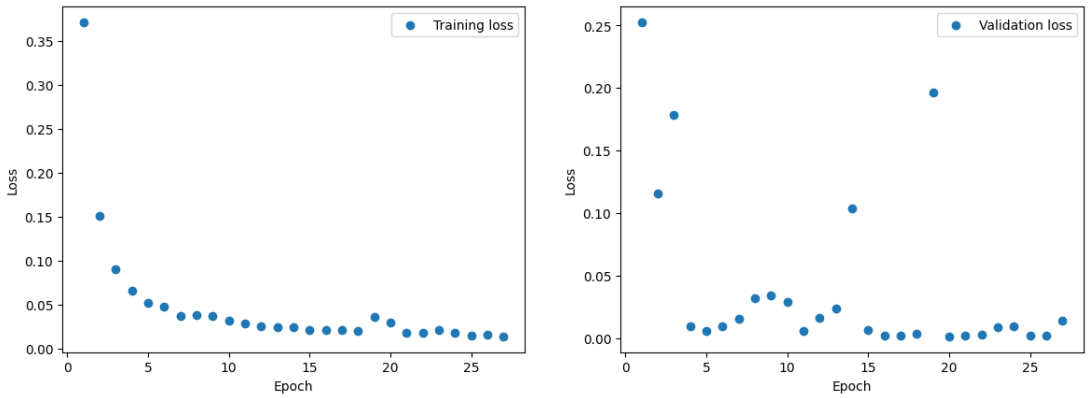
6.5.1 Unique hits

The first model was trained on all hits which were assigned to an event. A property which the model could pick up on are so called after-pulses. They appear in real data but not in simulations which could in principle make it easy for the model to distinguish data and Monte Carlo simulations [38]. After-pulses are delayed signals from for example released ions. For the second model these after-pulses will be removed from the training data. By doing so, it is assured that the model does not have the ability to use them, and other features, which may be unknown, can be investigated. To realise this only the first hit on each PMT in an event is used but with this approach not only after-pulses are removed. In the removed hits there may be hits from the background or hits from other muons. For simplicity in the following sections it will only be stated that after-pulses are removed but it should be remembered that other hits may be discarded as well. The PMTs do not have an unique identifier. Therefore, the `dom_id` - which is fortunately an unique identifier - from table 1 needs to be used to identify all DOMs assigned to an event. Then, the `channel_id` is used, that specifies which of the 31 PMTs of the particular DOM is considered. Thus, if there are hits with identical event number, DOM id, and channel id, only the first hit in time will be taken and the others will be discarded. For each single file around $2.5 - 4\%$ of the hits were affected, resulting in a decrease of 3.4% of the hits for the combined dataset. Nevertheless, the

combined dataset for training and validation still contains about $8.8 \cdot 10^5$ events and $1.5 \cdot 10^8$ hits in total. The hits in this new dataset after the filtering has been carried out will be called unique hits [38].

6.5.2 Training procedure

The potential problem from before was fixed and now both data and simulations have the same maximal sequence length of 314. Therefore, the maximal sequence length was also adjusted in the model setting. Applying the unique hits method had the effect that the sequence lengths dropped, thus only around 70 muon events had to be removed due to their sequence length exceeding the 314 threshold. Therefore, only the number of batches in the validation set decreased by one while the training set contains the same amount of batches as before. The maximal amount of epochs was chosen to be 30 and the number of epochs after the training is cancelled, when the validation loss does not decrease further, was increased to seven. This was used to give the model more opportunities to find even better parameters than before. All other adjustable parameters remained unchanged. The model was trained for 27 epochs, and the model with the lowest validation loss was created on epoch 20. Therefore, it was chosen as the best fitting model. This is the model referred to in the following sections as the model from the second training or also called the second model. Training and validation losses are shown in figure 12. It can be seen that the training loss seems relatively



(a) Training loss

(b) Validation loss

Figure 12: Training and validation loss for the second training where only unique hits have been used

stable as before with a few small outliers. Additionally, the validation loss seems to fluctuate in general less than in the first training but this could also be because the y axis scaling differs from the scaling for the first training. This time an additional metric was saved, namely the accuracy on the validation set. The interesting property would be the accuracy for each epoch. But due to the complicated code structure it was only possible to implement the accuracy per batch for every epoch. From this the accuracy per epoch can be calculated. If all batches had the same batchsize it would be directly possible to sum the accuracies of all batches in the epoch and divide the sum by the number of batches. Unfortunately, the last batch has a batchsize of 55 which differs from all other batches which have batchsize 64. Therefore, with

equation 39 the number of correctly classified events in all batches from an epoch can be calculated by rearranging the formula and inserting the total number of events in the batch. Then, all correctly classified events are summed up, giving the total number of correctly classified events in the epoch. The number of all events is also known which is just $64 \cdot (\#\text{ Number of batches} - 1) + 55$. This procedure can now be repeated for every epoch. The accuracy per epoch on the validation set is shown in figure 13. For

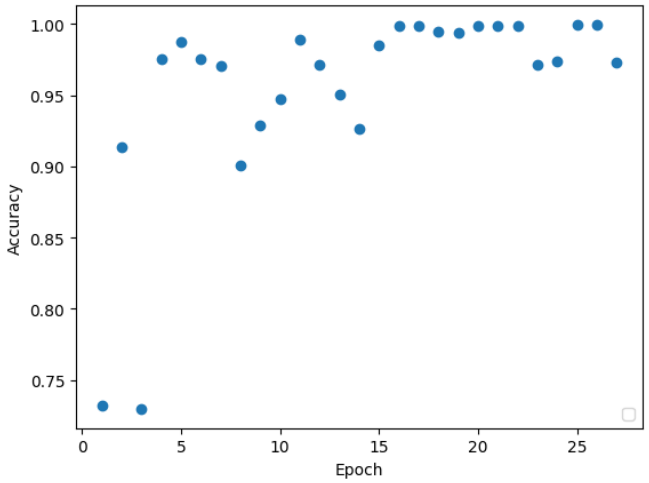


Figure 13: Accuracy per epoch for the training with unique hits calculated from individual accuracies of batches with small error introduced due to the estimation

earlier epochs the metric fluctuates while the fluctuations decrease for later epochs, but they still occur although to a lesser degree. For both models a constant learning rate was used which still gave a good result. For further training it may be better to implement a variable learning rate which gets smaller for higher epochs. This could help to counteract fluctuations in the later stages whereas fluctuations in earlier epochs happen probably because the model still converges to a minimum and did not yet settle. There, steps have a greater impact on the performance as in later training stages, and with a smaller learning rate at the end the optimal point can be found more easily.
As with the first model the predictions on the validation set can be computed and visualised. They are shown in figure 14. By comparing figure 11 and figure 14 it can be postulated that the model performance increased. The number of events stayed nearly the same and there are less events between the outermost bins. The performance can be quantified again by computing the accuracy as before with the same definition. The accuracy on the validation set amounts to 99.9% and therefore the performance increased. This effect can not only appear due to the 70 muon events which were removed between the two trainings due to their sequence lengths which not have to but may possibly been classified poorly. Only 20 of these events were even inside the validation set, and the number of correctly classified events increased by around 3000 events from the first to the second training procedure. Additionally, it can be said that the concern that the model's ability of distinguishing between data and simulation is mainly based on after-pulses and therefore it could perform worse by removing them, was unfounded. Although the second training started at a higher validation loss it quickly dropped into the same regime as in the first training process. The reason for

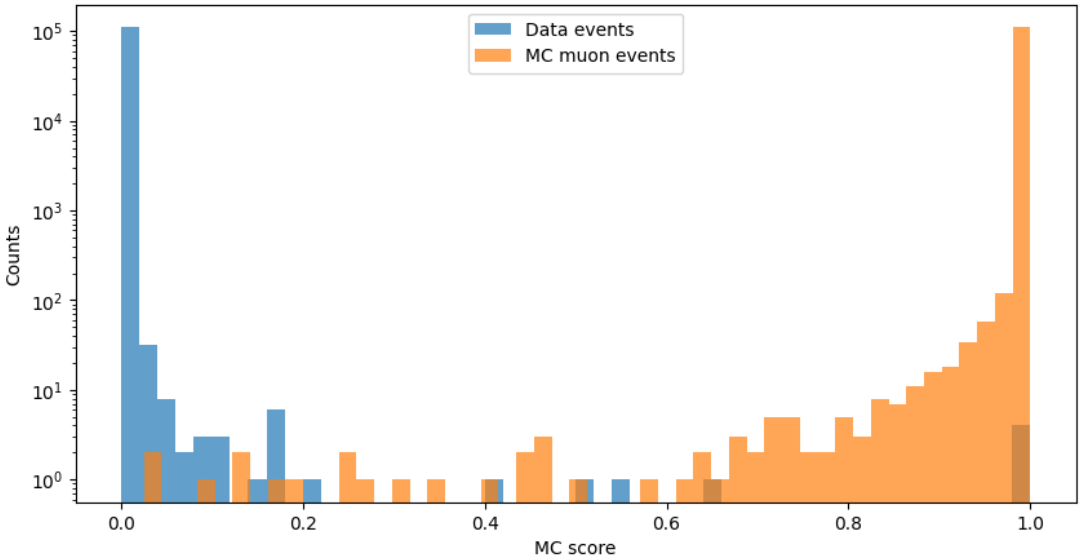


Figure 14: Prediction of validation dataset with best model from second training where only unique hits have been used in the dataset

the model performing better could be caused by a longer training which amounts to seven more epochs.

7 Results

In this chapter both models will be analysed on a test set to investigate patterns on which the models base their "decision". This new set will be used because it is easier to handle than the validation set. Training and validation sets from before are not divided into different files but they are combined in one file and are assembled in the training script. If the goal is to make changes to the data used for prediction like in the following section it can only be effectively done for the whole file, which would be a waste of time and resources because 75% of the data is assigned to the training set and this set must not be used to analyse the predictions of the models. At the same time a further check can be performed whether the model generalises. The focus will lie on the second model, which was trained on unique hits, because it not only performs better but additionally it is assured that the underlying patterns are not based on the already known after-pulses. The same definition of unique hits is used as before, where only the first hit on each PMT in an event is considered, removing the after-pulse but possibly also hits from other sources. The test set consists of two data runs and two muon simulation runs with run numbers 14358 and 14360. The events within these files are completely unknown to the model and were not used for either training or validation sets. The models will be evaluated on this newly chosen set, and additionally it will be investigated which features of the eight hit-level inputs are especially important for the model prediction.

7.1 Feature shuffling

To realise this proposed approach each feature will be shuffled individually which essentially means that their corresponding values will be randomly redistributed among the hits. All other features except the shuffled one will stay in the same order and remain assigned to the same hit. Therefore, values of this certain feature do not only swap between hits from the same class but also may swap between them. If the particular feature is important for the model's prediction the performance should change because by shuffling information was lost. If the particular feature is unimportant and barely considered by the model the prediction should not change drastically.
If the values would not be shuffled between types the result would not directly indicate if the shuffled feature is important for the model's decision or not. For example, in the case of a high amount of values of a feature which only appear for one type, being an indication, that may be used by the model, shuffling may not produce a performance loss because the events still get recognised as the correct type by considering these unique values of the feature. However, no or a small performance drop could also mean that the shuffled feature is not important for the model's decision. Therefore, to assure that the performance drop is related to the importance of the feature to the model's prediction shuffling between types is also permitted. The procedure of shuffling a complete column associated with one feature in the test set's data frame and saving the result takes around $25 - 30\,\mathrm{min}$.

7.2 First model

Firstly, the model from the first training process will be used to create the predictions on the test set. In the test set all events with a sequence length greater than 314

were discarded to avoid falling for the unlikely but potential bias which may have been introduced, because of the small amount of muon events with larger sequence length as mentioned before. The number of events in the test set amounts to $7.9 \cdot 10^4$ events per type and the whole test set contains about $2.8 \cdot 10^7$ hits. The model's prediction on the test set are shown in figure 15. The performance seems to be similar to the

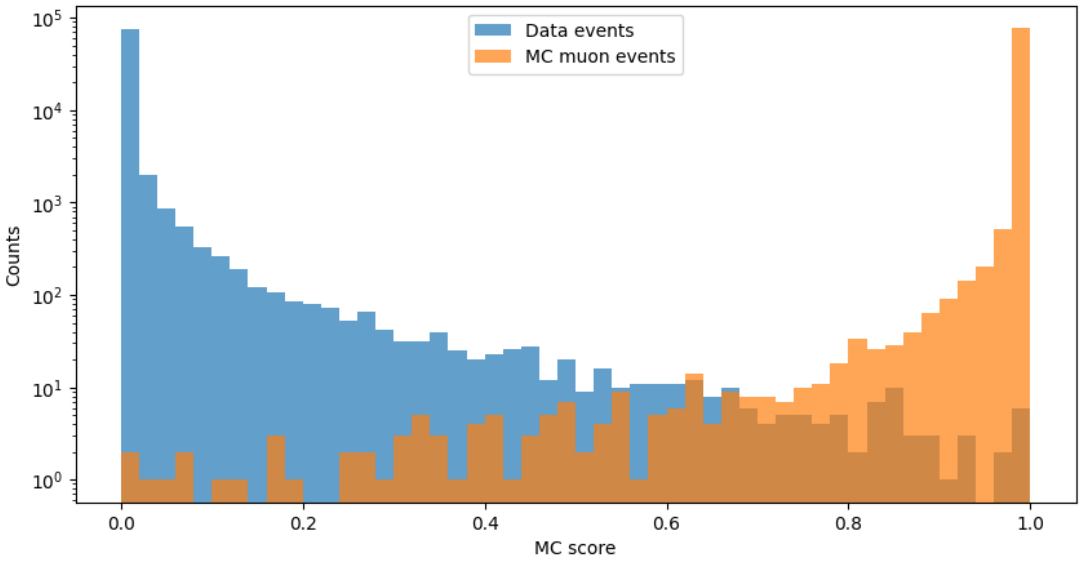


Figure 15: Prediction of test dataset with best model from first training

performance on the validation set as expected. The accuracy amounts to 97.8%. By inspecting the performance on the test set a reference was set up which can be compared to the prediction when shuffling has been carried out. Each of the features was shuffled individually and the corresponding prediction of the model was computed as well as the accuracy on each set. The different accuracies belonging to the set of the shuffled feature are shown in table 5. It can be seen that the accuracy drops to $\sim 50\%$ when shuffling `tot`, `pos_x`, and `pos_y`. A smaller decrease of $\sim 5\%$ occurred by shuffling `pos_z`. The arrival time only dropped less than 3% and shuffling the direction parameters did not make any significant difference. In this case, the strict definition of the accuracy is used as before and thus it is important to look also at the distribution. In principle, it could be the case that many events are now slightly outside of the boundaries and the accuracy drops drastically while most events are still relatively close to their true target value. Therefore, the distributions of the three features which when shuffled have a large accuracy drop are shown in figure 16, 17, and 18. The other distributions are shown in the appendix in chapter A.1. It can be seen that indeed for, e.g. shuffling ToT, some events are more spread out compared to the unshuffled distribution. Therefore, a more loose accuracy definition can be defined by choosing

$$c_{\text{loose}}(p,t) = \begin{cases} \text{correct } (1) & |t-p| < 0.4 \\ \text{false } (0) & |t-p| \geq 0.4 \end{cases} \tag{40}$$

to quantify if the accuracy drops with this definition are not that drastically. This new definition of accuracy is also shown in table 5. In either way the definition for

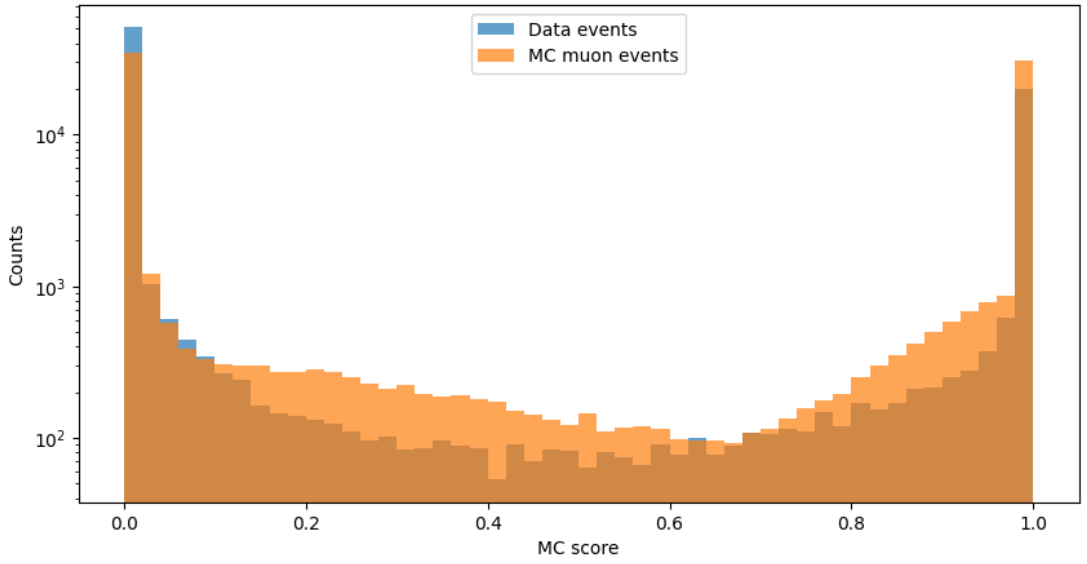


Figure 16: Distribution of first model's prediction on test set with shuffled `tot`

Figure 17: Distribution of first model's prediction on test set with shuffled `pos_x`

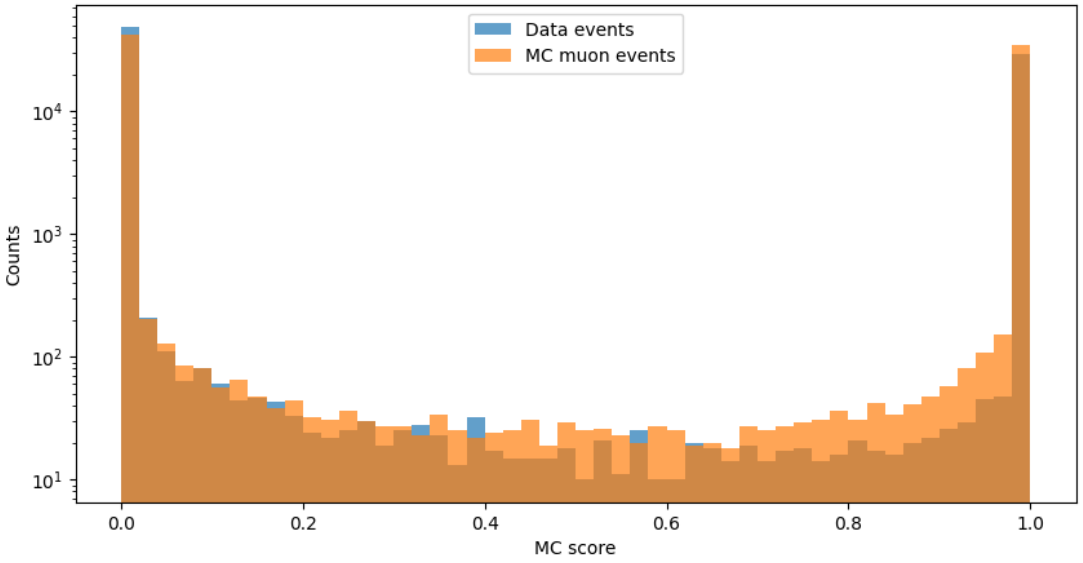


Figure 18: Distribution of first model's prediction on test set with shuffled `pos_y`

Shuffled feature	Accuracy (strict)	Accuracy (loose)
Unshuffled	97.8%	99.8%
`tot`	53.2%	58.1%
`t`	95.4%	99.1%
`pos_x`	51.8%	52.0%
`pos_y`	52.8%	53.6%
`pos_z`	92.9%	98.0%
`dir_x`	97.7%	99.8%
`dir_y`	97.8%	99.8%
`dir_z`	97.7%	99.8%

Table 5: Different accuracies (strict and loose) from first model's prediction on test set for shuffling features individually

"correct" is arbitrary and thus can distort the accuracy measure of the model if chosen too strictly. The differences between the different accuracy definitions do not shift much. The highest difference of $\sim 5\%$ can be found for the ToT parameter, while it does not compensate the large drop of accuracy between shuffled and unshuffled. However, the main reason why the accuracies drop by such a large factor is that many events are classified as the opposite type. In the outermost bins within all three figures the number of correctly classified events and the number of falsely classified events are in the same order of magnitude. To further analyse the performance - especially the phenomenon of classifying events falsely with high certainty - two additional metrics can be defined. The precision which is given by [27]

$$\text{Precision} = \frac{\#\text{ correct positive events}}{\#\text{ correct positive events} + \text{false positive events}} \tag{41}$$

as well as the sensitivity which is given by [27]

$$\text{Sensitivity} = \frac{\#\text{ correct positive events}}{\#\text{ correct positive events} + \text{false negative events}}\,. \tag{42}$$

The before investigated classification task does not separate between the for these metrics needed terms positive or negative. Therefore, these measures will be computed for two different purposes which then allow to define positive and negative, effectively subdividing the task into two smaller tasks. It will be distinguished between the purpose of classifying data events and the purpose of classifying muon simulation events, i.e. giving a number of events to the model and observing if the model can identify events from the desired type correctly. Consequently, positive and negative are well defined for these tasks. This choice is arbitrary and is only used to arrange the definitions in a compatible way with equation 41 and 42 and does not change the result. It would be also possible to compute these values in a way without these additional definitions and just comparing the raw numbers which would yield the same result, e.g. computing all events with an MC score close to zero and considering the part of true data events in this area would give the precision for data events. The definition of strict correctly will be used as before as a classification for the precision and sensitivity metrics, because the difference between the strict and loose definitions is relatively small and the main aspect here concerns the falsely classified events with high certainty. The computed measures are shown in table 6 for the unshuffled test set for comparison and for the three parameters which lead to the huge accuracy drop. By investigating these metrics

	Data classification		MC muon classification	
Set	Precision	Sensitivity	Precision	Sensitivity
Unshuffled	99.9%	96.6%	99.9%	99.1%
Shuffled `tot`	59.1%	65.8%	60.5%	40.3%
Shuffled `pos_x`	51.9%	71.8%	52.5%	31.6%
Shuffled `pos_y`	53.6%	61.4%	54.0%	44.1%

Table 6: Precision and Sensitivity computed with first model's predictions on the test set for different shuffled features and separated into two subtasks in order to define positive and negative

further information can be obtained whether there are differences in the performance between data and simulation. It can be seen that for the unshuffled set both precisions, which could be interpreted as how good or accurate the positive prediction for the task is, are equally high. The sensitivities can be interpreted as a measure on how many events, that should be detected because their true value is positive, have been recognised for the given task. They are high for both cases but here a small difference appears between data and simulation events. Thus, data events are a bit less frequently identified compared to simulation events. For the shuffled features the precisions are nearly the same for both tasks with a maximal deviation of 1.4%. This means that by shuffling these features only around $50 - 60\%$ of the number of events whose predicted score reflects certainty - because it deviates only slightly from the desired values - is actually correct. For sensitivity there are differences between recognising data and simulation. When shuffled less than half of the simulation events are detected as such whereas for the data classification task $17\% - 40\%$ more events get detected. These

misidentifications are the main reason for the accuracy drop.
This investigation gives hints which features the model mainly uses to base its decision on. It was to be expected that ToT has a great influence because apparently it is not simulated in an optimal way [37]. However, these results should be interpreted with care for now because the main reason for the distinction may be based on the after-pulse pattern. While this indication should not remain for the shuffled feature, a pattern could still be detectable in the remaining features. Therefore, the second model will be analysed to verify that the importance of features is distributed in a similar way.

7.3 Second model

The second model will be also used to make predictions on the test set. Analysing the second model has the advantage of removed after-pulses in the training data and therefore other patterns need to be the cause of the distinction. Here, the maximal sequence length of the test set is again 314 but now in addition the after-pulses in the test set will also be discarded. The model was trained on data where after-pulses have been removed. In principle, the model should not be able to make use of them but keeping the after-pulses in the test set may cause unpredicted behaviour which would then be reflected by the performance. The amount of events in the test set remains the same but due to this procedure around $7.5 \cdot 10^5$ hits have been removed which counts for around 2.6% of all pulses. The prediction of the events in the test set are shown in figure 19. The accuracy with its strict definition amounts to 99.9%, similar to the validation set and better than the accuracy of the model from the previous training. The same procedure as before will be pursued to investigate the features which have the most relevance to the model's prediction by shuffling each feature individually and computing the accuracy respectively. Additionally, by using the same method it is easy to compare if this model, trained on unique hits, evaluates the chosen importance of the features in a similar way. The distribution of the three features with the largest accuracy drop - which are the same as for the other model - are again shown in figure 20, 21, and 22 while the other distributions can be found in the appendix in chapter A.2. The accuracies with both definitions are arranged in table 7. It can be seen that the features with the highest accuracy drop are still the same. Only the respective values changed. Both accuracies of shuffled ToT increased about 7% while the position

Shuffled feature	Accuracy (strict)	Accuracy (loose)
Unshuffled	99.9%	100.0%
`tot`	60.5%	65.0%
`t`	98.8%	99.8%
`pos_x`	49.9%	50.0%
`pos_y`	51.6%	51.8%
`pos_z`	90.6%	96.2%
`dir_x`	99.8%	100.0%
`dir_y`	99.9%	100.0%
`dir_z`	99.9%	100.0%

Table 7: Different accuracies (strict and loose) from second model's prediction on test set with unique hits for shuffling features individually

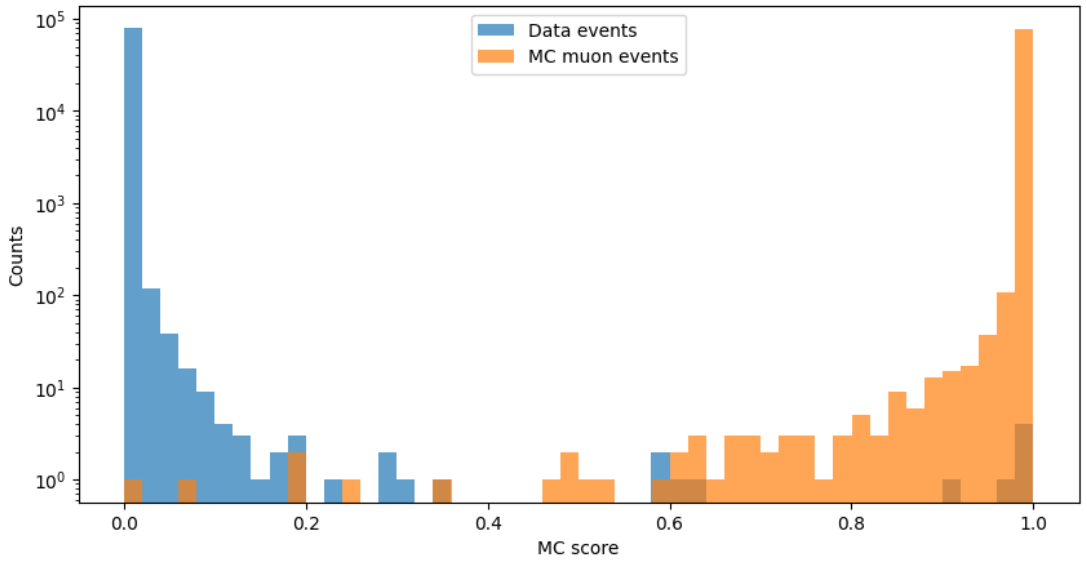


Figure 19: Prediction of test dataset with best model from second training where only unique hits were used

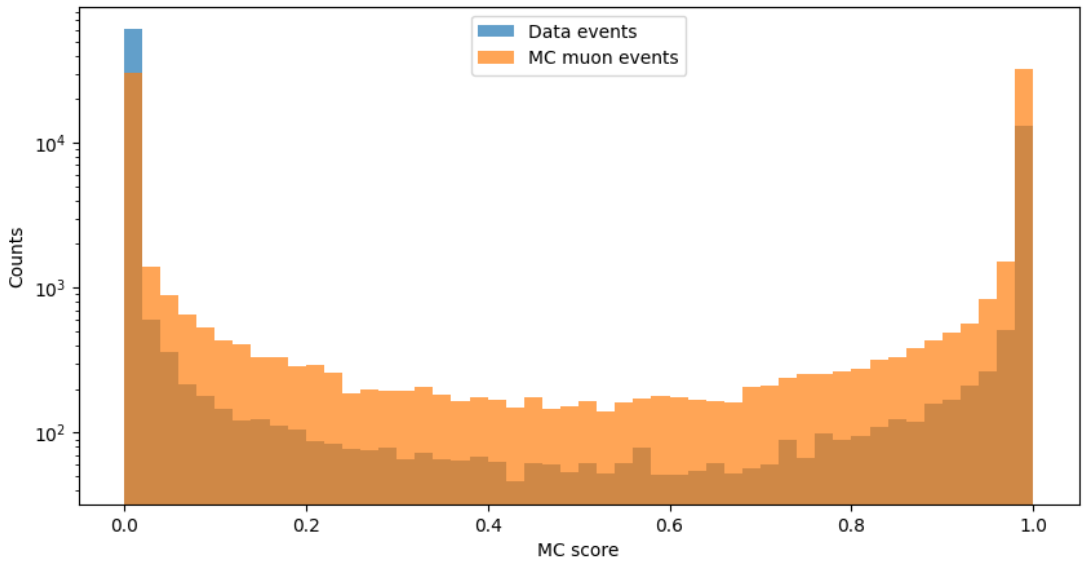


Figure 20: Distribution of second model's prediction on test set with unique hits where `tot` was shuffled

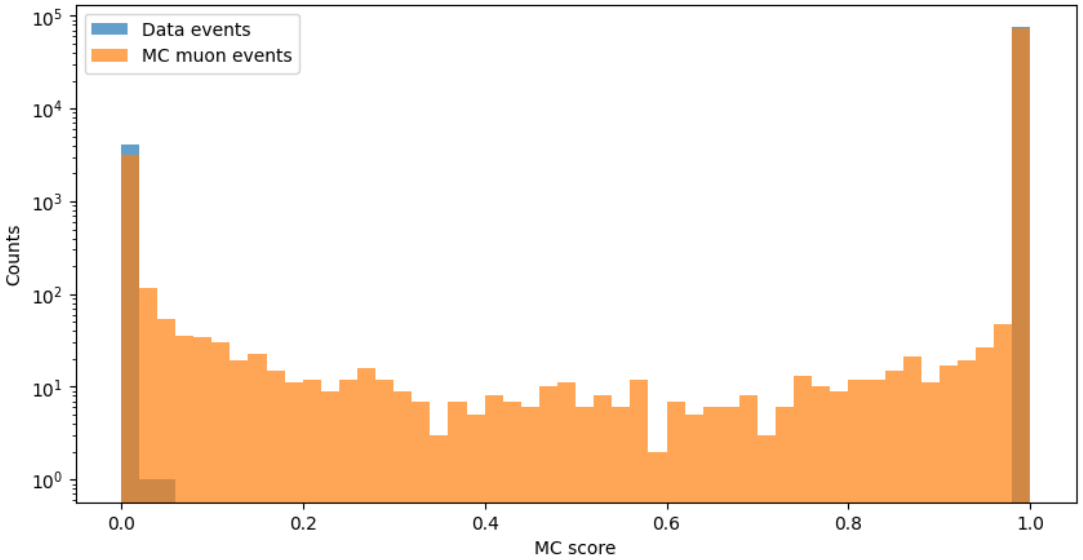


Figure 21: Distribution of second model's prediction on test set with unique hits where `pos_x` was shuffled

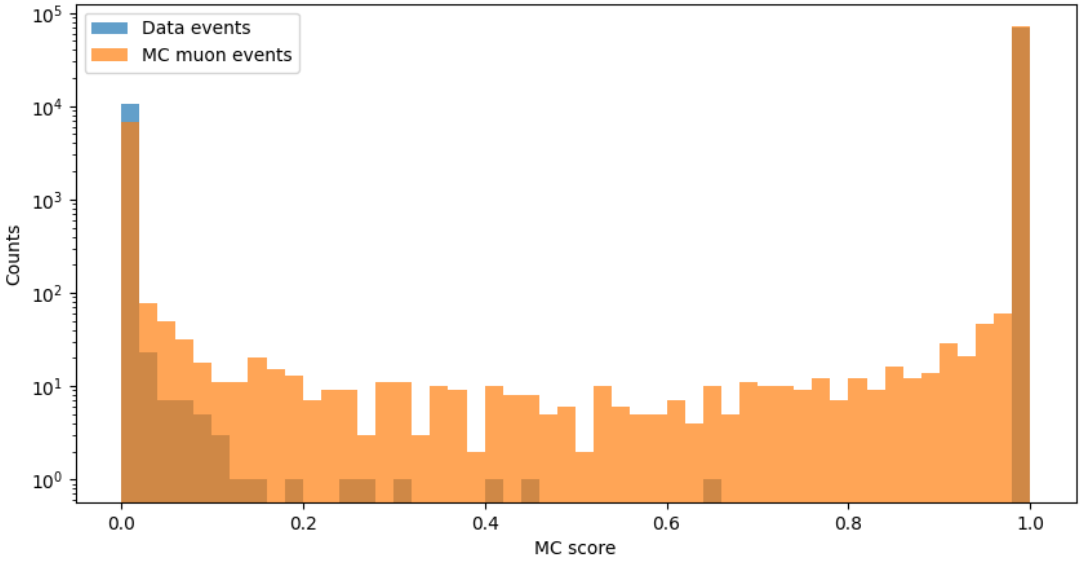


Figure 22: Distribution of second model's prediction on test set with unique hits where `pos_y` was shuffled

accuracies dropped around $1 - 2\%$. Small differences probably should not be given too much importance because the shuffling is still based on randomness. However, it can again be seen, when looking at the figures and the difference between loose and strict accuracy definition, that the spread out of the distribution is not the main cause of the loss of performance but rather the assignment of events to their opposite MC score. Therefore, the prediction and sensitivity metrics can be computed in the same way as before, and are shown in table 8 with the strictly correct definition from before. The measure of the predictions on the unshuffled set are at nearly 100% as

	Data classification		MC muon classification	
Set	Precision	Sensitivity	Precision	Sensitivity
Unshuffled	100.0%	99.9%	100.0%	99.8%
Shuffled `tot`	65.7%	77.3%	71.3%	43.5%
Shuffled `pos_x`	55.7%	5.2%	49.8%	95.1%
Shuffled `pos_y`	60.0%	13.1%	50.8%	90.5%

Table 8: Precision and Sensitivity computed with second model's predictions on the test set with unique hits for different shuffled features and separated into two subtasks in order to define positive and negative

expected. Here, the main reason why the accuracies drop on shuffled features is again the misclassification with high certainty, as also seen in the distributions. This is most prominently seen for the position x and y parameters. For them the interpretation of data events is false for almost all events where only 5% and 13% of these events get recognised as such. This imbalance can also be seen in the distribution but because of the logarithmic representation this exceptional effect is visually weakened. Finding a direct explanation for this phenomenon seems difficult. Because the shuffling is based on randomness, shuffling the same feature several times and viewing the statistics could give more insights whether this discrepancy persists to this degree for other configurations. It can be said that the same features as for the first model have the greatest influence on the prediction even though the distribution of accuracy drops changed. It should be noted that before shuffling no seed for the probability function used for shuffling was defined. Even if that would have been done the number of hits between the two different test sets used for each model differs, thus even the same seed would produce different results after shuffling. Therefore, the exact differences between the models for the different metric values should not be weighted too much because these likely also depend on the distribution of the hits after shuffling. Nevertheless, the sets have a large number of hits which should counteract these concerns.
While it was expected that shuffling important features produces performance changes, the misclassification of many events to the outside of the distributions was not directly obvious. A possible reason, which could explain why many events are classified as the opposite type with values close to zero or one respectively, when certain features are shuffled, could be based on the sigmoid function which is used in the last step in the Transformer. If the input values for this function are outside of the roughly defined interval $[-5, 5]$ the output values of the sigmoid function are extremely close to zero and one respectively, with deviations less than 0.1. When the Transformer receives random input, whose underlying patterns have not been seen before, in important features it may produce random output [37]. In this case the randomness would be achieved by

shuffling. This random output of the Transformer may not be located in the relatively small interval but within a much wider range which is responsible for the values in the middle regions of the MC score distribution. This explanation would be contrary to the human expectation of predictions having values around 0.5 when the Transformer deals with input data on which it should not be able to form a clear prediction. A further investigation could show the output distribution of the Transformer before the sigmoid function is applied.

7.4 Feature distributions

The result of the for the model important features can be further investigated by comparing the distributions of the respective feature between data and muon simulations. Two dimensional histograms of the position x and y features are shown in the appendix in section A.3 for data and muon respectively for the test set with unique hits. There, it can be seen that the distributions of the discrete coordinates seem more blurred for data events. To get a clearer view the distributions of `pos_x` and `pos_y` for the unique hit test set are shown in figure 23 and 24 where data and simulation are directly compared. The same amount of bins within the same range was used. Therefore, it can be seen

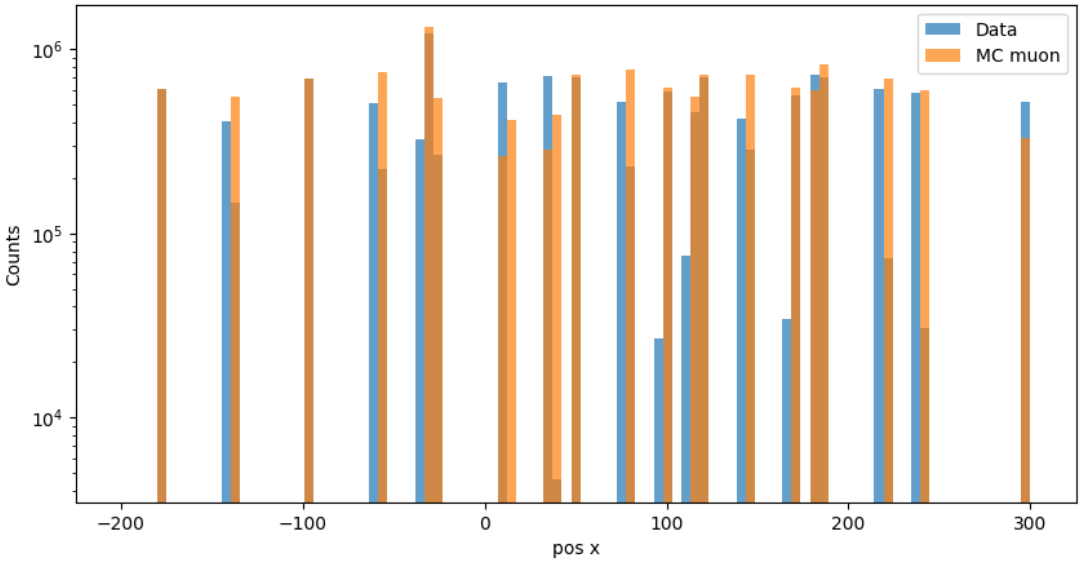


Figure 23: Distribution of `pos_x` for data and MC muon hits where the test set with unique hits has been used

that indeed each bin except one which is filled with simulation hits has a counterpart which is filled with data events - while their amount of course differs - but there are bins which are filled with data hits and in this corresponding value range no muon hits occur. By counting these specific hits one finds that their amount is 26% of all data hits for the position x feature and 3.5% of all data hits for the position y feature. This one bin in figure 23 in whose value range no data hits occur contains 3% of the simulation hits. While these can help the model to identify events easier it probably is not enough to account for everything. If these exceptions would be the only characteristic, one would not expect an accuracy decrease for shuffled ToT. The distributions for data

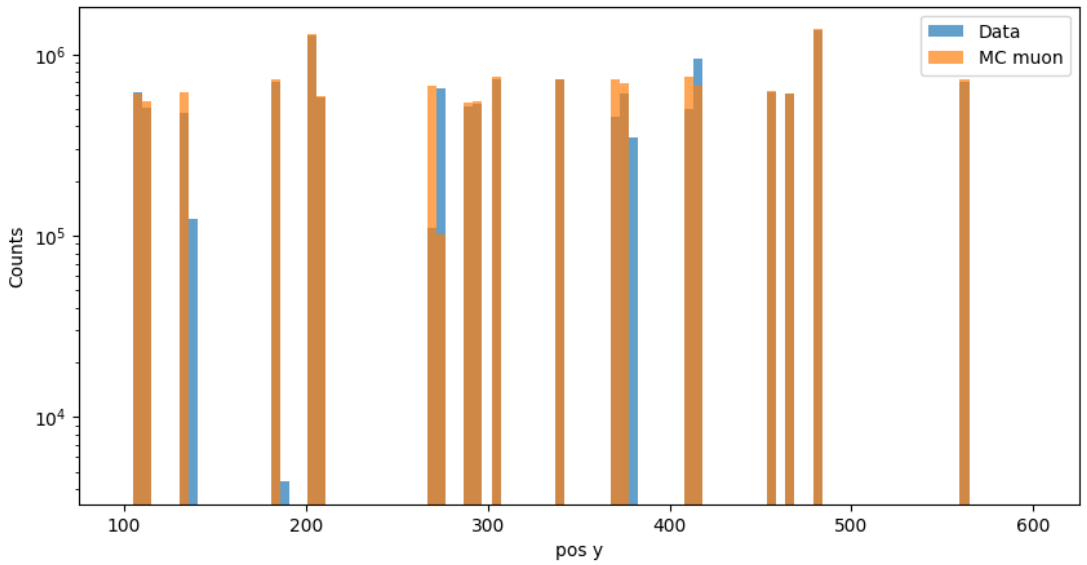


Figure 24: Distribution of `pos_y` for data and MC muon hits where the test set with unique hits has been used

and simulation of the ToT are shown in figure 25. Again, the same binning has been used to make better comparisons. It can be seen that for a certain value range of ToT only data hits occur. Unlike before the fraction is negligible because this characteristic applies to around 800 hits in total. Additionally, the bin with the lowest ToT value has a large discrepancy between the number of hits from data and simulation. Therefore, it would be reasonable if that value range is also an indication but all the muon simulation hits in this particular bin amount only to less than half a percent of all simulation hits. Thus, the directly obvious indication of ToT values being assigned to solely one hit type can not play a major role in the distinction process.
What is not directly observable are the connections between the different features. A first attempt in visualising these connections was done with the figures from chapter A.3 while there the only directly obvious characteristic was the mentioned "blurring". Therefore, a deeper structure may be the main cause which is not directly obvious from just looking at the feature distributions. On the other hand, there is a hypothesis which would be contrary to the one before. In particular, it could also be the case that the mentioned, special hits with feature values that no hits from the opposite type posses are favourably distributed over the events such that each event contains a few hits from these special value ranges. This could be definitely the case because the amount of special hits exceeds the number of events. However, this raises the question on whether the model would base its decision on only $\sim 25\%$ of the total hits. For the muon simulation events the fraction of special hits compared to the total hits only amounts to $\sim 3\%$ which seems to make a decision solely based on this small number unlikely. Nevertheless, the model could check if special values of data hits occur in the event and if not could be certain that it must be a simulation in this idealised view of favourable distribution of special data hits. Still, these suggestions would not explain the accuracy drop when the ToT feature is shuffled because in this case the number of special hits is far too small. Additionally, it would not explain why there is nearly

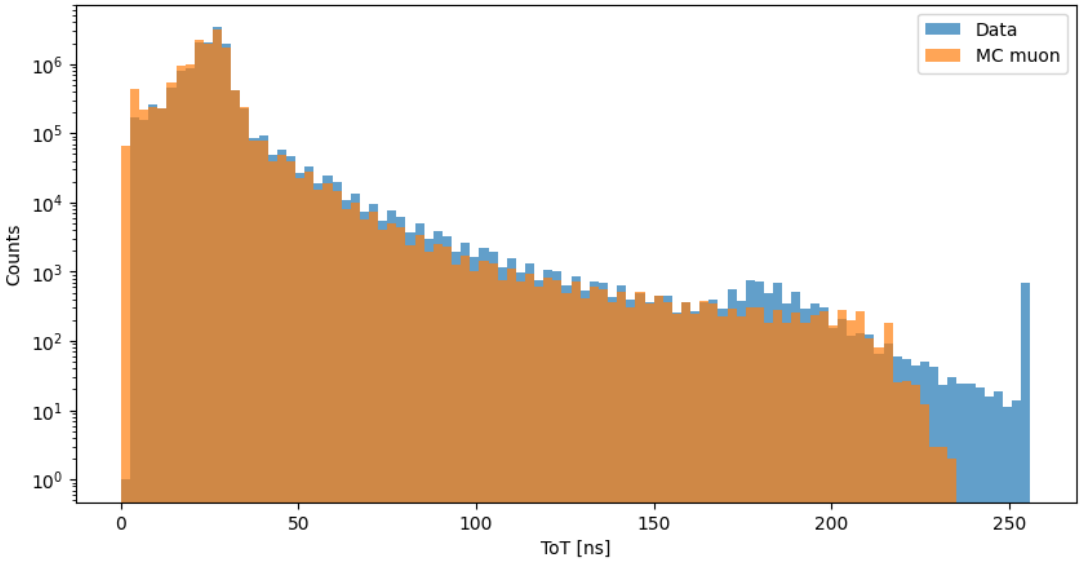


Figure 25: Distribution of `tot` for data and MC muon hits where the test set with unique hits has been used

equal accuracy drop for shuffling position y where the special hits make up a much smaller fraction. Therefore, the model could take these special hits into account, and thus could rely heavily on it as the accuracy drops suggest but there is probably more to its decision making than just this relatively simple idea.

8 Conclusion

8.1 Summary

This thesis introduced a procedure to use the Transformer from [39] on the task of extracting information from hits corresponding to a given number of experimental data and muon simulation events. Its input consists of all hits assigned to an event and to each hit eight features are assigned. The output consists of a continuos variable in the interval $[0, 1]$, the so called MC score, which is assigned to the event. A value of zero corresponds to a data event and one to a muon simulation event.
Firstly, the extractor was modified to get access to all needed parameters in the for machine learning suitable dataformat SQLite. A pool of 100 experimental data, muon simulation, and neutrino simulation files were converted. By applying constraints on certain parameters, which are assigned to the events, combined in the antinoise cut defined in [47] the initial number of available events was reduced to filter out noise events. The cut on the direction produced different results than initially expected due to the different convention used in the extractor, but the conclusions and observations made in later sections still hold true. The fraction of neutrino simulation events compared to the other events was negligible after the cuts were applied which is why the neutrino simulation events were not used in the further course. At the end of the preprocessing stage, ordered datastructures such as the dataloader have been assembled from the SQLite files which are compatible with the GraphNet workflow in which the Transformer is embedded. In order that this machine learning model can perform successfully on the given task, it needs to be trained. The data for the training process was chosen where 11 runs of data and MC muon simulation type have been selected which already contained a sufficient number of events which is in the order of $9 \cdot 10^5$. 75% of these events are used as the training set and the remaining 25% are used as the validation set. A maximal sequence length of the events was chosen while including nearly all events and at the same time saving resources. With the different components a GraphNet Standard Model was built and a first training was carried out. The accuracy of the first model applied to the validation set was 98.5%. To ensure that the cause of distinction between data and simulation is not based on the after-pulse pattern in data events a second training has been performed, where the same dataset has been used, but only the first hit on each PMT is considered. This removes the after-pulses, but hits from other sources may also be removed. The second model, which had an accuracy of 99.9% when applied to the validation set, performed even better than the first model. A possible reason could be that the training process contained more epochs than before. Therefore, another cause than after-pulses needs to be the reason for the successful distinction made by the model.
Both models have been analysed on the so called test set, which contained two data and two muon simulation runs. This additional set was used for analysing, because of easier handling of the input data compared to the validation set and liberating from biases because from the training the model with the best performance on the validation set has been chosen. Additionally, as a side effect the amount of events, on which the model is tested on, is increased. Then, it was attempted to find the input features, which are most important for the models' decisions. To achieve this, features have been shuffled and several metrics have been computed to quantify the performance changes, when features are shuffled. The shuffled features, which lead to the greatest accuracy drop,

have been identified to be `tot`, `pos_x`, and `pos_y`. It has also been found that when these features are shuffled the main cause of accuracy drops is the misclassification as events of the opposite type.
Lastly, the distributions of the features have been investigated and it has been found that there are special values of the features, which are only attained by one type. It has been proposed, that at least for the position x parameter a very efficient distribution of these special values on the events would be possible, and therefore the model would have an indication to which type the event belongs. However, this hypothesis does not explain the performance drops when position y and ToT are shuffled, because there the fraction of special hits is much smaller. Therefore, it is likely that these special hits play a role in the process of distinction, but it may not be the whole story.

8.2 Outlook

By looking at the Transformer and its results, a further investigation on why the shuffling affects the performance could be carried out. The approach could introduce a simple quantity for each event being the difference between the MC score before and after the shuffling. Then, outstanding events can be analysed further with large or small differences. It is expected that this will be a large part, because as seen before many events after shuffling are classified as extremes. For example, the events with the largest difference can be investigated. By doing so it can be seen, whether there are directly observable patterns in the hits, which have been redistributed to these particular events. The events with the smallest differences in the MC scores may be also worth considering. It may be, that events with a high difference in the MC scores received many hits from the opposite type, whereas events with small differences received hits from the same type. If this is the case, this would open up further questions, such as how many hits from the opposite type are needed to change the MC score.
An additional approach could be to generate events with hits, that only have values of the features, which were not realised in data and simulations. For the position variables this could be easy to implement, because there the distributions are discrete as seen before. The hits would not have any real meaning and it could be investigated, what the model predicts from these inputs. It could be expected, that the model gives a prediction which is around 0.5, but it could also be the case that many predictions with values close to zero or one are received, as seen in the distributions of the important, shuffled features. With this, the behaviour of the model when confronted with data on which a definite decision should not be possible can be further investigated.
Investigations can also be done on leaving out certain features in the training process, which is in principle easy to implement. It can be directly specified in the initialisation process of the datastructures and the model. As shown above, observations have been made that the direction features had no accuracy drops when they were shuffled. If the model would not have access to parameters such as ToT and position features, the question arises if it will perform badly and can no longer distinguish between data and MC muon simulations, or if there are hidden patterns in the direction variables, which now get the model's attention, apparently unlike before.

A Appendix

A.1 First model's prediction on test set with shuffled features

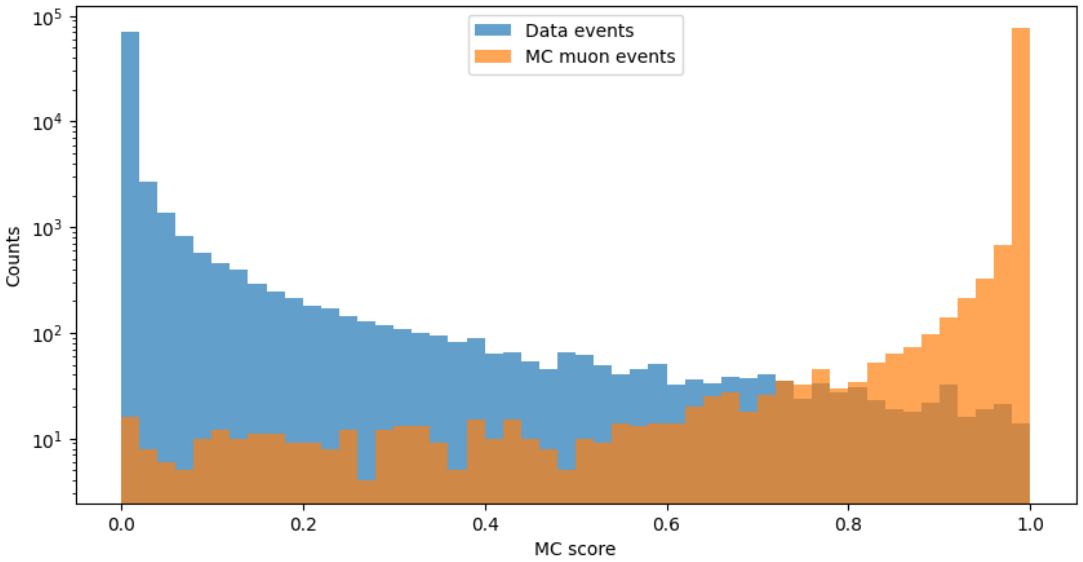


Figure 26: Distribution of first model's prediction on test set with shuffled `t`

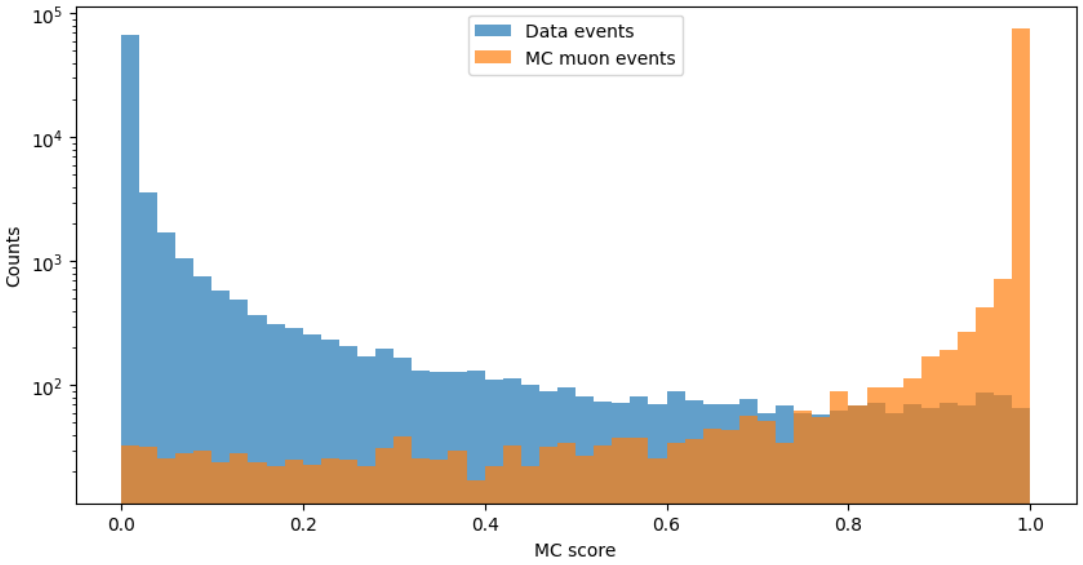


Figure 27: Distribution of first model's prediction on test set with shuffled `pos_z`

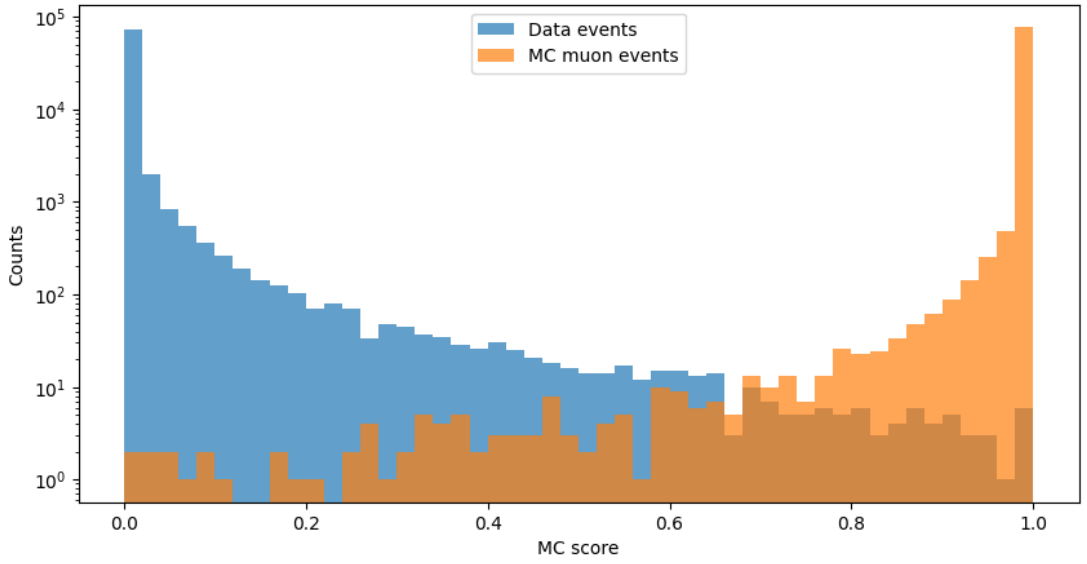


Figure 28: Distribution of first model's prediction on test set with shuffled `dir_x`

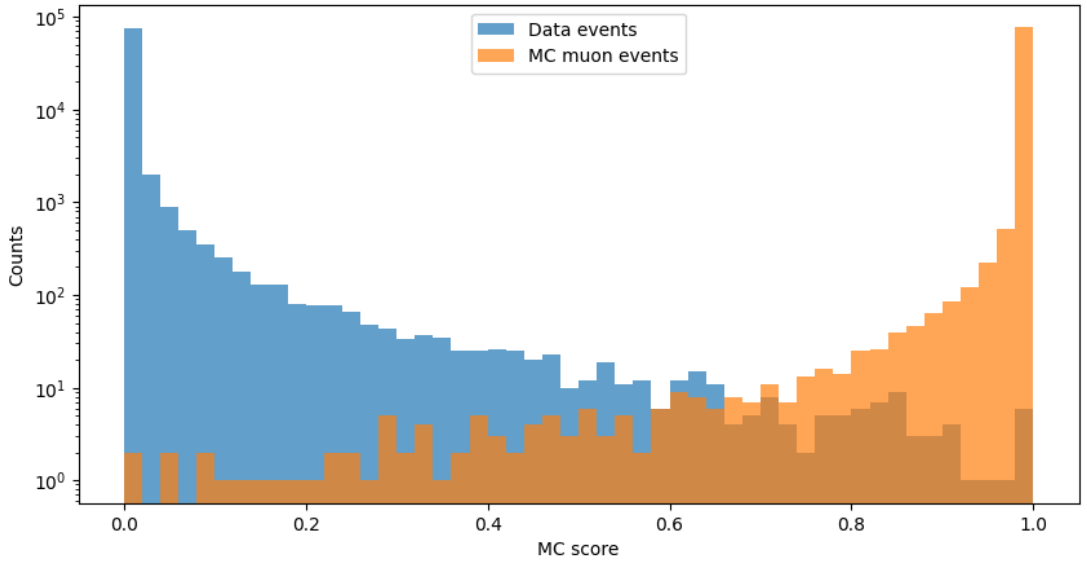


Figure 29: Distribution of first model's prediction on test set with shuffled `dir_y`

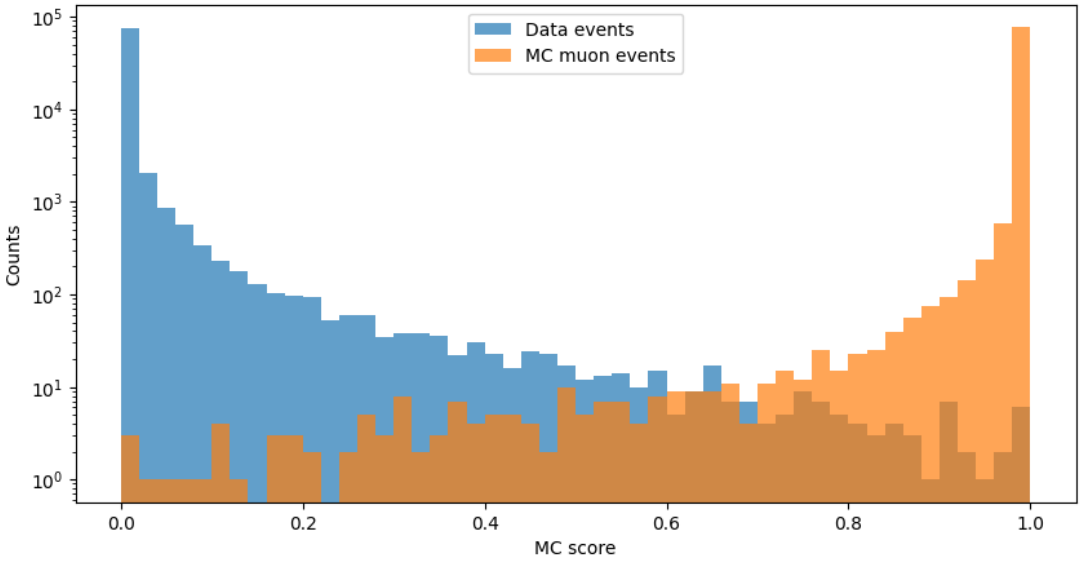


Figure 30: Distribution of first model's prediction on test set with shuffled `dir_z`

A.2 Second model's prediction on unique hits test set with shuffled features

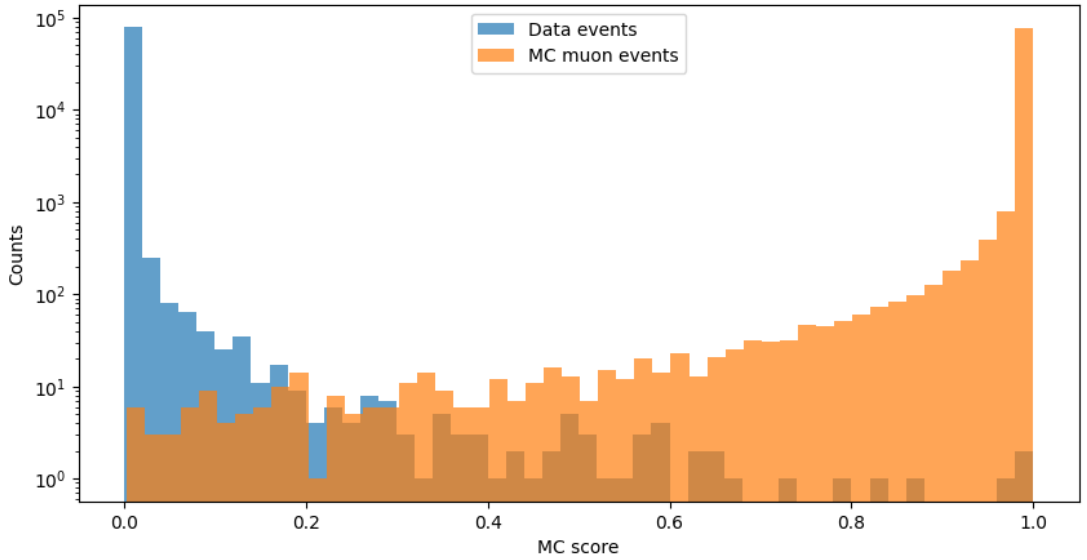


Figure 31: Distribution of second model's prediction on test set, containing unique hits, with shuffled `t`

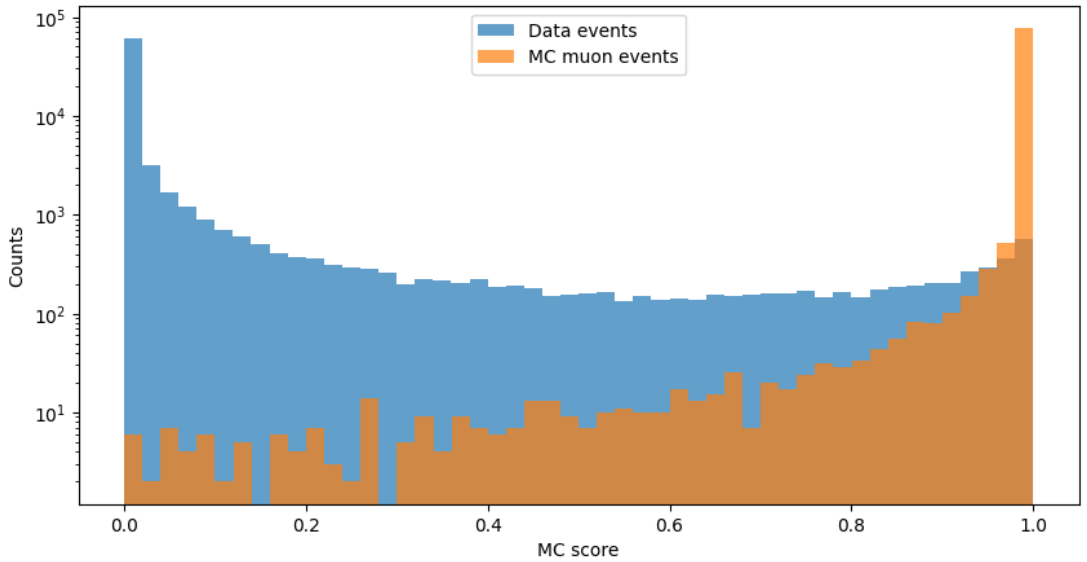


Figure 32: Distribution of first model's prediction on test set, containing unique hits, with shuffled `pos_z`

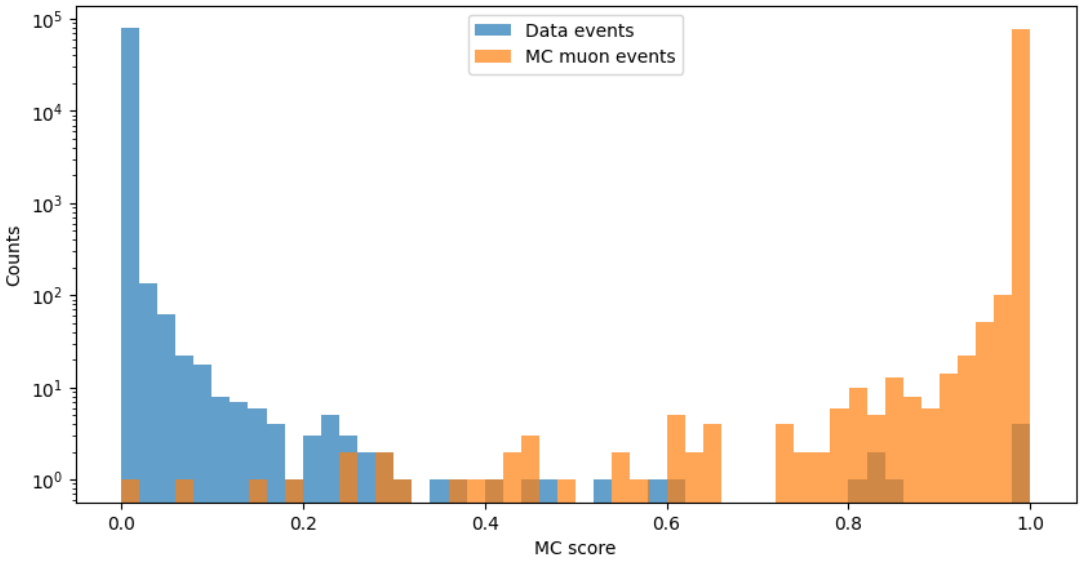


Figure 33: Distribution of first model's prediction on test set, containing unique hits, with shuffled `dir_x`

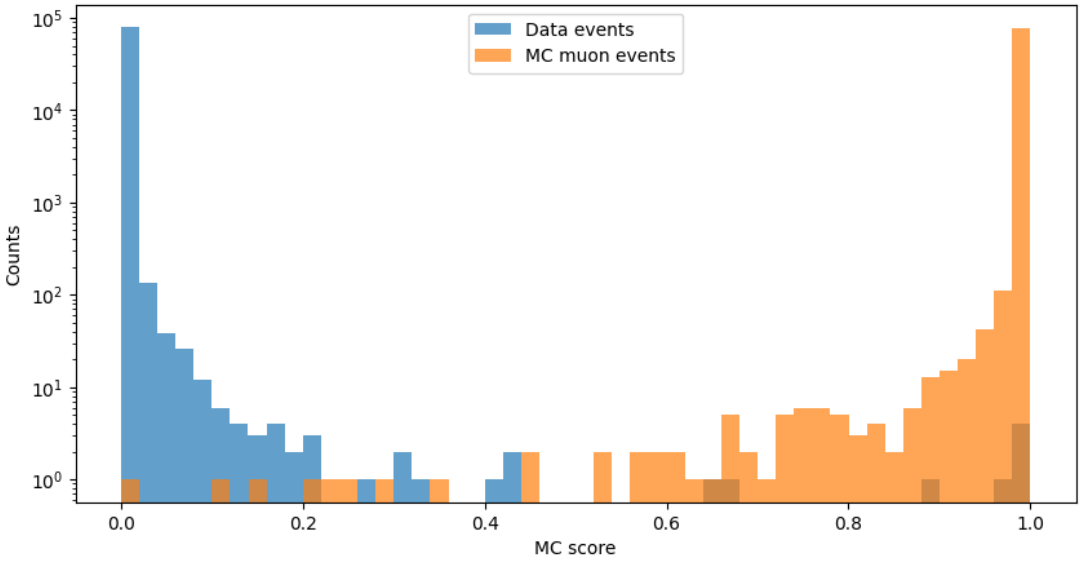


Figure 34: Distribution of first model's prediction on test set, containing unique hits, with shuffled `dir_y`

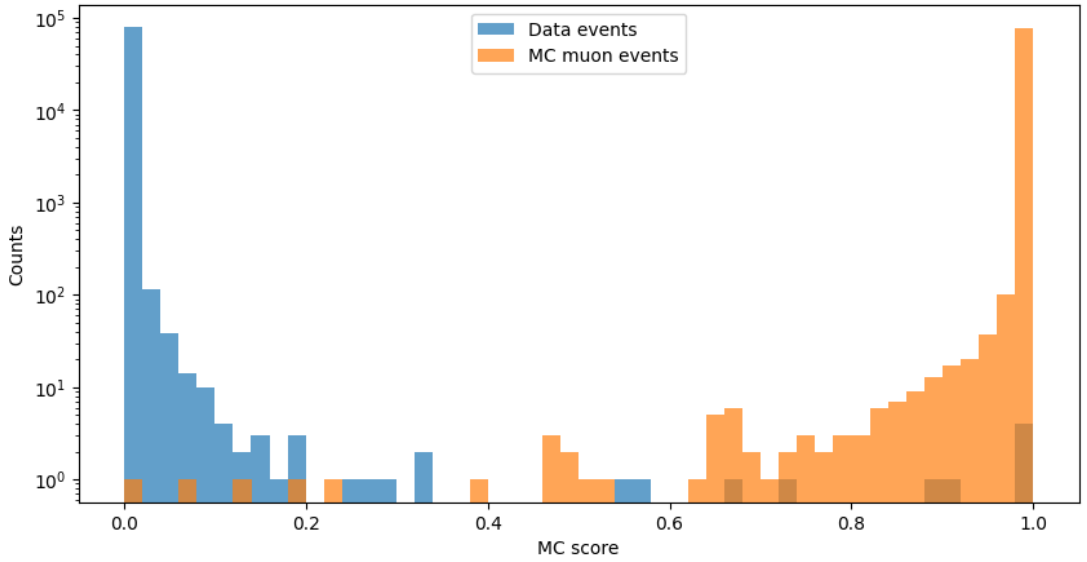


Figure 35: Distribution of first model's prediction on test set, containing unique hits, with shuffled `dir_z`

A.3 Feature distributions

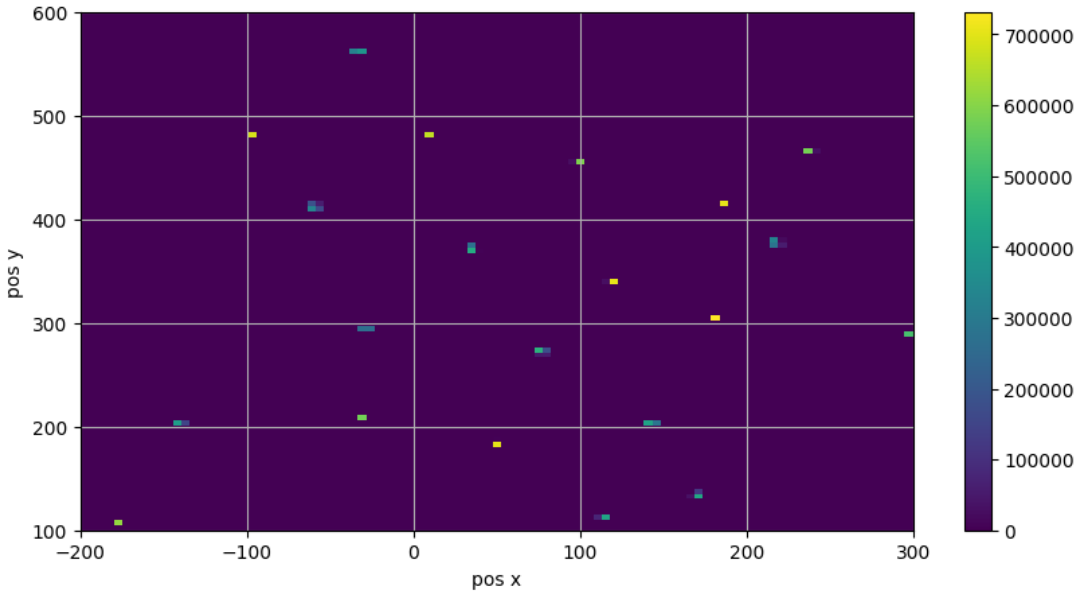


Figure 36: 2D histogram of position x and position y for all data hits in the test set with unique hits

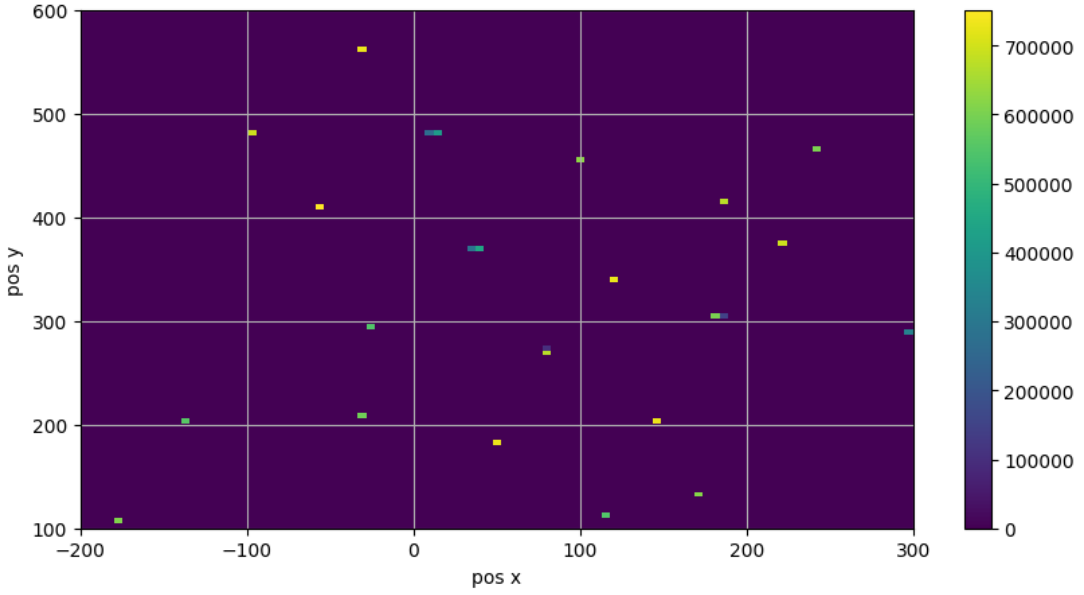


Figure 37: 2D histogram of position x and position y for all muon simulation hits in the test set with unique hits

Bibliography

[1] Steffen Hallmann. "Sensitivity to atmospheric tau-neutrino appearance and all-flavour search for neutrinos from the Fermi Bubbles with the deep-sea telescopes KM3NeT/ORCA and ANTARES". PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, 2021. eprint: https://ecap.nat.fau.de/wp-content/uploads/2021/02/Dissertation_HallmannSteffen_Opus.pdf.

[2] Christopher van Eldik. *Lecture notes on Advanced Experimental Physics: Particle and Astroparticle Physics*. 2025.

[3] Samoil Bilenky. *Introduction to the physics of massive and mixed neutrinos*. Springer Cham, 2018. DOI: 10.1007/978-3-319-74802-3.

[4] Mark Thomson. *Modern Particle Physics*. Cambridge University Press, 2013.

[5] Antonio Ereditato. *The State of the Art of Neutrino Physics*. WORLD SCIENTIFIC, 2018. DOI: 10.1142/10600. eprint: https://www.worldscientific.com/doi/pdf/10.1142/10600. URL: https://www.worldscientific.com/doi/abs/10.1142/10600.

[6] Kai Zuber. *Neutrino Physics*. Taylor & Francis, 2020. ISBN: 9781351764582. DOI: 10.1201/9781315195612. URL: https://directory.doabooks.org/handle/20.500.12854/158731.

[7] A.B. Balantekin and G.M. Fuller. "Neutrinos in cosmology and astrophysics". In: *Progress in Particle and Nuclear Physics* 71 (2013). Fundamental Symmetries in the Era of the LHC, pp. 162–166. ISSN: 0146-6410. DOI: https://doi.org/10.1016/j.ppnp.2013.03.008. URL: https://www.sciencedirect.com/science/article/pii/S0146641013000276.

[8] Vernon Barger, Danny Marfatia, and Kerry Whisnant. *The Physics of Neutrinos*. Princeton: Princeton University Press, 2013. ISBN: 9781400845590. DOI: doi:10.1515/9781400845590. URL: https://doi.org/10.1515/9781400845590.

[9] Brían Ó Fearraigh. "Following the light. Novel event reconstruction techniques for neutrino oscillation analyses in KM3NeT/ORCA". PhD thesis. Universiteit van Amsterdam, 2024. eprint: https://pure.uva.nl/ws/files/155195983/Thesis.pdf.

[10] J. A. Formaggio and G. P. Zeller. "From eV to EeV: Neutrino cross sections across energy scales". In: *Reviews of Modern Physics* 84.3 (2012), 1307–1341. ISSN: 1539-0756. DOI: 10.1103/revmodphys.84.1307. URL: http://dx.doi.org/10.1103/RevModPhys.84.1307.

[11] S Adrián-Martínez, M Ageron, F Aharonian, S Aiello, et al. "Letter of intent for KM3NeT 2.0". In: *Journal of Physics G: Nuclear and Particle Physics* 43.8 (2016), p. 084001. DOI: 10.1088/0954-3899/43/8/084001. URL: https://dx.doi.org/10.1088/0954-3899/43/8/084001.

[12] S. Aiello, A. Albert, A. R. Alhebsi, M. Alshamsi, et al. "Observation of an ultra-high-energy cosmic neutrino with KM3NeT". In: *Nature* 638.8050 (Feb.

2025), pp. 376–382. ISSN: 1476-4687. DOI: 10.1038/s41586-024-08543-1. URL: https://doi.org/10.1038/s41586-024-08543-1.

[13] Maurizio Spurio. *Probes of Multimessenger Astrophysics: Charged cosmic rays, neutrinos, γ-rays and gravitational waves*. Springer International Publishing, 2018. ISBN: 978-3-319-96854-4. DOI: 10.1007/978-3-319-96854-4_11. URL: https://doi.org/10.1007/978-3-319-96854-4_11.

[14] John David Jackson. *Klassische Elektrodynamik*. Berlin, Boston: De Gruyter, 2014. ISBN: 9783110334470. DOI: doi:10.1515/9783110334470. URL: https://doi.org/10.1515/9783110334470.

[15] Thomas Eberl. *Lecture notes on Neutrino Astronomy*.

[16] Jannik Hofestädt. "Measuring the neutrino mass hierarchy with the future KM3NeT/ORCA detector". PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, 2017. eprint: https://ecap.nat.fau.de/wp-content/uploads/2017/04/Dissertation_JHofestaedt.pdf.

[17] Wolfgang Demtröder. *Experimentalphysik 3. Atome, Moleküle und Festkörper*. 5th ed. Springer Spektrum Berlin, Heidelberg, 2016. ISBN: 978-3-662-49094-5. DOI: 10.1007/978-3-662-49094-5. URL: https://doi.org/10.1007/978-3-662-49094-5.

[18] Dr. Rodrigo Gracia-Ruiz. Private communication.

[19] Brían Ó Fearraigh and Bouke Jung. *Track reconstruction in KM3NeT*.

[20] Martin Schneider. *Measuring the attenuation length of seawater in KM3NeT/ORCA with atmospheric muons*. Master's thesis. https://ecap.nat.fau.de/wp-content/uploads/2022/07/martin_schneider_master_thesis.pdf. 2022.

[21] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series. MIT Press, 2012. ISBN: 9780262304320. URL: https://books.google.de/books?id=RC43AgAAQBAJ.

[22] Michael Moser. "Sensitivity studies on tau neutrino appearance with KM3NeT/ORCA using Deep Learning Techniques". PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, 2020. eprint: https://ecap.nat.fau.de/wp-content/uploads/2021/03/phd_thesis_michael_moser_v2.pdf.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[24] Grant Sanderson. *But what is a neural network?* https://www.3blue1brown.com/lessons/neural-networks. Last access: 10/10/2025. 2017.

[25] Izaak Neutelings. *Tikz.net: Neural networks*. https://tikz.net/neural_networks/. Last access: 11/10/2025.

[26] Markus Pirke. *Event reconstruction for ground-based γ-ray particle detectors with Transformer networks*. Master's thesis. https://ecap.nat.fau.de/wp-content/uploads/2024/11/2024_MA_MarkusPirke.pdf. 2024.

[27] Kenny Choo, Eliska Greplova, Mark H. Fischer, and Titus Neupert. *Machine Learning kompakt*. Springer Spektrum Wiesbaden, 2021. ISBN: 978-3-658-32268-7. DOI: https://doi.org/10.1007/978-3-658-32268-7. URL: https://doi.org/10.1007/978-3-658-32268-7.

[28] *BCELoss*. https://docs.pytorch.org/docs/stable/generated/torch.nn.BCELoss.html#torch.nn.BCELoss. Last access: 11/10/2025.

[29] *Modified nuT_graphnet repository from Lukas Hennig, forked from Ivan Mozun-Mateo/nuT_graphnet*. https://git.km3net.de/lhennig/nut_graphnet. Last access: 11/10/2025.

[30] S.K. Nayar. "Neural Networks". In: *Monograph FPCV-5-4, First Principles of Computer Vision*. Columbia University, New York, 2025.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[32] Richard E. Turner. *An Introduction to Transformers*. 2024. arXiv: 2304.10557 [cs.LG]. URL: https://arxiv.org/abs/2304.10557.

[33] Avinash Kak and Charles Bouman. *Transformers: Learning with Purely Attention Based Networks. Lecture Notes on Deep Leaning*. Purdue University. 2025. URL: https://engineering.purdue.edu/DeepLearn/pdf-kak/Transformers.pdf.

[34] *The km3io Python package*. URL: https://git.km3net.de/km3py/km3io.

[35] *Awkward Array documentation*. Last access: 1/11/2025. URL: https://awkward-array.org/doc/main/index.html.

[36] *NumPy. The fundamental package for scientific computing with Python*. Last access: 1/11/2025. URL: https://numpy.org/.

[37] Lukas Hennig. Private communication.

[38] Ivan Mozún Mateo and Antonin Vachere. *nuT: Neutrino Reconstruction with Transformers*. 2025.

[39] Ivan Mozún Mateo. *nuT_graphnet repository*. URL: https://git.km3net.de/imozunmateo/nut_graphnet.

[40] *GraphNeT - A Deep Learning Library for Neutrino Telescopes. Documentation*. URL: https://graphnet-team.github.io/graphnet/index.html.

[41] *PyTorch documentation*. Last access: 28/10/2025. URL: https://docs.pytorch.org/docs/stable/index.html.

[42] *PyTorch Lightning documentation*. Last access: 28/10/2025. URL: https://lightning.ai/docs/pytorch/stable/.

[43] *pandas package*. Last access: 3/11/2025. URL: https://pandas.pydata.org/.

[44] *The KM3NeT data format*. URL: https://git.km3net.de/common/km3net-dataformat.

[45] *Snakemake documentation*. Last access: 1/10/2025. URL: https://snakemake.readthedocs.io/en/stable/.

[46] *Slurm documentation*. Last access: 5/11/2025. URL: https://slurm.schedmd.com.

[47] Vittorio Parisi. *Data/MC comparison in the point-like analysis*. 2025.

[48] *Monte Carlo particle numbering scheme*. Last access: 24/10/2025. URL: https://pdg.lbl.gov/2007/reviews/montecarlorpp.pdf.

[49] PD Dr. Thomas Eberl. Private communication.

Acknowledgements

Many thanks to PD Dr. Thomas Eberl, Lukas Hennig, and Dr. Rodrigo Gracia-Ruiz for great support with code and software, many helpful discussions, answering many questions of mine, and reviewing this thesis.

Declaration of Originality

I, Julian van Laak, student registration number: 23124913, hereby confirm that I completed the submitted work independently and without the unauthorized assistance of third parties and without the use of undisclosed and, in particular, unauthorized aids. This work has not been previously submitted in its current form or in a similar form to any other examination authorities and has not been accepted as part of an examination by any other examination authority.

Where the wording has been taken from other people's work or ideas, this has been properly acknowledged and referenced. This also applies to drawings, sketches, diagrams and sources from the Internet.

In particular, I am aware that the use of artificial intelligence is forbidden unless its use an aid has been expressly permitted by the examiner. This applies in particular to chatbots (especially ChatGPT) and such programs in general that can complete the tasks of the examination or parts thereof on my behalf.

Any infringements of the above rules constitute fraud or attempted fraud and shall lead to the examination being graded "fail" ("nicht bestanden").

Place, Date	Signature