Flavor-agnostic energy-reconstruction using GNNs on IceCube neutrino data

Bachelorarbeit aus der Physik

Vorgelegt von
Marvin Hoffmann
31. Oktober 2024

Erlangen Centre for Astroparticle Physics
Lehrstuhl für Experimentelle Astroteilchenphysik
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Prof. Dr. Claudio Kopper
Betreuer: Dr. Christian Haack

Abstract

The IceCube Neutrino Observatory is cubic kilometer neutrino detector, located at the South Pole. This enormous detector consists of optical sensors arranged in an approximately hexagonal grid. Reconstruction of events is a crucial step in the analysis of neutrino events. The irregular sensor geometry and variable optical properties of the ice complicate potential reconstructions. Traditional event reconstruction methods are often specialized in the neutrino flavor and require multiple stages of background filtering to handle the abundance of noise from cosmic rays. By representing an IceCube event as a point cloud, one can apply graph neural networks (GNNs) for, e.g., reconstruction purposes.

In this thesis, a flavor-agnostic GNN trained on IceCube simulation data, Level2 data and below, is presented for deposited energies between 10 GeV to 10^8 GeV. The GNN was implemented using the GraphNet framework. The model can handle all three neutrino flavors at the same time, and could act as a filter without need for prior classification. The network performance was tested on different filtering levels and different event signatures. The flavor agnostic model displayed relative resolutions between 16% and 18% above $\sim 10^4$ GeV acrosss all event signatures, with worse performance below $\sim 10^4$ GeV. Better resolutions are possible when excluding events that passed none of the IceCube online filters or when considering only events that passed specific filters.

In general, the GNN model shows no biases concerning neutrino flavor, neutrino direction, or detector geometry and is comparable to other literature methods.

Furthermore, the model has a filtering rate of ~ 0.29 kHz on one GPU core and therefore has potential as a real-time IceCube filter to draw attention towards interesting astrophysical events.

Zusammenfassung

Das IceCube-Neutrino-Observatorium ist ein Kubikkilometer großer Neutrinodetektor am Südpol. Dieser riesige Detektor besteht aus optischen Sensoren, die in einem annähernd hexagonalen Gitter angeordnet sind. Die Rekonstruktion von Ereignissen ist ein entscheidender Schritt bei der Analyse von Neutrinoereignissen. Die unregelmäßige Geometrie der Sensoren und die variablen optischen Eigenschaften des Eises erschweren mögliche Rekonstruktionen. Traditionelle Rekonstruktionsmethoden sind oft auf die Neutrino-Flavors spezialisiert und erfordern mehrere Filterungs-Stufen, um das Rauschens der kosmischen Strahlung auszublenden. Indem IceCube-Ereignisse als Punktwolke dargestellt werden, können Graph Neural Netze (GNNs), z.B. für Rekonstruktionszwecke, eingesetzt werden. In dieser Arbeit wird ein flavor-agnostisches GNN vorgestellt, welches auf IceCube-Simulationsdaten, Level2 und darunter, für deponierte Energien zwischen 10 GeV und 10^8 GeV trainiert wurde.

Das GNN wurde unter Verwendung des GraphNet-Frameworks implementiert. Das Modell kann alle drei Neutrino-Flavors gleichzeitig verarbeiten und kann als Filter ohne vorherige Klassifizierung genutzt werden. Die Leistung des Netzwerks wurde mit verschiedenen Filterstufen und verschiedenen Ereignissignaturen getestet. Das flavor-agnostische Modell zeigte relative Auflösungen zwischen 16% und 18% über $\sim 10^4$ GeV für alle Ereignissignaturen, mit einer schlechteren Auflösung unterhalb von $\sim 10^4$ GeV. Bessere Auflösungen sind möglich, wenn man Ereignisse ausschließt, die keinen der IceCube-Online-Filter passiert haben, oder wenn nur Ereignisse berücksichtigt werden, die bestimmte Filter durchlaufen haben. Das GNN-Modell zeigt keinen Bias in Bezug auf Neutrino-Flavor, Neutrino-Richtung, oder die Detektorgeometrie und ist mit anderen Literaturmethoden vergleichbar. Mit einer Filterrate von ~ 0.29 kHz hat das Modell Potenzial für einen IceCube-Echtzeit-Filter.

Contents

1 Introduction

Neutrinos were first proposed by W. Pauli in 1930 [1] as a theoretical solution to preserve the conservation of energy and momentum during beta decays. It was not until 1956 that Clyde Cowan and Frederick Reines provided experimental proof of the (anti)neutrino's existence [2].

Neutrino astronomy is a field of astrophysics offering a unique perspective on the cosmos, because unlike protons or photons, neutrinos can travel vast distances across the universe without being deflected or absorbed by interstellar matter [3]. This property makes neutrinos invaluable messengers, thus capable of providing information about processes and environments that are otherwise inaccessible [3].

The quest to observe astrophysical neutrinos and deepen our knowledge about the Standard Model of particle physics (SM) encouraged efforts to design experiments capable of detecting these elusive particles [4]. However, detecting high energy astrophysical neutrinos turns out to be challenging due to their extremely low flux and small interaction cross section [4]. As a result, it became increasingly clear that large scale detectors are needed to achieve the necessary sensitivity [4].

As such, the IceCube neutrino observatory (IceCube), the first gigaton neutrino detector ever, was built into a cubic-kilometer of the clear glacial ice at Earth's South Pole [4]. IceCube is located at Amundsen-Scott South Pole Station, which offers the logistical support needed for the construction and operation of the observatory. Construction was completed on December 18, 2010, and commissioning was completed in 2011.

IceCube's primary scientific objective is the discovery of astrophysical neutrinos, reached in 2013, and the identification and characterization of their sources [4]. Capturing Cherenkov radiation emitted when neutrinos interact with the ice allows scientists to trace the origins of neutrinos back to their astrophysical sources [4]. This feature provides insights into the most energetic phenomena in the universe, such as supernovae, gamma-ray bursts, and active galactic nuclei [3].

Increasing the amount of data collected by IceCube can enhance statistical power. However, this approach faces diminishing returns: Doubling the data collection period from 15 to 30 years would improve the statistics by only a factor of $\sqrt{2}$. The limited improvement highlights the necessity for more sophisticated filtering and reconstruction techniques to achieve better sensitivity. Energy reconstruction in neutrino detection is crucial for analyzing neutrino sources accurately and understanding the physics of the detected events. Traditional methods often rely on specific assumptions about neutrino flavors, potentially introducing biases and limiting the applicability of the results or enhancing workload by having three reconstructions, one for each flavor.

To address these challenges, this thesis proposes the application of a neutrino flavor-agnostic energy reconstruction method using deep learning techniques, specifically graph neural networks (GNNs). By leveraging the structural information of neutrino interactions measured by IceCube's digital optical modules (DOMs), the GNN model aims to improve the precision and accuracy of energy reconstruction. This could pave

the way for the creation of more effective energy filters and enhance real-time detection and analysis of significant astronomical events.

The following chapters will go into detail about the theoretical foundation, methods, and results of this thesis.

2 Neutrino astronomy

Neutrinos are electrically neutral fermions without color charge that primarily interact via the weak interaction [3]. Due to their low mass, long thought to be zero, the gravitational force acting on the neutrinos is extremely weak. According to the Standard Model of particle physics, there are three flavors of neutrinos: Electron neutrino ν_e, muon neutrino ν_μ, and tau neutrino ν_τ. Since the weak interaction is a short ranged force, neutrinos usually pass through matter unhindered, which made them the last component of cosmic radiation to be detected [2]. Their low interaction probability makes them a useful messenger particle for cosmic phenomena but provides a challenge while trying to detect them [4].

2.1 Neutrino sources

Neutrinos can originate from a variety of sources [5]. Due to their stability and infrequent interactions, neutrinos are the most abundant component of cosmic radiation at ground level [2]. High-energy neutrinos are commonly produced alongside muons $\mu^\pm$, particularly through the two-body decays of charged pions, $\pi^\pm$, and kaons, $K^\pm$, as follows:

$$\begin{aligned} \pi^+/K^+ &\to \mu^+ + \nu_\mu \\ \pi^-/K^- &\to \mu^- + \bar{\nu}_\mu \,. \end{aligned} \tag{1}$$

Pions and Kaons can be generated through hadronic interactions [2].

Additionally, neutrinos are generated when a muon (antimuon) decays into an electron (positron), which is a process that is significant in the atmosphere, especially at lower energy levels [2, p. 126] and can be expressed as follows:

$$\begin{aligned} \mu^+ &\to e^+ + \nu_e + \bar{\nu}_\mu \\ \mu^- &\to e^- + \bar{\nu}_e + \nu_\mu \,. \end{aligned} \tag{2}$$

The flux of neutrinos on Earth as well as their associated energy range depend on their respective origin [6]. Figure 1 illustrates the energy spectrum of different neutrino sources as measured at Earth.

[5] describes possible sources of neutrinos in detail.

Cosmological neutrinos stem from the early stages of the Universe. They are also called cosmic neutrino background (CνB), like the cosmic microwave background (CMB) but with neutrinos. These neutrinos are extremely low in energy today, µeV to meV, because they have been cooled as the universe expanded over billions of years.

Solar neutrinos are released as byproducts of nuclear fusion reactions in the Sun, where hydrogen nuclei fuse to form helium. Their energy ranges from keV to $\sim$10 MeV.

Reactor neutrinos refer to neutrinos produced in man-made nuclear fusion power plants or nuclear fission reactions. They typically have energies of a few MeV.

Terrestrial neutrinos, also with a few MeV, on the other hand describe neutrinos that are produced by radioactive decay processes within the Earth's crust and mantle.

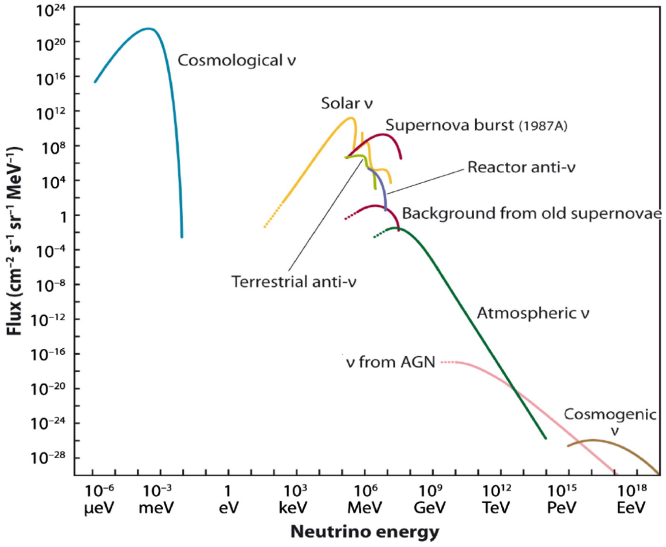


Figure 1: The spectrum of neutrinos as observed at Earth for different sources. Taken from [6].

Atmospheric neutrinos are generated when cosmic rays, i.e., high-energy protons or atomic nuclei, collide with atomic nuclei in Earth's atmosphere, producing secondary particles like pions and kaons. These unstable can mesons decay into muons and neutrinos as shown in Equation 1. Atmospheric neutrinos typically have energies ranging from a MeV to tens of TeV.

Astrophysical neutrinos is a term for all neutrinos originating from astrophysical objects. They typically have much higher energies, TeV to even EeV, than atmospheric neutrinos and can be traced back to phenomena like supernovae, gamma-ray bursts, and active galactic nuclei (AGNs).

Cosmogenic neutrinos, also known as ultra-high-energy neutrinos (UHE neutrinos), are a type of astrophysical neutrinos produced by the interactions of ultra-high-energy cosmic rays (UHECRs) with the CMB radiation (GZK effect [7], [8]). They are part of the highest-energy population of neutrinos, but have a low flux [5].

2.2 Relevant neutrino interactions

Elastic and quasi-elastic scattering describe interactions, where a neutrino scatters off an entire nucleon — potentially ejecting one or multiple nucleons from the target [9]. Resonance production, on the other hand, involves the neutrino exciting the target nucleon into a resonance state. This resonance state then decays into various mesonic final states. In contrast, deep inelastic scattering (DIS) describes an interaction where the neutrino has enough energy to scatter off a quark inside the nucleon [9].

As a result, DIS becomes the dominant neutrino interaction at higher energies, above about 10 GeV [10]. Since individual quarks cannot be directly detected and quickly recombine with other quarks, DIS always results in the formation of a hadronic shower

[9]. The resulting particle cascade, driven by the strong force, leads to the production of various particles, including nucleons, pions, kaons, and other mesons [9].

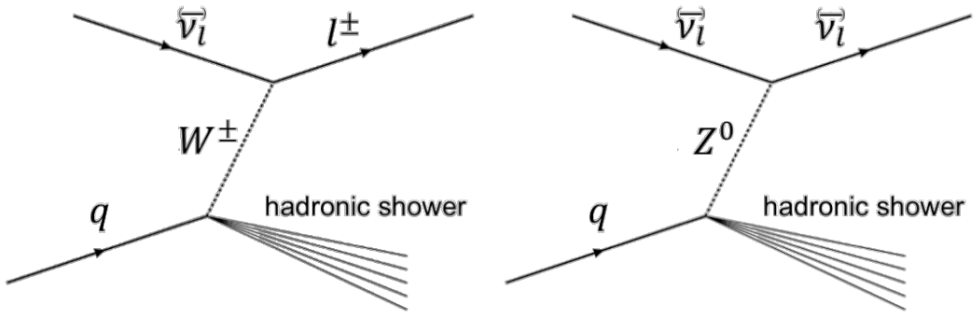


Figure 2: Feynman diagramm for DIS. An incoming neutrino interacts with a quark inside a nucleus. The interaction is either mediated by $W^{\pm}$ (left) or a Z^0 (right) exchange boson. Graphic taken from [11].

One can distinguish between two types of quark-neutrino interactions, depending on the exchange boson: Either a $W^{\pm}$ for charged current (CC), or a Z^0 for neutral current (NC) interactions, as illustrated in Figure 2. The different interaction channels are described in [12].

In CC interactions, the outgoing charged lepton corresponds to the neutrino's flavor and is emitted at the interaction vertex. A muon loses its energy gradually in the form of, e.g., bremsstrahlung while it travels through matter. In contrast, an electron loses its energy rapidly and triggers a localized electromagnetic shower. Electromagnetic showers are particle cascades, generated and driven by particles that interact primarily via the electromagnetic force, i.e. electrons or positrons. Unlike muons or electrons, tauons have a short lifetime and may decay quickly. Depending on the decay channel, the signature can vary.

Approximately 65% of the time, the tauon decays into hadrons, forming an additional hadronic shower. In about 17% of cases, the tauon decays into a muon and two neutrinos, resulting in a visible muon track similar to those in muon production, but with reduced energy due to the decay. Finally, around 18% of the time, the tauon decays into an electron and two neutrinos, creating an electromagnetic shower similar to electron production in direct CC interactions [12].

In the NC case, the initial neutrino survives and leaves only the hadronic shower as signal [4]. Since no lepton is created, it is impossible to reconstruct the neutrino's flavor from the interaction.

An overview of all the different interaction types, their respective end products as wells as an illustration of the end product's behaviour can be seen in Figure 3.

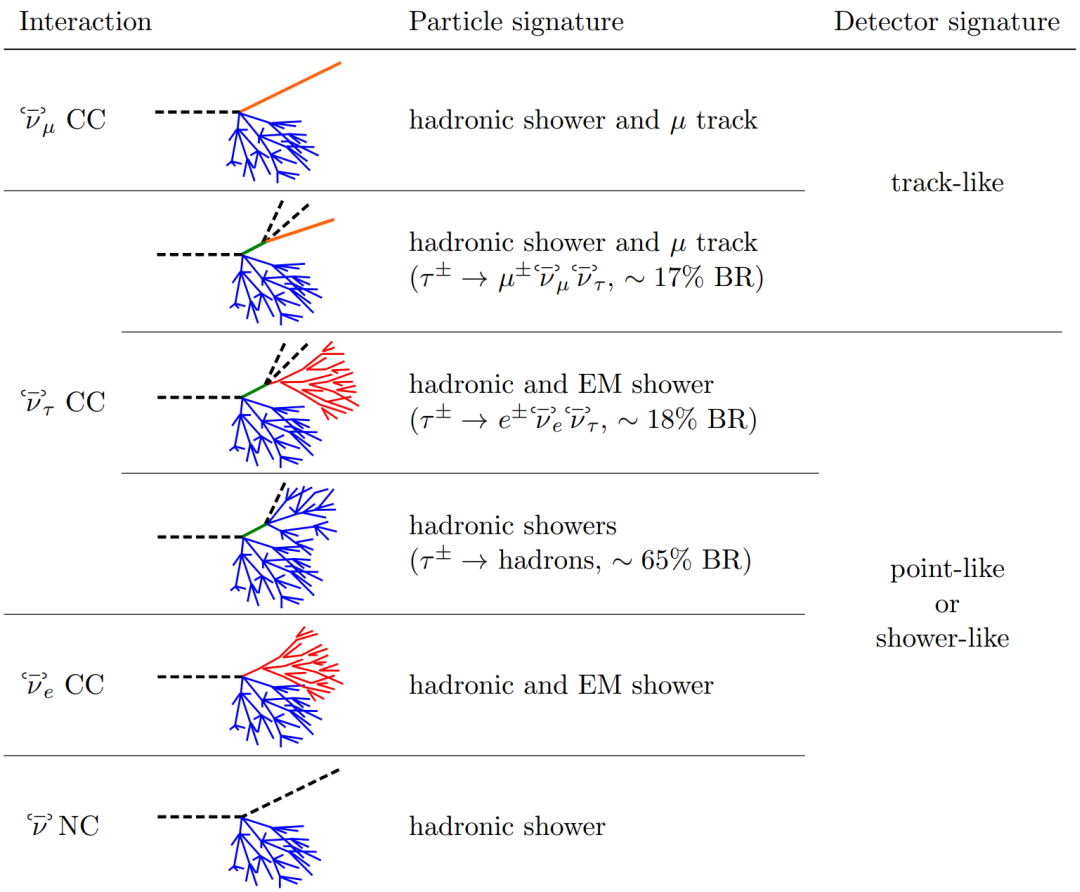


Figure 3: Overview and illustration of the possible DIS interactions. Black dashed lines are neutrinos, orange lines are muons, blue lines are particles in hadronic showers, red lines are electrons/positrons in electromagnetic showers and green lines are tauons. BR stands for branching ratio and characterises the probability of the different decay channels. For details on detector signature refer to section 3.2. Graphic taken from [13].

2.3 Detection principle

To detect neutrinos, one can make use of the phenomenon called Cherenkov radiation. When an electrically charged particle moves faster than the local phase velocity of light in a given dielectric medium, e.g., ice, Cherenkov photons will be emitted, which are usually in the visible to ultra violet spectrum [4]. The angle θ_C under which the photons are emitted can be calculated via

$$\cos\theta_C = \frac{1}{\beta n} = \frac{c'}{v}\,, \tag{3}$$

where $\beta = v/c$ is the ratio of particle velocity to the vacuum speed of light c [6]. n is the refractive index of the medium the light is passing through and $c' = c/n$ is the local speed of light within the medium. An illustration of this phenomenon can be seen in Figure 4.

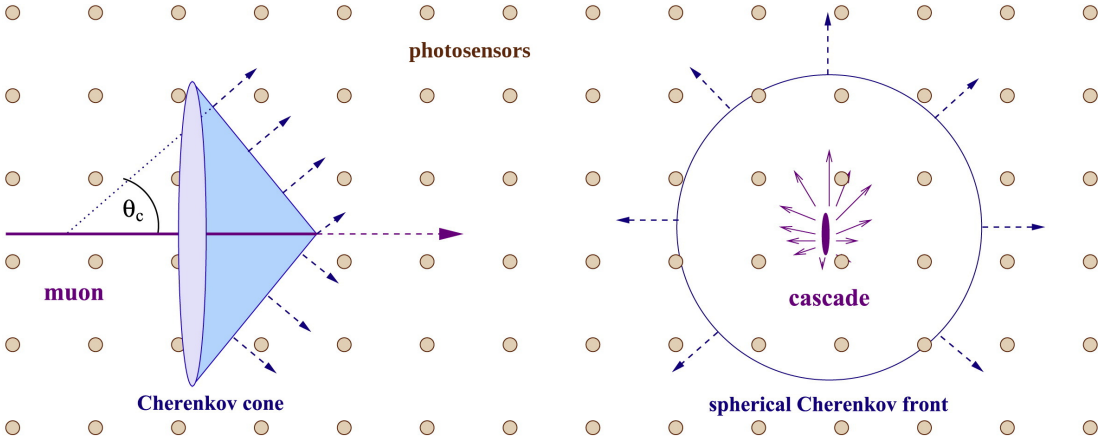


Figure 4: Detection principle of Cherenkov radiation for relativistic charged particles moving through a dielectric medium. Taken from [6], modified by [14].

Since neutrinos themselves are not electrically charged, it is imperative that they interact with the detector medium first, producing charged particles. While these secondary particles travel through the detector, Cherenkov radiation is emitted and measured by the detector.

3 The IceCube Neutrino Observatory

Due to the small interaction cross section of neutrinos with other particles and the low fluxes expected at Earth from astrophysical objects and events a vast detection volume is required. Relying on naturally occurring transparent media, like water or ice, is therefore economically favorable and an obvious choice to realize a neutrino detector project. The ice cap at the South Pole is about three kilometers thick and constitutes a suitable site, due to the large amounts of interaction material with good optical properties [4]. These properties are used by the IceCube neutrino observatory, or IceCube in short, to detect Cherenkov photons which is outlined in this chapter.

3.1 Detector design

IceCube is built into a cubic kilometer of antarctic glacial ice and is the first gigaton neutrino observatory ever constructed [4]. The detector can be split into three functional segments: The in-ice array, DeepCore,and IceTop, as can be seen in Figure 5.

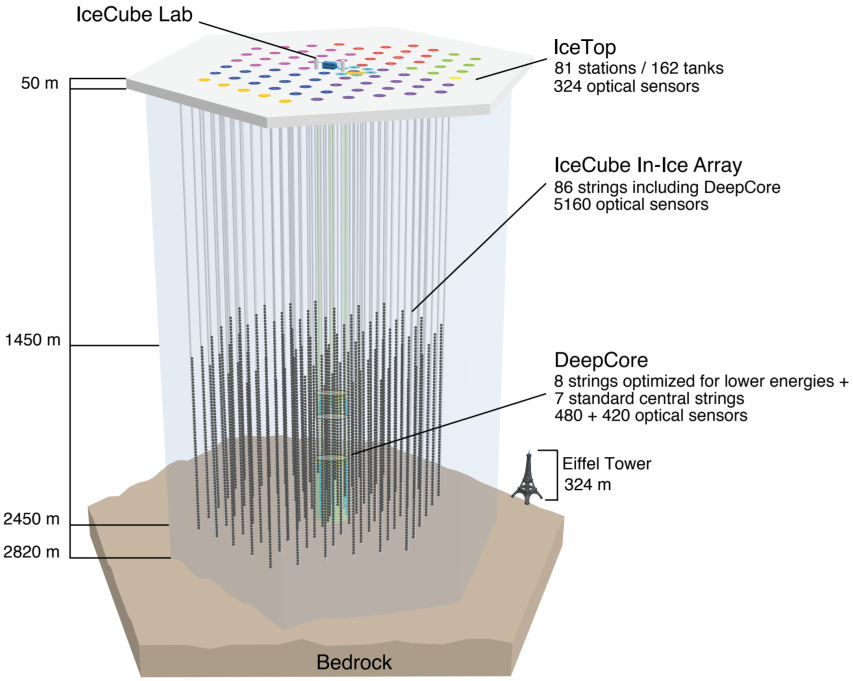


Figure 5: Illustration of the IceCube detector setup, with the Eiffel Tower as size reference. Taken from [4].

The in-ice component consists of 5160 digital optical modules (DOMs) [4]. A DOM is made of a thick, pressurized glass vessel to withstand the outside pressure of the ice and protect the inside electronics. An illustration of a DOM can be seen in Figure 6. Inside the DOM, a ten inch downwards facing photomultiplier tube (PMT) converts the incident Cherenkov light into electronic signals [4]. When a photon impinges onto a photomultiplier tube (PMT), a photoelectron can be produced with a certain

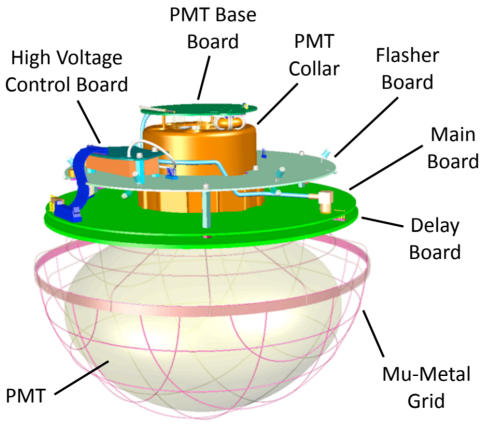


Figure 6: Illustration of a Digital Optical Module. The bottom part consists of a PMT, detecting incoming Cherenkov photons from neutrino interactions in the ice. The upper part contains the electronics, including waveform digitizers. Taken from [4].

probability called the quantum efficiency. This photoelectron is subsequently amplified, thus producing a measurable signal. Furthermore, a DOM includes electronic circuits to process and digitize the signal and LED flashers to calibrate and measure the local optical properties of the glacial ice [4].

Sixty DOMs are attached to one vertical string [4]. In total, 86 strings are deployed, each string in a water-filled borehole. These boreholes, spaced 125 m apart, freeze in place shortly after deployment. The DOMs are situated at a depth of 1450 m to 2450 m with a vertical separation of 17 m. This setup forms an approximately hexagonal grid over a cubic kilometer [4] [15]. With this spacing, the Cherenkov photon yield of $\mathcal{O}(10^5)$ visible photons per GeV of secondary particle shower energy, combined with the long optical attenuation length make it possible to effectively measure neutrino interactions [4].

Cherenkov photons traveling through the ice may scatter and be absorbed. These interactions impact the reconstruction performance of neutrino energy and direction. Thus, extensive studies, described in [16], [17] or [18], have been conducted to refine the current model of the optical properties of the glacial ice in the relevant depths. Among others, scattering and absorption coefficients have been measured and show that the ice has low impurity, apart from a region known as the dust layer at a depth of 2000 m to 2100 m, where the absorption length is unusually low.

Each neutrino interaction deposits a certain amount of energy into the ice. The only measurable quantity however are the Cherenkov photons. The Cherenkov photons, detected by the PMTs, are digitized using waveform digitizers [19]. These waveforms are processed through a procedure that aims to estimate the photon arrival times and the charge, referred to as pulses. These pulses represent the detector's response to an interaction, forming a time series. Each element in this sequence corresponds to the time t when the PMT readout registers a pulse with charge q [19]. The charge is measured in photoelectrons, where each photoelectron represents the most likely deposited charged

from a photon detected by the DOM [20]. The higher the charge, the more photons were detected, which suggests a more energetic event.

DeepCore is a densely instrumented subarray located in the bottom-center of the IceCube detector [4]. It consists of additional DOMs with higher quantum efficiency (sensitivity to incoming photons) and is designed to lower the energy threshold for neutrino detection. While IceCube is optimized for detecting neutrinos above 100 GeV, DeepCore focuses on neutrinos in the lower energy range between 10 GeV and 100 GeV. DeepCore as well as the in-ice array benefit from the natural shielding provided by the overlying ice, which reduces the amount of background noise from cosmic ray interactions.

IceTop, on the other hand, is a surface array of tanks filled with clear ice [4]. It is supposed to detect cosmic rays interacting with the Earth's atmosphere. Additionally, IceTop serves as a veto for background events in the IceCube detector, aiming to identify relevant neutrino interactions more accurately [4].

3.2 Event topologies in IceCube

The different interactions discussed in section 2.2 result in charged particles. When these particles travel through the glacial ice, Cherenkov photons are emitted and can be measured by IceCube's DOMs. Not every event is contained entirely inside the instrumented volume. The so-called level of containment describes how fully the event signature is captured by the detector. Depending on the interaction, one can distinguish between different event topologies. Generally, they can be divided into track-like and shower-like (or cascade-like), as illustrated in Figure 7.

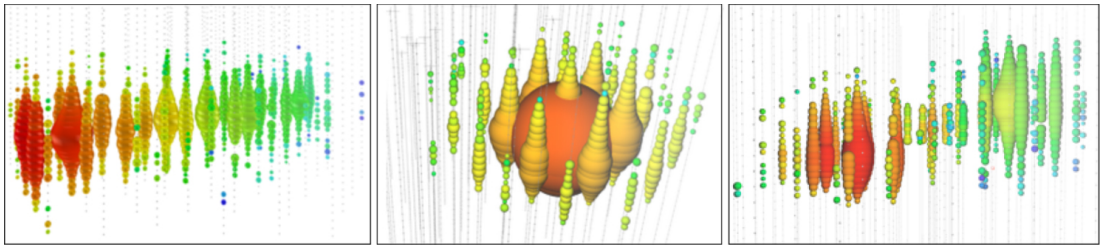


Figure 7: Example event topologies in IceCube. A track-like event (left), a shower-like event (middle) and a simulated double-bang event (two overlaying cascades as result of a tau decay) (right). The colored spheres show incoming Cherenkov photons at that point, while the sphere size indicates their amount. The colors themselves correspond to the relative arrival time of the photons, encoding earlier photons in red and later photons in blue. Taken from [21].

Track-like events

Ideally, a track would be visible as a straight line through the detector. However, the photons are scattered in the ice and consequently, the track is smeared as shown in the left illustration of Figure 7.

Tracks are a signature primarily originating from muons, either after a ν_μ^{CC} or a ν_τ^{CC} interaction, when the resulting tauon decays further into a muon [20].

The amount of Cherenkov light emitted along the muon path provides information about its energy, while the different photons arrival times, and the resulting signature orientation, are used to reconstruct the direction of the muon which allows to draw conclusions about the neutrino's origin.

At low energies $\lesssim 100\,\mathrm{GeV}$, all muon energy is deposited in the detector [20]. However, at higher energies, muons can travel long distances through the detector volume, often much longer than the dimensions of the detector itself, resulting in a track-like event that can span across a large part of the IceCube array [20]. Track events frequently extend beyond the detector volume, meaning the observed energy, or deposited energy, provides only a lower bound on the neutrino's true energy. At muon energies over $\approx 1\,\mathrm{TeV}$ stochastical energy losses due to e.g. bremsstahlung are dominant. As a result big differences in the observed energy are possible, even for muons with the same energy [20]. These event-to-event variations complicate the energy reconstruction. A directional reconstruction for track-like events, however, would be easier, since the muon direction is correlated to the neutrino direction [22].

Not all particles producing track like signatures are muons: (ν_τ^{CC}) interactions at energies above approximately $1\,\mathrm{PeV}$ can result in tauons that leave a track of direct electromagnetic losses and travel detectable distances before decaying [20].

Shower-like events

Showers form a more spherically extended pattern as shown in the middle illustration of Figure 7. This signature originates from particle cascades at the interaction vertex. Showers can be either electromagnetic or hadronic in nature.

The resulting hadronic showers for NC interactions show similar signature when compared to their electromagnetic counterpart, but with lower light yields [20]. The production of heavier particles with less kinetic energy is the main cause for the smaller amount of detected photons [23].

Due to the localized energy deposition, it is generally easier to reconstruct energy for showers than for tracks, whereas inferring information about the neutrino's direction turns out to be difficult [20].

A special type of shower-like event is shown in the right illustration of Figure 7. During a ν_τ^{CC} interaction, a hadronic cascade is formed (left shower in red) and a tauon is produced [24]. Due to its short lifetime, the tauon decays quickly, creating another shower close the first one (right shower in green). This particular event signature is called double bang. This signature forms two well separated showers at energies above $\sim 2 \times 10^6\,\mathrm{GeV}$ [24]. Instead of a hadronic shower this tauon could also decay into a muon and leave a detectable track like topology. Refer to Figure 3 for an abstracted illustration of the discussed processes.

Detectable Cherenkov light is radiated by a charged particle and any associated showers [20]. Less energetic neutrinos result in a lesser amount of charged particles and may therefore result in an emission of a lower amount of Cherenkov photons.

3.3 IceCube filtering

Detected tracks can also result from high-energy cosmic rays hitting the Earth's atmosphere and producing extensive air showers. These air showers may contain a variety of secondary particles, including muons and atmospheric neutrinos [25].

While electrons and, sub dominantly, tauons are also produced in air showers, electrons lose energy very quickly in the atmosphere and rarely arrive at the detector. The tauons also decay most likely before reaching the detector. Due to the high flux of cosmic rays, air shower muons and their tracks represent the overwhelming part of background events in IceCube [25].

However, only air showers from the southern hemisphere [26] and atmospheric neutrinos constitute a relevant background. Muons produced after an incident cosmic ray on the northern hemisphere would lose their energy through a variety interactions on the way through the Earth and decay before hitting the detector. In that case, the Earth acts as a shield for air showers in the northern hemisphere.

The background of cosmic-ray induced downgoing muons amounts to 10×10^6 muon events per neutrino event [26].

IceCube has an average trigger rate of about 2.7 kHz [4] and most of these detected events are background events such as muons produced in air showers. To sort out background events, likely noise signals and low quality signals, filters are used. These filters are algorithm that sift through large amounts of data and identify events which have certain predefined properties. Basically, filters can be classified into online filters and offline filters. Figure 8 shows the filtering process in a simplified illustration.

Online filters are applied directly at the observatory [27]. These filters are supposed to quickly reduce the massive amounts of raw data to be sent north via satellite, since the bandwidth is limited. This data reduction is achieved by dismissing clear background events, while trying to maintain a low false rejection rate. These online filters are designed to work in real time and have to sacrifice some precision and rely on less sophisticated algorithms to meet the time constraints. The online processing pipeline and filters are able to select any triggered events that pass established event quality checks [28]. These selection criteria searching for single events are established and verified in offline studies. The IceCube alert system relies heavily on these online filters to identify coincident astrophysical neutrino events and potentially trigger a multi wavelength follow-up [28]. Rejected events are stored physically for future analysis.

Offline filters run on already transferred data. They are designed for more detailed event reconstruction and analysis. Since offline filters are not subjected to the strict time and bandwidth limitations, they may use more computationally intensive algorithms than online filters.

During the filtering process the data can be divided into Levels. These Levels denote increasing complexity of the applied algorithms. Level0 describes the raw data, while Level1 refers to basic noise reduction and the application of basic reconstruction algorithms. Most relevant for this work is Level2. This stage refers to the point in the data processing pipeline, where more advanced reconstruction algorithms are applied. The data may still be considered preliminary but has a better signal to noise ratio when

compared to Level0 data. A simplified overview of the filtering process can be seen in Figure 8.

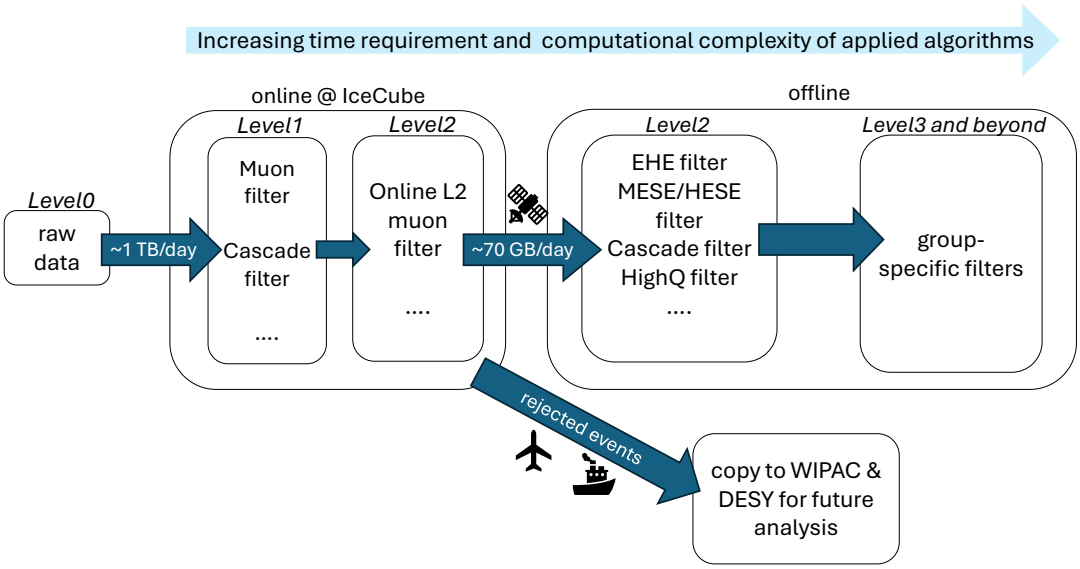


Figure 8: Simplified flowchart for the filtering process in IceCube. Raw data is first reduced via the muon filter and several other Level1 filters before being passed on to the Online Level2 filtering systems. A fraction of the data is sent north through satellite, where more sophisticated algorithms can be applied. Events that were rejected at the online stages are stored in disks and transported north once a year.

The event selection starts with the (Level1) muon filter, that is supposed to identify high-quality muon tracks [29]. Since the main background in IceCube is from air showers, this filter aims to identify background muons based on their track-like signatures and arrival direction to reject or analyze them separately from neutrino events. The muon filter has a filter rate of $\sim$30 Hz [29].

The cascade filter is used to detect shower-like events [30], by quantifying the spherical properties of the events signature. The filter rate of the cascade filter is $\sim$30 Hz [30].

The Online Level2 muon filter is applied to a small subset of promising muon events directly at the South Pole. This online filter uses the the result from the first muon filter to further reduce background contamination [31]. At the end of that step, the rate of outgoing events is $\sim$ 5 Hz [28].

The HESE (High Energy Starting Event Selection) and MESE (Medium Energy Starting Event Selection) filter aim to select events, where the interaction vertex is inside the detector volume [32]. Especially HESE plays an important role in the alert system [27]. The filter rate of MESE is at about 11 Hz [32]. The HESE filter has a really low filter rate of $\sim$10 mHz [32].

The HighQ filter, formerly know as the extremely high energy, or EHE, filter is designed to catch events that produce more than 1000 photoelectrons [33]. The filter

was based on an offline search for cosmogenic neutrinos and is now included to the online alert system [27]. The HighQ filter has a filter rate of $\sim$1 Hz [29].

Real-time filtering is achieved by distributing each detected event to one of approximately 400 dedicated filter clients [27].

Due to the limited data transfer quota via satellite, only essential pulse data is transferred north [23]. The properties that are calculated by the filters would constitute additional data and are computed offline again after the transfer.

3.4 Event simulation

To understand the expected signals and background noise, it is essential for IceCube to simulate events and to model how neutrinos interact with the detector.

The simulation process involves generating theoretical neutrinos with energy sampled from a spectrum, which is usually done with a program called NuGen, and then forcing them to interact by artificially raising the interaction probability.

SimWeights [34] is a python package that is used to ensure that the simulated data accurately represents real detector conditions by calculating simulation weights, or sim weights in short. These sim weights corrects for the falsely assumed spectrum in the simulation, since usually a less steep spectrum is used to generate more high energy neutrinos. In addition to that, sim weights accounts for any modified physics, like the enhanced interaction cross sections, made during simulation. In analysis, it can then be used as weights in histograms to reproduce the physically expected circumstances.

After the interaction, the resulting hadrons, leptons and, photons are propagated, while considering stochastic losses and the complex optical properties of the glacial ice.

Since the flux of atmospheric muons is six orders of magnitude greater than that of astrophysical neutrinos, some muons will statistically coincide with neutrino events in the detector. This overlap makes it difficult to separate the events, which is why muon events need to be simulated as well.

Lastly, the detector response has to be simulated, including PMT noise and response as well as the electronics that convert the analog signals into digitized events. A flowchart illustrating the process can be seen in Figure 9.

After the simulation pipeline, the events are treated like real data and filters as described in section 3.3 are applied. For a more in detailed description, see [35].

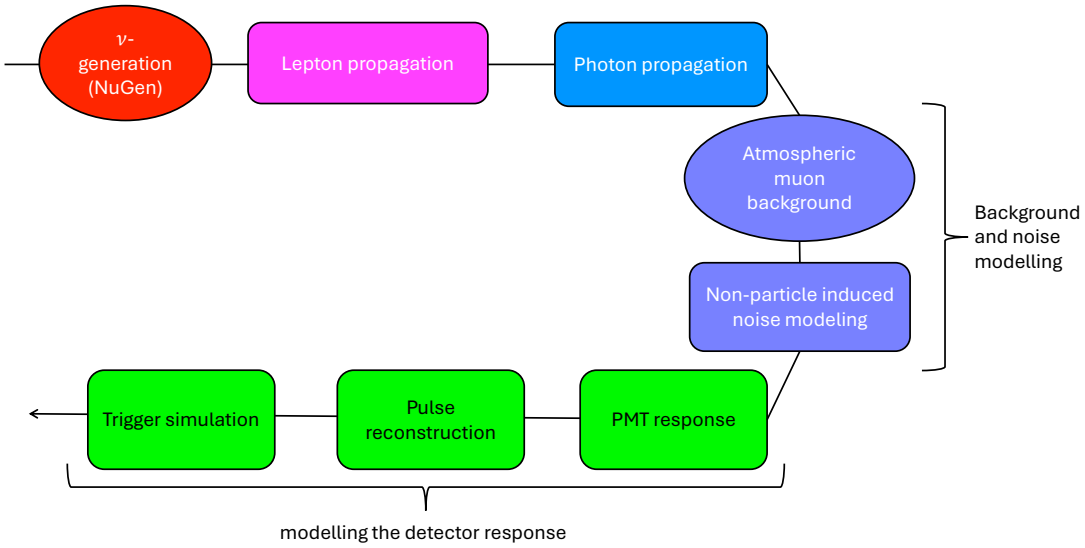


Figure 9: A schematic flowchart showing the individual steps in the simulation chain. Inspired by [35].

4 Machine Learning and Graph Neural Networks

In recent years, the field of machine learning (ML) has revolutionized the way we process and interpret complex data. ML encompasses a range of algorithms that enable systems to learn patterns and make decisions based on data, without being explicitly programmed for specific tasks. Unless explicitly stated otherwise, the information for section 4.1 and section 4.4 is from [36]. The information in section 4.2 is largely taken from [37]. Similarly, the information about graph neural networks (GNNs) in section 4.3 stems from [38], unless state otherwise.

4.1 General overview of Machine learning

At its core, machine learning can be divided into three primary categories: Supervised learning, unsupervised learning, and reinforcement learning [36].

Supervised learning involves training models on labeled datasets, where the desired output is known, allowing the model to learn from examples. Unsupervised learning, on the other hand, deals with unlabeled data, where the goal is to find hidden patterns or structures in the data. Lastly, reinforcement learning lets the algorithm interact with an environment and make decisions. Based on that, the algorithm receives feedback in the form of rewards or penalties, improving its strategy over time to maximize received rewards. The approach followed in this thesis is that of supervised learning.

The development of deep learning (DL), a branch of machine learning, has expanded the possibilities of data analysis by employing more complex networks to automatically identify and extract complicated features from datasets. The huge amounts of data to be dealt with make ML and DL methods a necessity, since their ability to handle large scale data and model complex non-linear relationships is second to none.

4.2 Artificial Neural Networks

Artificial neural networks (ANNs) are a common model to implement ML concepts. They are inspired by and trying to draw insights from the functionality of animal brains. The human brain, for instance, consists of a massive number of interconnected neurons, its basic computational unit, estimated to be at about one hundred billion. Simply put, the neuron works by transmitting an electrical signal, if the incoming stimuli from other neurons is strong enough. This stimulus is received by synapses which are the intersection points between neurons. Similarly, an ANN is composed of nodes linked by edges, whereby nodes receive and process information and spread them via the edges through the network. The ability of ANNs to model complex relationships emerges from large amounts of these nodes and a non-linear operation connecting them.

It is common to visualize and think about the structure of an ANN as organized into layers. Usually, there is one input layer, one or more hidden layers, and finally one output layer. The input layer is comprised of neurons that hold the initial input information. Each neuron constitutes a feature of the entire input data. Between input and output layer, there can be one or several hidden layers that receive information from neurons of the previous layer. This previous layer can either be the input layer or

another hidden layer. Hidden layers perform calculations on the data and generate an output that is passed on to the next layer. The output layer produces the final output or prediction. Depending on the specific task that the network is designed for, the output layer may have one or more neurons.

The depth of a network is determined by the amount of hidden layers plus the output layer, since no calculation is performed in the input layer. An ANN with at least two hidden layers is considered deep [36].

If the flow of information is unidirectional from the input to the output layer, the network is called feedforward neural network.

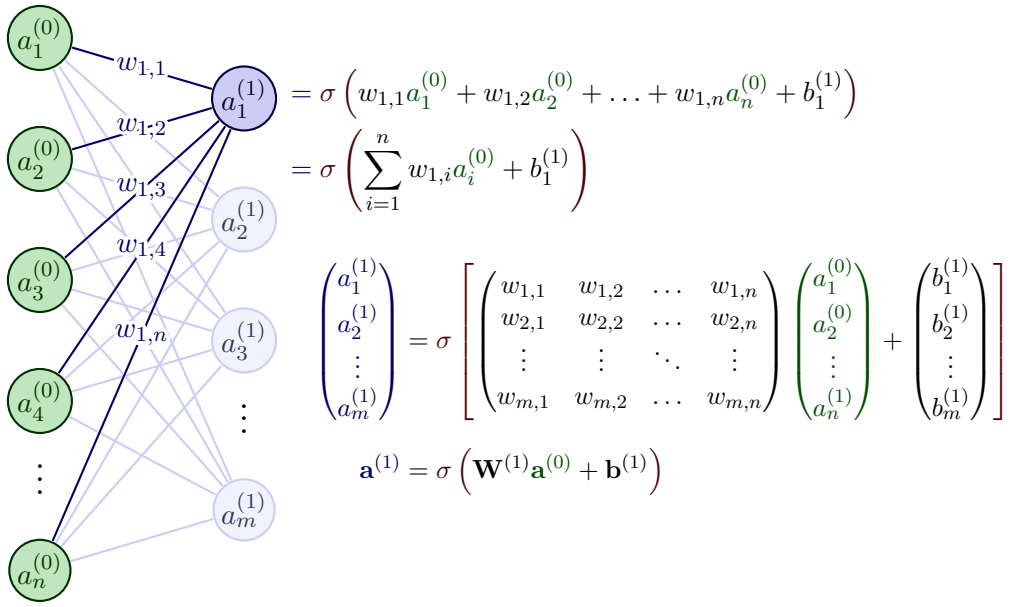


Figure 10: Schematic representation of the activation for the neuron $a_1^{(1)}$ in the first layer of a multilayer perceptron, with weights w, weight matrix $\mathbf{W}$ and biases b. The activation function σ is applied to either the scalar (first two equations) or each specific component of the resulting vector (last two equations). Adapted from [39].

The left side of Figure 10 illustrates an excerpt of a network, showing two layers, where every node in the zeroth layer is connected to all nodes in the first layer. A network with this kind of structure would be called a fully connected ANN. If, in addition to that, the flow of information is only from input to output, i.e., feedforward, the network would be called a multilayer perceptron (MLP). Each connection in the network is associated with a weight w, which is basically an evaluation on how relevant the information of the linked neuron is [36]. This can be seen as the strength of synapses in the simplified biological model.

A neuron implements a two-stage process to map inputs to an output. During the first stage, the weighted sum of the inputs to the neuron is calculated and a bias b is added to it. The weights represent the importance of the connected node's information. The primary role of a bias is to supply each node with a constant value that can be adjusted. The second stage takes the result of this calculation and passes it through a function that maps the results to the neuron's final output value. The process by which

the information flows through the network from input to output and generates a value is known as the forward pass. When designing a neuron, there are a variety of different types of functions for this second stage. They are typically called activation functions σ, because the output value of a neuron is known as its activation value. The bias effectively shifts this activation function, allowing the network to control the threshold for activation [40].

An easy way to represent this problem is by using matrix vector notation. All weights of a layer are aggregated into a matrix, where each row represents the connections between one layer k and a particular neuron in the following layer $k+1$. The activation of layer k and all biases are summarized into a respective vector. That means the activations in layer $k+1$ correspond to one term in the matrix vector product. This is particularly favorable because matrix vector multiplication is often heavily optimized in coding libraries, assuring efficient implementation. This whole process as well as the notation are illustrated on the right side of Figure 10.

The activation of any arbitrary neuron can be calculated as

$$a_j^{(k)} = \sigma\left(\sum_i^n w_{j,i}^{(k)} a_i^{(k-1)} + b_j^{(k)}\right), \tag{4}$$

with the activation $a^{(k)}$ of layer k and the components of the weight matrix $w_{j,i}$, the bias b and the activation function σ. Similar to Equation 4, the activation of an entire layer can be calculated as follows:

$$\mathbf{a}^{(k)} = \sigma\left(\mathbf{W}^{(k)}\mathbf{a}^{(k-1)} + \mathbf{b}^{(k)}\right) \coloneqq f_{\boldsymbol{\theta}_k}^k(\mathbf{a}^{(k-1)}), \tag{5}$$

where $f_{\boldsymbol{\theta}_k}^k$ represents the activation in layer k and is dependent on the parameters θ_k in that layer. Bold letters denote tensors. In this case, the activation function σ is applied to each component of the resulting vector inside the brackets.

The activation functions are responsible for implementing a non-linearity into the network, which is important to successfully represent complex relationships. A common function to achieve that is the computationally simple rectified linear unit (ReLU) defined as follows:

$$\sigma_{\mathrm{ReLU}}(x) = \max(0, x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases} \tag{6}$$

The activation function can be interpreted as the firing rate, only firing if the summed signals exceed a threshold. Activations in one layer determine the activation in the next layer, which is loosely analogous to the biological reality on how certain neurons firing results in other neurons firing.

The ultimate goal of a network like the MLP is to approximate a mapping $\mathbf{F}$, which can be written as follows:

$$y = \mathbf{F}(x, \theta_{opt}), \tag{7}$$

from the input x to the desired output y, by learning the optimal parameter values θ_{opt} for a broad class of highly parameterized functions.

4.3 Graph Neural Networks

Graphs can be found everywhere in the real world, as many systems are best understood by focusing on the relationships between objects rather than the objects themselves. A graph represents these relationships as nodes (objects) and edges (connections), making it a powerful tool for modeling complex, interconnected data. In recent years, graph neural networks (GNNs) have emerged as a framework to analyze data that can be represented in such a graph-structure.

Graphs and real world data

A graph $\mathcal{G}$ can be understood as a mathematical object that can be defined as a 3-tuple $\mathcal{G} = (U, V, E)$ [41]. Thereby, $V = \{\vec{v_i}\}_{i=1...N^v}$ stand for the set of N^v nodes, or vertices, in a graph, whereby each $\vec{v_i}$ is a node attribute (vector). Similarly, $E = \{(\vec{e_k}, r_k, s_k)\}$ represents a set of a total of N^e edges. An edge attribute can be represented by an arbitrary $\vec{e_k}$, whereas r_k denotes the index of the receiving node and s_k is the sending node. U represents a global attribute of the graph [41]. An illustration of a graph can be seen in Figure 11.

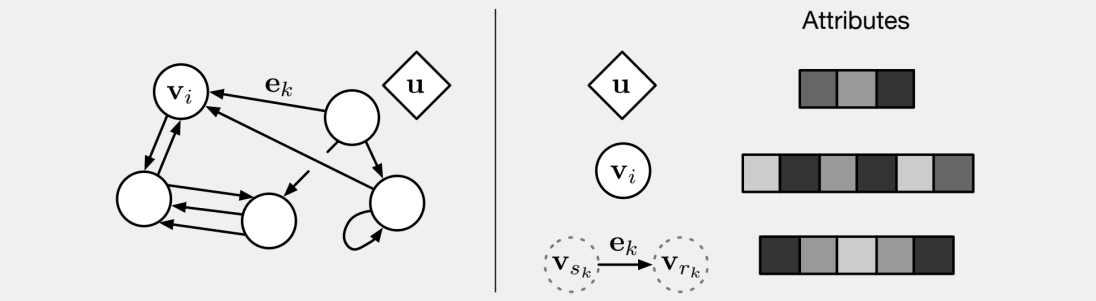


Figure 11: Exemplary illustration of a graph with a global attribute U. A node is denoted as v_i and an edge as e_k. s_k and r_k indicate the indices of the sender and receiver nodes for edge k. Taken from [41].

A common example for graphs representing real world data would be a social network. Thereby the nodes would be all the people inside the network, while the node attributes, like, e.g., name, age, gender, and occupation, can be summarized into a vector belonging to that node. The edges describe the interactions or relationships between different people inside the network. Real world graphs can vary largely in their structure, even within a given dataset. Some graphs, may have many nodes with few connections or or a lot of connections between few nodes.

Challenges in graph representation for machine learning

Representing the graph data in a way that is compatible with DL poses a challenge, since common ML models use grid-like input patterns. Grid-like input patterns are data arranged in rows and columns, like pixels in an image or values in a sequence. Graphs on the other hand, rely heavily on permutation invariance. In this context, permutation invariance refers to the idea that the representation of a graph should remain the same, regardless of how the nodes are ordered. This is important because the nodes themselves often do not have any inherent order, since what matters is the relationships between them.

The nodes of a graph can be represented quite easily with a node feature matrix, where each node n is assigned an index i and the corresponding attributes are stored at that index position in the matrix. The graph's connectivity could be represented as an adjacency matrix. For a graph with undirected edges, meaning the relationship goes both ways equally, adjacency matrices are symmetrical square matrices whose rows and columns are the vertices of that graph. Their dimension is $N_{\text{nodes}} \times N_{\text{nodes}}$, with N_{nodes} denoting the amount of nodes in the graph. In case an edge exists between a node n_i and another node n_j, the corresponding adjacency matrix has a non-zero entry at (i, j). With no loops, i.e. no node may connect to itself, the diagonal elements of the adjacency matrix are 0. The adjacency matrix in an exemplary form can be seen in Figure 12.

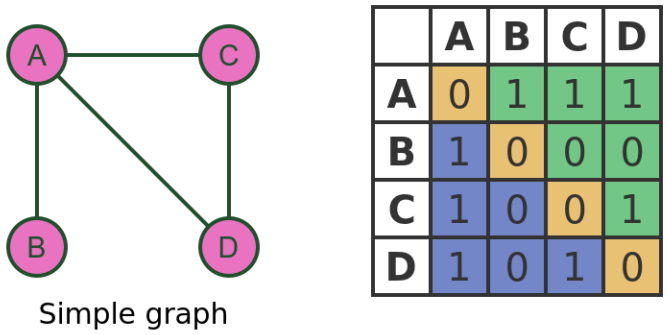


Figure 12: Illustration of a simple graph on the left. The graph can be represented as the adjacency matrix on the right. The matrix contains all the information of the graph but in a different format. Taken from [42].

For graphs with a lot of nodes there are a lot of redundancies in this matrix. Furthermore, if there is a small amount of edges, the matrix would be rather sparse, i.e. have a lot of 0 entries, which makes it poor choice from a computation point of view. Another problem is the fact that the matrices themselves are not permutation invariant with respect to the node indices. The solution to the sparse matrix problem is provided by an adjacency list, which describes the connectivity of an edge e_k between the nodes n_i and n_j as a tuple (i, j) in the k-th entry of the list. This list is preferred because the number of connections $\mathcal{O}(N_{edges})$ is expected to be much smaller than the number of entries in an adjacency matrix $\mathcal{O}(N_{nodes}^2)$.

Structure of a generic GNN

In a general sense, like any ANN, a GNN is a highly parameterized function. In particular, a GNN is an optimizable transformation on all graph attributes that aims to preserves graph symmetries (permutation invariances). It is also composed of a layered structure.

Since the connectivity of the input graph stays the same, the output graph can be described with the same adjacency list and the same number of feature vectors as the input graph.

Generally, one can differentiate between graph-level, node-level and edge-level tasks. When trying to predict one property based on a whole graph representation, that would be called a graph-level task. For a node-level or edge-level task, the prediction is supposed to be a property of a node or edge respectively.

To make predictions for any kind of task, the network has to collect all necessary information from the nodes or edges and aggregate them. This can be achieved by a pooling. Pooling is a non-parameterized operation, and aims to reduce the size of the current (node) feature vector or matrix. This is usually done by pooling/aggregation functions ρ, like sum, mean or max.

To make more sophisticated predictions, neighboring nodes or edges of the graph should exchange information to influence each other's updated feature vector. This is done via convolutional layers. These layers basically work as follows: Firstly, all neighboring node features are gathered for each node. Secondly, the features are aggregated through an aggregation function ρ and lastly, an update function or mapping, like a MLP with weights and biases, is applied. Theoretically, the second and last step could be switched, resulting in a different output, but it would still be a permutation invariant operation. Since the node ordering does not matter in GNNs because the key focus is on the structural relationships between nodes rather than their specific positions in a list, GNNs capture that fact by using permutation invariant aggregation functions and convolutional operations. An illustration of a single convolutional GNN layer can be seen in Figure 13.

With multiple convolution layers stacked together, a node can gradually gather information from across the entire graph. After three layers, for instance, a node has access to information from nodes that are three steps away from it.

To achieve, e.g., a scalar prediction, the graph representation has to be broken down to a single value. This may be achieved by a final pooling and a MLP, that can be applied after all the convolution layers. The MLP is not a update function, since it does not update, e.g., the node attributes, but takes the pooled attributes from all nodes and outputs a single value.

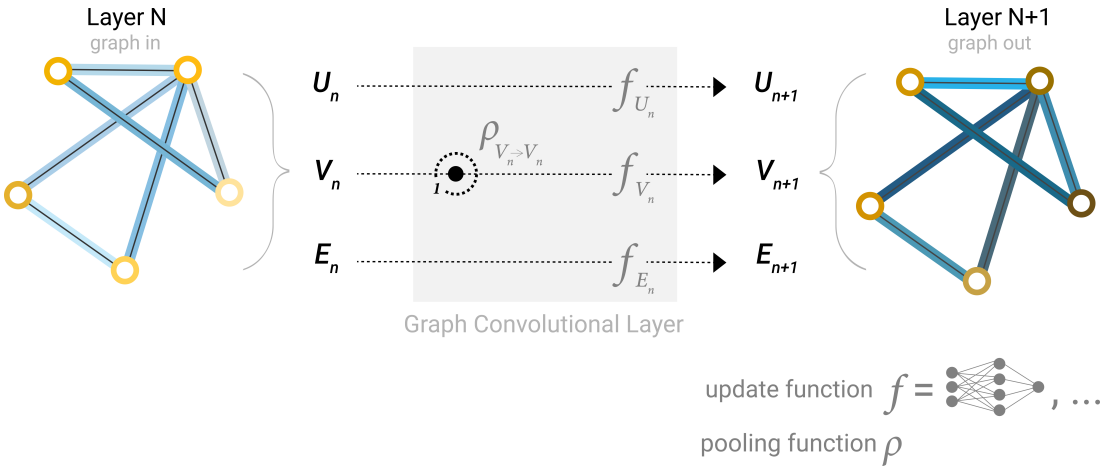


Figure 13: Illustration of a single GNN layer. A graph at the n-th layer is the input. Each component (U, V, E) is updated using a separate MLP to produce a graph with new attributes at layer $n + 1$. In this particular schematic, the update for the graph's node representations is achieved by also pooling information from neighboring nodes at a distance of one degree (denoted by $\rho_{V_n \to V_n}$). This is therefore a convolutional layer. Taken from [38].

k-nearest neighbor algorithm

When defining the neighboring nodes, i.e. drawing edges, the k-nearest neighbor (KNN) is a very handy non-parametric algorithm [43]. The goal is to identify the k closest data points based on a predefined metric, like euclidean distance:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}\,, \tag{8}$$

where n represents the number of features or dimensions in the data. For comparing two points $\mathbf{x} = (x_1, x_2, x_3)$ and $\mathbf{y} = (y_1, y_2, y_3)$ in 3D space, the associated n would be 3. In a higher-dimensional space, like IceCube data, n could be much larger, and could be representing various input features such as time, position, charge, number of hits etc.

The k-value determines the amount of neighbors a node will have, While a small k is prone to lead to noisy predictions, a larger k may result in overfitting the data. For calculation all distances for every data point in the training set is computed, which can be slow for large datasets. After calculating the distances, the algorithm selects the k data points with the smallest distances [43].

Aside from it being computationally expensive, another shortcoming is that if some features are noisy or irrelevant, they can skew distance calculations and affect the accuracy of the model [44]. But the advantages are, that no complex parameter tuning is needed beyond choosing k. Also, no training phase is needed, since no model is built for this algorithm during training. Instead, it stores the dataset and defers computation until a prediction is required [44].

4.4 Gradient-based learning

The output y of a k layered network, with input at $k = 0$, can be expressed as a concatenation of the previous activations, based on Equation 7:

$$y = \mathbf{F}(\mathbf{x}) = f^{k-1}_{\boldsymbol{\theta}_{k-1}} \left(f^{k-2}_{\boldsymbol{\theta}_{k-2}} \left(... f^{0}_{\boldsymbol{\theta}_0}(x) ... \right) \right) . \tag{9}$$

The parameters θ are randomly initialized at first. This most likely constitutes a poor guess to describe the data, but here the concept of learning comes into play. Learning, or training, is an iterative process by which the network tries to find the set of parameters that models the available data or patterns within the data best. All weights and biases in each layer contribute to the amount of trainable parameters for the neural network. The strengths of deep learning lies in its proficiency in handling large scale data. Common techniques of supervised learning are outlined in the following.

Loss function

During supervised learning, the generated output is compared to the true target output at the end of the forward pass. The loss, a quantity that measures the discrepancy between the predicted output and the target value, is calculated via the loss function $\mathcal{L}$. A loss function is supposed to harshly penalize the network if the discrepancy between prediction and true value is too big, while not being as strict for small differences. A suitable loss function for energy regression would be the LogCosh loss function [19], which is expressed as follows:

$$\mathcal{L}_{\text{LogCosh}}(\theta) = \sum_{i=1}^{N} \log \left(\cosh \left(y_i^{pred} - y_i^{true} \right) \right) , \tag{10}$$

where the model output of a data point i is referred to as y_i^{pred} and the true target value is y_i^{true}. The advantage of the LogCosh function is that it is symmetric around zero, thus punishing over- and under-estimation equally [19]. Additionally, LogCosh provides a steady gradient around zero and is approximately linear for large discrepancies [19]. The end goal is to minimize the loss for the entire network to get as close as possible to the optimal representation of the input data.

Backward pass & gradient calculation

The backpropagation algorithm [45] allows the information from the loss to flow backward through the network. In this backward pass, the gradients of the loss function are calculated with respect to the network's parameters by using the chain rule of calculus [37, p. 200]. Updating a weight requires an understanding of how that weight influences the overall error of the network. The fundamental concept of backpropagation is that each weight should be adjusted according to its impact on the network's total error. If changing a weight does not affect the overall error of the network, then the network's error is independent of that weight, meaning the weight did not contribute to the error.

The gradients represent the rate of change of the loss function with respect to each parameter and provide information about how to adjust the parameters to decrease the loss. The negative gradient points in the direction of the steepest descent. While calculating an analytical expression for the gradient is straightforward, evaluating it numerically can be computationally costly. The backpropagation algorithm simplifies this process by efficiently applying the chain rule of calculus, making it computationally inexpensive [37].

Parameter update

Backpropagation refers to the method for computing the gradient. Another algorithm, often called optimizer, such as stochastic gradient descent (SGD), is used to perform the parameter update with the previously calculated gradient. SGD computes the new weights as follows:

$$\vec{\theta}_t = \vec{\theta}_{t-1} - \alpha \cdot \vec{\nabla}_\theta \mathcal{L}\left(\vec{\theta}_{t-1}\right) , \tag{11}$$

where t represents the number of update iterations and α denotes the learning rate [46]. In the following, the notation with arrow for θ is chosen to indicate that θ is a high dimensional vector.

The learning rate is a hyperparameter (not trainable parameter or an external setting) that controls the adjustments of the weights. If the learning rate is too small, the training process may take a long time to converge to an optimal set of weights. Conversely, if the learning rate is too high, the weights may fluctuate excessively, preventing the training from converging altogether.

The stochastic part of its name comes from randomly selecting a training data point from a larger data set and computing the weight update [46]. The gradient used in this method would be an estimate of the true gradient, which would be calculated by using the entire dataset [47]. The SGD algorithm is computationally inexpensive but fairly noisy and possesses an unstable convergence behavior [48], which is why modifications exist that use e.g. the average of all gradients calculated in a batch to mitigate these shortcomings.

Another method to improve upon the standard SGD is the Adam optimizer, which was used in this thesis. All information about the optimizer stems from the original paper [48]. The name is derived from adaptive moment estimation and the functionality suggests the use of momenta to smooth out the learning process. While SGD uses a single constant learning rate α for all weight updates during training, Adam maintains a learning rate for each network weight and separately adapts it as learning proceeds. Adam calculates these adaptive learning rates for each parameter at time step t by keeping track of both the first moment m, the mean, and second moment v, the uncentered variance, of the gradients.

Specifically, Adam computes an exponential moving average of the gradient g and the squared gradient. The hyperparameters β_1 and β_2 control the decay rates of these moving averages. This first calculation step can be formulated as follows:

$$\begin{aligned} \vec{g_t} &= \vec{\nabla}_\theta \mathcal{L}(\vec{\theta}_{t-1}) \\ \vec{m}_t &= \beta_1 \cdot \vec{m}_{t-1} + (1-\beta_1) \cdot \vec{g_t} \\ \vec{v}_t &= \beta_2 \cdot \vec{v}_{t-1} + (1-\beta_2) \cdot \vec{g_t}^2 \,. \end{aligned} \tag{12}$$

In this equation, the mean is not subtracted from the gradient to calculate the variance, which is why it is called uncentered. [48]

However, the moving averages $\vec{m}_0$ and $\vec{v}_0$ are initialized as (vectors of) 0's, leading to moment estimates that are biased towards zero, particularly in the early steps of training and when the decay rates are small (i.e., when the β's are close to 1). Therefore, a bias correction has to be implemented:

$$\begin{aligned} \vec{\hat{m}}_t &= \frac{\vec{m}_{t-1}}{1-\beta_1^t} \\ \vec{\hat{v}}_t &= \frac{\vec{v}_{t-1}}{1-\beta_2^t} \,, \end{aligned} \tag{13}$$

where $\vec{\hat{m}}_t$ and $\vec{\hat{v}}_t$ are the bias corrected momenta and β_i^t denotes the respective decay factor raised to the power of t.

The final parameter update is then defined by the bias corrected momenta as follows:

$$\vec{\theta}_t = \vec{\theta}_{t-1} - \alpha \cdot \frac{\vec{\hat{m}}_t}{\sqrt{\vec{\hat{v}}_t} + \vec{\epsilon}} \,, \tag{14}$$

with an additional hyperparameter ϵ.

In the weight update, the contribution of the bias corrected momenta varies exponentially over each completed epoch. Therefore, the contribution of gradients to the updated weights varies over epochs, although the learning hyperparameter α is constant. Adam is therefore called adaptive.

By using these moving averages, Adam accumulates an exponentially decaying average of past gradients. If it comes across a single dissimilar gradient after calculating many similar gradients, the optimizer will largely disregard it. This smooths the learning process, helping the optimization process to accelerate in directions with consistent gradients and dampen oscillations.

The training process during supervised learning

By itself, the gradient descent algorithm can only be used to train a single output neuron. However, the algorithm is not sufficient for training a deep network with multiple hidden layers because it does not address how the error should be distributed across the layers. This is known as the credit assignment problem, determining how to allocate responsibility for the network's overall error among the individual neurons.

To train a deep neural network, both the gradient descent and backpropagation algorithms are used in cooperation. Backpropagation solves the credit assignment problem by determining how the error should be attributed to each neuron, while gradient descent provides the learning rule that updates the network's weights based on this information.

The whole process used to train a deep neural network can be characterized as follows: First, the weight of a network are randomly initialized and an initial output is produced. Secondly, the loss is computed, and backpropagation is applied to find out which weights affect the network how strongly. After that, the parameters are iteratively updated based on an update rule provided by an optimizer.

It is common to split the entire amount available data into three datasets: Training dataset, validation dataset, and a testing dataset.

The training set is used for updating the network parameters. During training, the data is usually divided into batches, small subsets of the whole dataset, rather than processing the entire dataset at once, since that would be computationally unfeasible. An epoch refers to one complete pass through the entire training dataset, so a traversing through all available batches. The loss function can be calculated for all the data inside a batch. Afterwards, the loss may be summed up. In this case, the variable N in Equation 10 corresponds to the amount of data points in one batch, i.e., the batch size. Subsequently, the network updates its parameters after, e.g., each batch using a variety of methods.

After the whole training set is iterated over, an epoch is finished, and the validation dataset is used to evaluate the network on an independent dataset. Using the validation dataset is a crucial method to avoid what is known as overfitting. Overfitting occurs when a model learns the training data too well, capturing not only the underlying patterns but also the noise and outliers. This results in poor generalization to unseen data, as the model becomes overly sensitive to small variations in the training set. With each epoch, the loss of the training datasets should decrease. At the beginning, the loss curve usually falls steeply and flattens out for later epochs, approaching a minimum. Overfitting can be identified when the loss of the validation dataset diverges from the training loss. The progression of the loss curves and overfitting is illustrated in Figure 14. Finally, the testing set is used to evaluate the fully trained network's performance on another independent dataset.

4.5 Classification vs Regression

What the neural network is supposed to achieve, plays a huge role during the design phase of the network itself. In that context, a task refers to a specific problem that is to be solved. Tasks are generally classified based on the type of output they are trying to predict, such as classification or regression. Both, classification or regression are a form of supervised learning [50].

Classification describes the assignment of input data to one of several predefined categories or classes [50]. The output is discrete and can refer to e.g., binary labels such as 0 or 1. A concrete example would be predicting whether a particle interaction is a neutrino or a non-neutrino event.

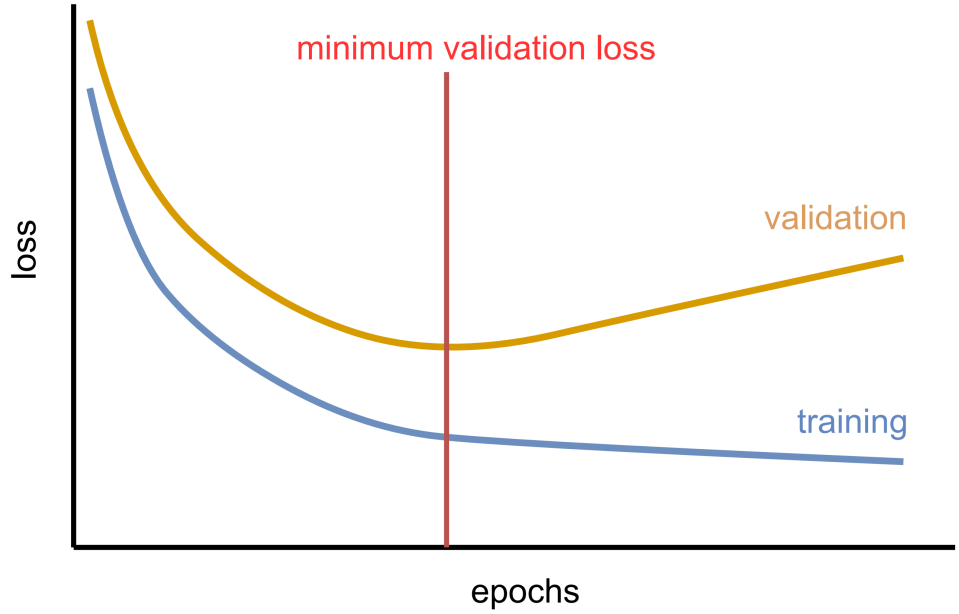


Figure 14: Illustration of the behavior of a network. The training curve falls steeply at the beginning and flattens out afterwards. The minimum validation loss signifies the model with the best fitting parameters and is marked in red. Overfitting happens when the validation loss increases while the training loss decreases. Taken from [49].

In contrast to that, regression refers to mapping the input to a continuous real number output [50]. Reconstructing the energy of a neutrino event based on sensor data would be an example of an applied regression task.

5 Application of GNNs to IceCube simulation data

In IceCube, pulses represent the electrical signals generated by optical sensors after detecting Cherenkov radiation. The number of pulses generated by an event largely depends on the event's energy, as discussed in section 3.2, making the identification and reconstruction of low-energy neutrino events especially challenging. At low energies, track events can be so short that they are difficult to distinguish from cascade-like events. While the irregular geometry of IceCube helps in differentiating events that would otherwise produce identical detector responses in a regular geometry, it also adds complexity to the development of reconstruction algorithms. Furthermore, the reconstruction process is complicated by the ice properties varying as a function of position in the IceCube array [19].

The goal of this thesis is to use a GNN to reconstruct the energy of neutrino interactions independent of the concrete event topology described in section 3.2. Additionally, this thesis will only look at events for which DIS is the primary interaction mechanism.

The following chapters go into detail about the data that was used, how the GNN and its training was configured and implemented.

5.1 Motivation for the use of GNNs

Traditional reconstruction methods may make use of maximum likelihood estimations. Since e.g. both electromagnetic and hadronic showers have a nearly constant light emission profile, it is possible to define "templates" to match observed data to an expected output [20]. This approach relies on a lot of simplifying assumptions, for example that the light output scales linearly with energy across a large energy range, and poses limited flexibility. A similar strategy, using a scalable template, works for low energy muon tracks, but large stochastic differences on an event-to-event basis lead to poor resolution [20]. To mitigate these deficits, one could use an ANN to learn relevant parameters without previous approximations and model non-linear relationships.

The authors of [51] explored how machine learning techniques, particularly GNNs, can be applied to the problem of jet tagging in high-energy particle physics. Jet tagging refers to the classification of jets, which are collimated sprays of particles produced in high-energy collisions, essentially directed showers. The usual representation of particles in jets is hierarchical: Each particle is sorted in a specific way and aggregated into a 1D-array. However, [51] introduced the concept of particle clouds to represent these jets. A particle cloud describes each constituent particle of a jet as a point in that cloud with the possibility to ascribe additional features to the respective particles. Particle clouds are permutation invariant, meaning particles in a jet can be reordered without changing the jet's identity. It was shown that particle clouds in combination with GNNs result in superior performance when compared to previous data representations.

This concept of clouds can be analogously applied to IceCube events. IceCube data is naturally structured as a 3D grid of DOMs embedded in the ice. Each DOM hit can e.g. be treated as a node, with edges representing possible correlations or the distance between neighboring DOMs. In this scenario, the recorded Cherenkov light hits would form a "photon hit cloud".

Similar to jets, the DOM hits in IceCube are permutation invariant as well. The order in which the hits are processed by the ML model does not need to follow a specific sequence. GNNs are capable of handling this naturally, unlike traditional machine learning algorithms that may depend on feature ordering. This makes GNNs particularly well-suited for neutrino event analysis in detectors like IceCube and is the reason why the model of choice used for energy regression in this thesis is that of a graph neural network.

5.2 The databases

To feed the network, input data is needed. A special relevance of simulated data, as described in section 3.4, to machine learning lies in its use to train and validate algorithms. Machine learning techniques often rely on large, labeled datasets to learn patterns and make predictions. Simulated events provide a controlled environment to develop and test these algorithms, whereas real raw data is obviously not labeled and lacks the necessary information to apply supervised training.

A total of 9 simulation datasets were considered. Each neutrino flavor was divided into 3 energy ranges respectively: Low-energy from $10^2\,\mathrm{GeV}$ to $10^4\,\mathrm{GeV}$, mid-energy from $10^4\,\mathrm{GeV}$ to $10^6\,\mathrm{GeV}$ and high-energy from $10^6\,\mathrm{GeV}$ to $10^8\,\mathrm{GeV}$. The high-energy events were simulated with a less steep spectrum of $\propto E^{-1}$ compared to the mid- and low-energy event spectrum of $\propto E^{-1.5}$. This was done to receive adequate statistics.

The preparation of the data was largely part of Fabian Wohlfahrt's Bachelor thesis [52]. His thesis focuses on GNNs to build a starting event filter, similar to HESE (see section 3.3) and a double bang filter to detect double bang events. To train these networks he used simulated data. His work included converting the custom simulation data format, called .i3, into a more suitable format for ML training, such as the database format .db. Additionally, he differentiated the events into a multitude of classifications. Simulation weights are calculated to account for the falsely assumed spectrum and the increased interaction cross section for neutrinos. The events in the finished database are filtered on level 2 but rejected events remained in the database.

The huge amount of data that was processed for Fabian's thesis was cut down to events and relevant classifications that this thesis is supposed to explore. The relevant events were written into a separate database files.

The separate databases were split according to their function during training later on into training, validation, and testing databases. For an overview, refer to Table 1. Each flavor provides 1.5×10^6 events, which can be considered a sensible approach because it prevents bias toward any specific flavor. A balanced dataset helps the model learn features for each class equally. This is important since the energy reconstruction needs to perform equally well across all flavors, regardless of how often each flavor appears in nature. The event classifications considered in this thesis include hadronic cascades from all neutrino flavors as a result of NC interactions. For ν_e^{CC} interactions, electromagnetic and the following hadronic cascade are included as one signature in the relevant classifications. For ν_μ^{CC} interactions, both starting and contained muon tracks are examined. A contained track refers to a muon that is both produced and decays within the detector volume. Additionally, ν_μ^{CC} interaction can yield throughgoing and

Flavor	Energy Range [GeV]	Train db	Val db	Test db	Total
ν_e	$10^6 - 10^8$	81 000	9000	10 000	100 000
	$10^4 - 10^6$	324 000	36 000	40 000	400 000
	$10^2 - 10^4$	810 000	90 000	100 000	1 000 000
ν_μ	$10^6 - 10^8$	81 000	9000	10 000	100 000
	$10^4 - 10^6$	324 000	36 000	40 000	400 000
	$10^2 - 10^4$	810 000	90 000	100 000	1 000 000
ν_τ	$10^6 - 10^8$	81 000	9000	10 000	100 000
	$10^4 - 10^6$	324 000	36 000	40 000	400 000
	$10^2 - 10^4$	810 000	90 000	100 000	1 000 000
Total		3 645 000	405 000	450 000	4 500 000

Table 1: Amount of events considered for the training, validation and test databases (db). Each flavor is split into 3 energy ranges respectively. In total there were 4.5×10^6 considered events.

stopping muon tracks originating from outside the detector. Lastly, double bang events, resulting from ν_τ^{CC} interactions, that can be followed by either an electromagnetic or hadronic shower, are included. The amount of events for each classification can be seen in Table 2. The event topologies themselves were picked based on random chance. This random distribution of event signatures ensures that the model encounters a wide variety of neutrino interactions and aims to avoid biases.

Classification	Train db	Val db	Test db	Total
hadr_cascade	575 297	64 292	70 858	710 447
em_hadr_cascade	981 285	108 822	121 129	1 211 236
starting_track	286 286	32 067	35 485	353 838
contained_track	161 848	17 933	19 822	199 603
double_bang_em	154 622	17 100	18 947	190 669
double_bang_hadr	608 623	67 568	75 131	751 322
throughgoing_track	698 578	77 470	86 532	862 580
stopping_track	178 461	19 748	22 096	220 305
Total	3 645 000	405 000	450 000	4 500 000

Table 2: Amount of considered events based on classifications for the training, validation, and testing databases (db). These events were chosen at random to avoid biasing the selection.

Simulation weights are calculated for each of the original databases. Weights based on both an astrophysical and an atmospheric flux model are computed. Since both fluxes contribute to the neutrino population in the detector, the total simulation weight is the sum of both calculated weights. A database as a whole with associated weights is

considered as the accurate physical reality. Therefore, when sampling from the original databases and creating new ones, they should also contain adjusted simulation weights. These weights are then used for analysis after training to reflect real world conditions.

The chosen events do not include background events. They do, however, include events below the Level2 filtering stage, which can be largely considered junk and and poses an extra challenge for the network.

5.3 The GraphNet framework

In this thesis, the so called GraphNet framework [53] is used. GraphNet is an open-source python framework for deep learning applications and aims to be at the interface of complex ML tasks and (neutrino) physics data analysis. It expands upon PyTorch, an open source library for machine learning. GraphNet relies on four key subpackages: graphnet.data, graphnet.modules, graphnet.training and graphnet.deployment. graphnet.data makes it easy to convert domain-specific data into standard file formats and read that data. This package was used to create the .db-files from the simulated data. The graphnet.models subpackage is used to configure and build complex GNN models. The graphnet.training subpackage includes all necessary classes and methods to manage training and logging, while graphnet.deployment is responsible for applying a trained model to a dataset and make predictions.
A typical workflow with the GraphNet framework can be seen in Figure 15.

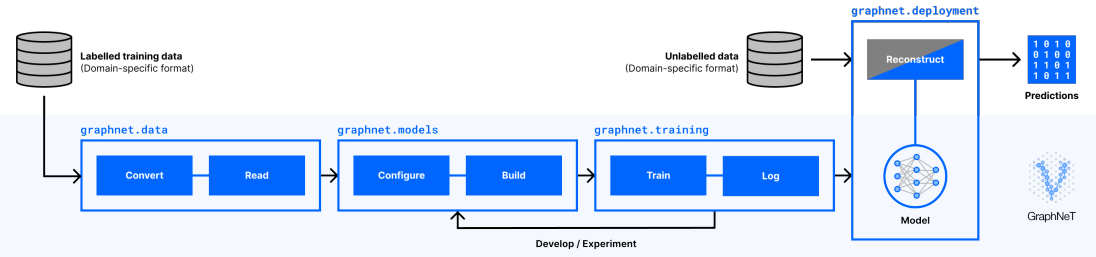


Figure 15: GraphNet flow chart. Taken from [53].

The subpackage graphnet.models includes the *Model* class. This class offers a flexible frame for constructing models and enables users to build, save, load, and configure models for their specific tasks. A *Model* can be straightforwardly defined through the *StandardModel* subclass, which provides additional functionality. The *StandardModel* is comprised of a *GraphDefinition*, a *Backbone* and a *Task*.

The *GraphDefinition* defines how the input data is structured as a graph. It describes the nodes, edges and features, determining how input data is represented for processing by the network. In defining a *GraphDefinition* it is possible to assign each DOM hit or each DOM itself to a node.

The *Backbone* refers to the actual core architecture that processes the graph data. In a sense, the *Backbone* is the GNN model itself, described in section 4.2 and section 4.3, and operates on the graph created by the *GraphDefinition*. The *Backbone* performs operations like edge convolutions and pooling.

The *Task* describes to the specific goal of the model, such as classification or regression. *Tasks* are the objectives the model is designed to optimize during training.

One can import a few pre-defined components and chain them together to create a complete and trainable model [53].

5.4 Specific implementation of the GNN with GraphNet

5.4.1 Preprocessing and dataset definition in GraphNet

GraphNet offers teh option to convert the data in databases into a suitable dataset format for machine learning. All training operations are executed on the dataset instances or the dataset-related objects. The training dataset is set to be shuffled, meaning a new training script execution leads to a different event distribution inside the batches.

The parameter that the model will be trained on is the total deposited energy inside the detector volume. Deposited energy refers to the total energy transferred from the neutrino interaction into the ice, which corresponds to the energy of any secondary particle that produces a shower or track (i.e. muon and tauons). However, it does not include energy that is carried away by particles that are not detected, such as outgoing neutrinos. Deposited energy is a fraction of the neutrino's initial energy. While the pulse information, via Cherenkov photons, are the signal used to describe events, the photons do not carry the bulk of the deposited energy themselves. Instead, the GNN model essentially has to infer the energy of the particles that created the Cherenkov photons based on the observed light pattern. The true deposited energy parameter is by itself not included in the databases and has to be calculated separately:

$$E_{true}^{depo} = E_{\text{first\_vertex}} + E_{\text{second\_vertex}} + E_{\text{third\_vertex}} + E_{\text{visible\_track}} + E_{\text{visible\_spur}} \,. \quad (15)$$

The vertex energies denote the deposited energy in the first, second or third interaction vertex of the event inside the detector and describe the energy associated hadronic or electromagnetic showers. The visible track energy $E_{\text{visible\_track}}$ denotes the deposited energy in tracks inside the detector, so primarily stochastic (or ionization) muon energy losses. Lastly, the visible spur energy $E_{\text{visible\_spur}}$ describes the deposited energy in spurs inside the detector. Spurs is a term coined to characterize the stochastic energy losses of a tauon. All the energies on the right hand side of the equation are included in the database. GraphNet provides a custom class with the possibility to calculate labels on the fly and then adding them to the dataset object. Only events with a total energy deposition of above 10 GeV are considered to be relevant. This was already accounted for in the event selection, therefore all events inside the databases, see Table 1, are above that threshold.

Another issue is the large amount of NULL entries inside the relevant columns in the database. Since a throughgoing track for example does not interact inside the detector, the first to third vertex energies are not defined and therefore NULL. This poses a problem because the GraphNet dataset implementation then ignores these important columns, not including them into the dataset, or even raises errors in response. To circumvent that, a custom dataset class was implemented that could handle the NULL entries.

The detector class defines detector properties and provides the option to apply the whole model to different detector geometries, thereby offering a high amount of customizablity. The standard detector class includes the PMT-area as a metric as part of its feature map. This is redundant since all the in-ice-array DOMs posses the same PMT-area and therefore, a custom detector class with an updated feature map was defined.

5.4.2 The GraphDefinition

The first step in implementing the GNN is to define what the input data is supposed to look like as a graph. The idea is to represent each IceCube event as a graph. Each node may represent a DOM hit as discussed in section 5.1. The feature vectors of these nodes would contain attributes of the respective hits, e.g. xyz-position, time, and charge. While this is certainly possible, the number of nodes would then equal the number of hits, which means a high energy event would have an immense amount of nodes. This is in most practical cases not feasible, since the large graph significantly slows down the computation. An alternative is to define each node as a DOM that was hit and calculating representative values from all hits of that DOM. This technique is referred to as summary statistics [23]. The concrete summary statistic used in this thesis is percentile clustering. A cluster refers to a group of data points that are similar to each other based on certain features or characteristics. The goal of clustering is to group data in such a way that points within the same cluster are more similar to each other than to points in other clusters [54]. The n-th percentile, P_n, represents the value below which $n\%$ of the data points fall. The chosen percentiles for this thesis were $n \in [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$. Percentiles are used to provide a robust description of the data distribution, since the data is spread out between P_0 and P_{100}. Summary statistics come with some drawbacks. By condensing the input data, there is an unavoidable loss of information. To retain enough information, enough percentiles have to be used. However, this leads to a second issue: For DOMs with only a few hits, the number of percentiles may exceed the number of hits, making the node larger than the combined individual nodes would be without using percentiles. In addition to the percentile implementation, GraphNet provides the option to add *counts* to the node feature vector, which is *True* per default and was not changed. Summary statistics are useful for high energy events, since all the hit information can not easily be processed, but is suboptimal for low energy events due to the possibility that there is not enough information to fill the percentiles.

The node attributes are usually defined by the feature map of the detector class. With that in mind, the nodes attributes are therefore comprised of the following metrics:

1. x, y, z are the euclidean coordinates of the DOM,
2. *rde* is the DOM's relative detection efficiency,
3. *hlc* (hard local coincidence) refers to a specific trigger condition in which two or more DOMs detect photons within a short time window and in close proximity to each other,
4. $P_n(t)$ represents all percentiles of the hit times,

5. $P_n(q)$ represents all percentiles of the reconstructed charges. Charges are the integrated pulse information over time, which corresponds to the total number of photons detected during the event, and

6. *counts* refers to the number of photon hits of that DOM in a specific event.

The percentiles were defined by clustering on the first 3 of the above metrics. *counts* on the other hand is not part of the feature map, but since it is added later, it also is not included in the cluster.

One node has therefore 28 attributes: 3 euclidean coordinates, 1 for *rde*, 1 for *hlc*, 1 for *counts*, and 2 times 11 percentiles.

These values not passed to the *backbone* as they are, but are instead rescaled to manageable intervals to prevent exploding values because of too large inputs. The rescaling operations are as follows:

$$\begin{aligned}
(x, y, z)^\intercal \to &\quad \frac{(x, y, z)^\intercal}{500.0} \\
t \to &\quad \frac{t - 1.0 \times 10^4}{3.0 \times 10^4} \\
q \to &\quad \log_{10}(q) \\
rde \to &\quad \frac{rde - 1.25}{0.25} \\
counts \to &\quad \log_{10}(counts)\,.
\end{aligned}$$

The edges are defined by the KNN algorithm, as described in Figure 4.3. In the first step, nodes that are closer together, based on position in euclidean space, are connected to each other, while far apart nodes have no edge between them. This fact follows the idea that a close proximity of DOMs means that their information is more correlated, compared to DOMs that are further away. In this case, k was chosen to be 8, which signifies that the 8 nearest DOMs have an edge between them. For a larger amount of DOMs hits, this would constitute a sparse graph, as described in section 4.3, and the adjacency list formalism is computationally more suitable. In later steps, the inputs to the metric, defining the connected nodes changes. For more detail on that, see section 5.4.3.

5.4.3 The Backbone: DynEdge

DynEdge is a GNN model designed specifically for dynamic edge operations on graph-structured data. It has been used in IceCube-related projects. An implementation comes per default with the GraphNet installation and offers a lot of possibilities for customization. A depiction of the model functionality can be seen in Figure 16.

The first step is the computation of global variables across the graph for each event. This includes the calculation of homophily ratios for x, y, z and t. The homophily ratio is a measure that captures how similar connected nodes in a graph are in terms of their features and corresponds to a number between 0 and 1. Depending on the interpretation

of the word homophily, there may be several outcomes. Following the published paper on the DynEdge architecture [19], the homophily ratio of xyz would indicate what fraction of connected pulses originate from the same PMT. As an example it is stated, that for a graph where all nodes are connected to each other, a homophily value of 0.5 would indicate that half the pulses in the event are same-PMT pulses. "As such, the homophily ratio quantifies how many of the pulses originate from the same PMT and how many of the pulses happened at the same time." [19] When applying summary statistics and representing each DOM, a fair assumption would be that the homophily ratio would be trivially 0. That is because each DOM possess one associated PMT, and all same-PMT pulses are already aggregated into one node as percentiles. Another interpretation follows [55], where the homophily is the fraction of edges in a graph which connects nodes that have the same class label. Since the specific implementation of this thesis does not rely on class labels, one could widen the interpretation to mean alike features, i.e. spatial and time coordinates that are similar. Similarly, the documentation in the implementation of the calculation method states, that homophily measures the likeness of features in nodes. This would also suggest that DOMs with similar xyz-coordinates would contribute to a non-zero homophily ratio. Due to the unclear interpretations, the homophily ratio from the not excluded from the standard implementation in this thesis. In addition to the homophily ratio, the global means of the node features are calculated. In this case that would signify the mean of each feature listed in section 5.4.2. Furthermore the logarithm (base 10) of the total amount of pulses also contributes to the global variables as an additional attribute. In the end of this first step, 33 global attributes are calculated.

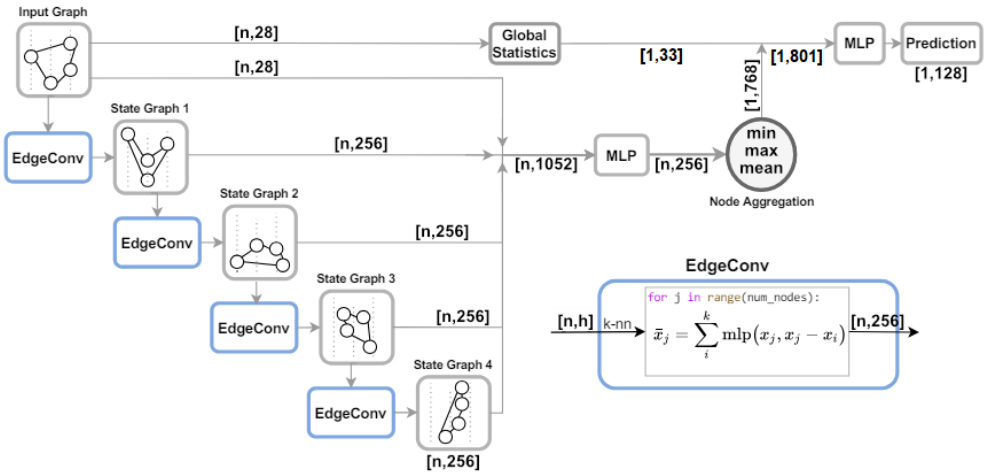


Figure 16: Schematic overview of the DynEdge model. The concrete functionality highly depends on individual implementation. For this thesis, the global statistics step show above is not added after pooling, but instead, the [1,29] dimensional vector is added on top of each node before any EdgeConv block to provide global information. n denotes the number of nodes, which in this case is the number of DOMs that were trigger for the considered event. Adapted from [19].

Next, depending on the flag *add_global_variables_after_pooling*, the global variables are either added after all convolutions, which is shown on the top of Figure 16, or alternatively distributed to all nodes before any edge convolution. In this thesis it was opted to for the latter option. This can influence further operations, as the node features can carry global information during each edge convolutions.

The core of the DynEdge model is its series of 4 edge convolution layers, each with its own MLP. In these steps DynEdge relies on the EdgeConv operator, which convolves every node with node features x_j with local neighborhood of that node. Note that, in the implementation for this thesis, the node features are always extended by a copy of the 33 global variables before any convolution, unlike what Figure 16 suggests. The EdgeConv block can be formulated as follows:

$$\tilde{x}_j = \sum_{i=1}^{N_{neighbors}} MLP(x_j, x_j - x_i)\,, \tag{16}$$

where $\tilde{x}_j$ describes the new convolved features of node j. The MLP takes the unconvolved features of node j and the difference of the unconvolved features of node j and its neighbor i as input and produces an output. This output is summed up for each neighbor i and constitutes the new convolved feature. EdgeConv is instantiated with the KNN algorithm, as described in section 4.5, whereby the number of neighbors corresponds to the k-value and was chosen to be 8. The MLP uses the ReLU activation function. A full convolution layer would apply Equation 16 to all nodes in the graph. The important feature of DynEdge is that the network's structure allows edges between nodes in the graph to be dynamic. That means at each layer, the model can recompute which nodes are neighbours based on the features learned up to that point. Each EdgeConv block produces an output which forms the input for the next block. For the first layer, so the first EdgeConv block, DynEdge calculates the 8 nearest euclidean neighbors based on xyz coordinates from the input graph. The subsequent convolution blocks redraw the edges based on KNN algorithm and the computed convolved features. So instead of true xyz-space the edges are now drawn in an "increasingly abstract latent space". This means that at each layer, the nodes may receive information from a different set of neighbors, allowing the network to capture more complex and potentially non-local relationships in the data. In essence, this method allows the model to not only learn the optimal features but the optimal neighborhood for any given task as well [19]. At each convolutional block, the output is stored and at the end of all convolutions, these intermediate representations are concatenated along with the original node features to form a set of all node features. These node features are passed to a post-processing MLP that further transforms these features and reduces their dimensionality to a total of 256 features. After that, global pooling is applied using the aggregation methods minimum, maximum and mean. The 3 resulting [1,256] vectors are stacked to produce a 786 dimensional feature vector as output. This high dimensional vector, made up of the pooled features, is lastly fed into a readout MLP to produce the final model output with dimensions [1,128]. The *backbone* has a total of 1.4×10^6 trainable parameters [56].

5.4.4 The Task: Energy reconstruction

A *task* refers to a specific goal the model is trained to accomplish, such as classifying or regressing a particular feature from the input data. The parameter that the model is trained on is defined by the *task*. For this thesis, the deposited energy was chosen as the training parameter, which can be calculated via Equation 15 for all event topologies. This parameter is a sensible choice since deposited energy provides a quantity that is actually measurable by the detector and can then, in subsequent steps, be related to the incoming neutrino's energy. A model trained on deposited energy can generalize well to a wide range of events because it focuses on accurately reconstructing what the detector measures.

The *task* takes latent output from the *backbone* and adjusts it via a MLP to the desired output, in this case one real value number. During the forward pass, a transformation to the input x is applied to ensure that the energy remains positive:

$$\tilde{x} = \frac{1}{\beta} \log\Big(1 + e^{\beta x}\Big) + \epsilon \,, \tag{17}$$

with scaling parameter $\beta = 0.05$ and $\tilde{x}$ denoting the transformed input. The first summand is called *softplus* and behaves linearly for large values, while returning a slightly bigger value than 0 for negative inputs. Even though *softplus* mathematically returns strictly positive values, the values can get extremely close to zero for very negative inputs. In such cases, due to the limitations of floating-point precision in computers, these very small values could be rounded down to exactly zero. This is why ϵ is added: To combat a possible underflow problem.

Before calculating the loss, the logarithm with base 10 is applied to both the prediction and true target values. Predicting log-transformed values allows the model to handle wide-ranging data more efficiently by compressing large values and spreading small ones. This reduces sensitivity to large values, and leads to more stable learning. The log-transformation is the reason for applying Equation 17 previously: The logarithm is not defined for negative values.

The loss calculation is performed using the LogCosh-loss, as described in section 4.4 and Equation 10, on these log-transformed values:

$$\mathcal{L} = \ln\left(\cosh\left(\log_{10}\left(\frac{E_{\text{pred}}}{\text{GeV}}\right) - \log_{10}\left(\frac{E_{\text{true}}}{\text{GeV}}\right)\right)\right) \,, \tag{18}$$

wherby the argument of the LogCos-loss function can be defined as the residual $\mathcal{R}$:

$$\mathcal{R} = \log_{10}\left(\frac{E_{pred}}{\text{GeV}}\right) - \log_{10}\left(\frac{E_{true}}{\text{GeV}}\right) \,. \tag{19}$$

Later, during inference, prediction on unseen data after training, the model will output predictions in the same format as during training. Therefore, after applying the log transformation to the target values during training, the reverse transformation, i.e. 10^x is applied to obtain predictions ready for interpretation on the original scale.

5.4.5 Training configuration

For the training in this thesis, the Adam optimizer is used, as detailed in section 4.4. The default parameters were maintained, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. However, the learning rate α was adjusted to 10^{-4}. In addition to that, the `ReduceLRonPlateau` scheduler was used to control the learning rate. A scheduler is a sensible addition to the training routine, since without it, a constant or too-high global learning rate can cause the model to "overshoot" local or global minima, resulting in unstable training or slow convergence. The scheduler works on top of Adam to reduces the learning rate globally, after a set amount of patience, when the loss curve indicates that further rapid progress is unlikely. The patience was set to 2 epoch, and the chosen learning rate reduction factor was $\frac{1}{5}$.

Adam processed batches of 12 events at a time, any larger batch size would result in memory problems due to the pulse information for high energy events being a lot more extensive. An optimization step was performed after every 10 batches to effectively simulate a bigger batch size. This allowed the gradients from all previous batches to accumulate and reduce statistical fluctuations. A total of 50 epochs was set.

Furthermore, syntax to support checkpointing was implemented. Checkpointing in ML refers to saving the model's current state, including its weights and training progress, at regular intervals or after specific conditions are met. This allows the training process to be resumed from a saved point if interrupted, preventing loss of progress due to system failures or time constraints.

To deal with overfitting, checkpointing in GraphNet also provides the possibility to store the best-performing model during training, based on the validation loss, which is then used for ensuring the most optimal model is performing inference or the prediction.

5.5 Computing setup

The GNN training was conducted on FAU's high performance cluster, utilizing both CPU and GPU nodes. CPU nodes from the 'woody' cluster [57] were used to handle database splitting, while the 'tinyx' GPU cluster [58] was employed for model training. All databases had a size of approximately 511 GB and had to be accessed during training. The initial storage spot came with read-speed limitations and severely impacted the training duration. To improve data access speeds, the relevant databases were copied to a GPU node-local temporary directory every time before executing the training script. The temporary directory, *TMPDIR*, provides high bandwidth and low latency with an average measured write speed of 0.5 GB/s for 4 parallel copy processes. Given the time constraint of a maximum of 24 h for uninterrupted GPU allocation, checkpointing was essential to save progress and allowing the model to resume training afterwards. On 6 GPU cores, the average training time with this setup was ≈ 1.5 h per epoch.

5.6 The interquartile range

The interquartile range (IQR) is a measure of statistical dispersion, usually representing the middle 50 % of a dataset. It is calculated as the difference between the 75 %

percentile (Q3) and the 25 % percentile (Q1), providing a range where the central half of the data lies.

For this thesis, the middle 68 % of data was equated with the IQR. The middle 68 % is commonly used in contexts like normal distributions, and would correspond to a standard deviation of $\pm 1\sigma$. The IQR is useful for understanding the spread of the data while being resistant to outliers, since it focuses on the middle of the distribution. [23] To evaluate the resolution of the GNN model, the interquartile range based on the residual $\mathcal{R}$, defined in Equation 19, was calculated:

$$\text{relative resolution} = \frac{p_{84\text{th}}\left(\mathcal{R}\right) - p_{16\text{th}}\left(\mathcal{R}\right)}{2}\,, \tag{20}$$

where $p_{84\text{th}}$ and $p_{16\text{th}}$ correspond to the 16th and 84th percentiles, respectively.

6 Results and discussion

In this chapter, the results achieved with the GNN model developed and trained in this thesis for flavor-agnostic energy-reconstruction are presented and discussed.

From this point forward, any reference to the flavor-agnostic model pertains to the GNN model proposed in this thesis.

For the analysis of the network's performance, the interquartile range, described in section 5.6, and other methods, e.g. bias and relative error calculations, are applied.

6.1 Model training progress

After training on the deposited energy parameter, the best fitting model was reached at epoch 41. At that epoch, the model parameters reached the lowest training and validation loss simultaneously. As described in section 4.4, this minimum means the model is not overly specialized on the training data but has potential for generalizations on unseen data. After epoch 41, no reduction in validation loss, but reduction in training loss can be observed, which would indicate overfitting. Both loss curves can be seen in Figure 17. The data points shows a steep initial fall for early epochs and a gradual

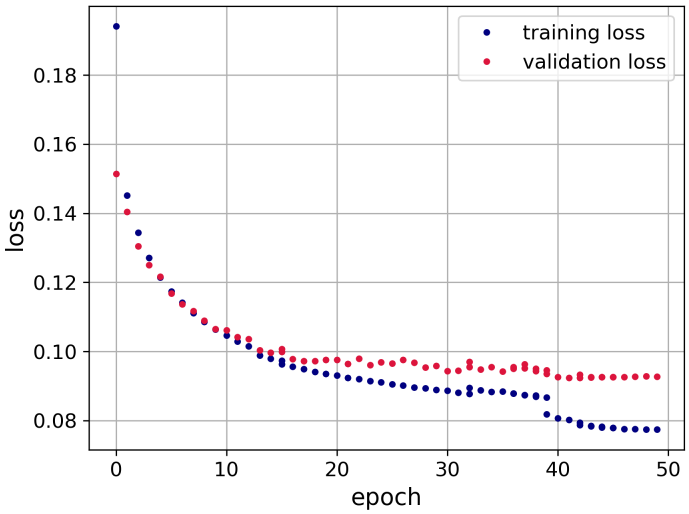


Figure 17: Loss curves for training and validation.

decline afterwards, which is the expected progression. For some epochs, e.g. epoch 15 or 32, two values for the training and validation loss can be seen. This is due to the usage of checkpoints and the fact that the training data was shuffled each time the script was executed. After the 24 h time allocation limit for a GPU node ends, the best fitting model is saved in a checkpoint, which does not necessarily have to be the model at the latest epoch. When starting from the best fitting model, for example at epoch 14, the newly shuffled dataset therefore leads to new gradients and a different loss value for the following epoch (epoch 15). The most notable jump in the data at epoch 39, might be

because the optimizer finds a local minimum in the loss landscape after a checkpoint at epoch 35.

With the best-fitting model found at epoch 41, the next step was to investigate how well it performs on unseen data. The testing set provides a clear picture of the model's ability to generalize beyond the training set. The following sections concentrates on this evaluation, discussing metrics to assess how accurately the model can reconstruct the deposited energy.

6.2 Overall network performance

The energy prediction on unseen data took approximately 25 min 45 s on one Nvidia A100 GPU core, which corresponds to a prediction rate of ∼0.29 kHz. In other words, the network can reconstruct the energy of 290 events per second in real-time. This reconstruction rate is about 17 times slower than the literature value of 5.1 kHz for a GNN trained on low energy data (between 1 GeV and 100 GeV) [19]. The reason for the large difference is that the model in this thesis was trained on events with energy depositions up to 10^8 GeV. Higher energy events hit more DOMs, leading the associated graph representation to increase in size. Additionally, the prediction rate is about an order of magnitude smaller than the median trigger rate of IceCube at 2.7 kHz. The entire Level2 filtering takes approximately 4 msec per event [23], whereas the prediction rate corresponds to an average computation time of 3.45 msec per event. That means that the GNN is slower than any individual IceCube online filter. However, the prediction rate was not yet optimized, which could be the subject for further studies. Despite all of that, it could be possible to run multiple instance of the GNN model in parallel on different filter clients. Therefore, the GNN model has potential for real-time filtering applications, especially after optimization.

Another parameter that reflects the accuracy of the network's energy reconstruction is its deviation from the true energy. Figure 18 shows the logarithmic reconstructed energy minus the logarithmic true deposited energy, i.e., the residual $\mathcal{R}$ as a histogram, whereby the counts are weighted by the simulation weights. The mean and standard deviation of the weighted residuals are calculated and displayed in the plot. The histogram for ν_μ is almost completely covered by the combined histogram for all flavors, which is especially evident due to the logarithmic y-scale of the plot. This is expected, as the (atmospheric) flux of muon neutrinos is significantly higher than the fluxes of the other two neutrino flavors. On the one hand, the higher flux stems from the production mechanisms in cosmic ray interactions, where more ν_μ are generated than other flavors. Additionally, the effective detector volume for muon neutrinos is larger because the muons produced in ν_μ^{CC} interactions are capable of traveling significant distances after being produced. Muons can trigger detection even if they are generated outside the immediate volume of the detector. Such events are classified as throughgoing or stopping tracks in the used databases. In contrast, electron and tauons resulting from ν_e^{CC} or ν_τ^{CC} interactions are much shorter-ranged and tend to deposit most of their energy within a limited area around the interaction point. As a result, these neutrinos are less likely to produce detectable signals if they interact outside the detector volume, thus reducing their effective detection volume.

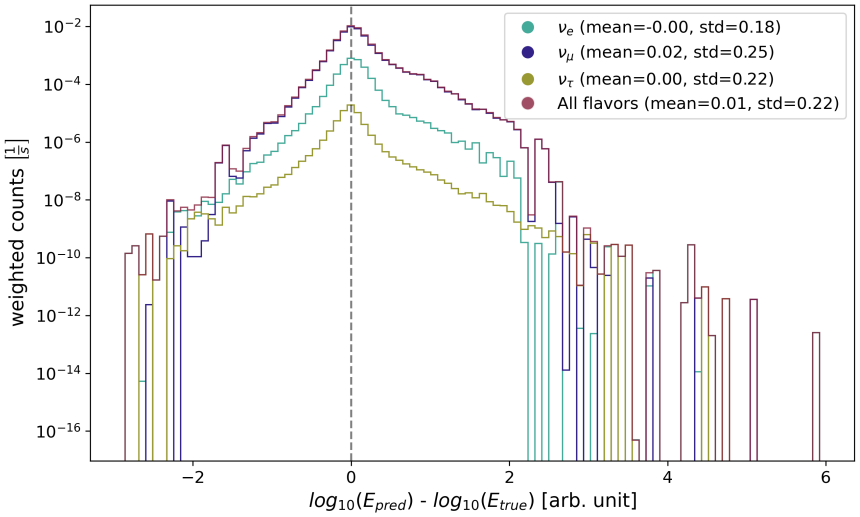


Figure 18: Weighted histogram for logarithmic reconstructed energy minus logarithmic true deposited energy. The counts are weighted using the calculated simulation weights. The y-axis is plotted on a logarithmic scale.

Despite that, the weighted histogram shows no significant bias toward any particular neutrino flavor, being able to predict all 3 flavors equally well.

To evaluate whether or not the network has a bias for a specific true deposited energy, one can investigate the bias shown in Figure 19. The bias, in this case, is calculated as the mean residual for a given true deposited energy bin:

$$\text{bias} = \text{mean}\left(\log_{10}\left(E_{pred}\right) - \log_{10}\left(E_{true}\right)\right) = \text{mean}\left(\mathcal{R}\right) \tag{21}$$

Figure 19 shows a nearly linear decline in bias for energies between 10 GeV to $\sim 10^2$ GeV, falling from a bias value of ~ 1.11 in the leftmost bin to ~ 0.16 at the bin with bin center at $\sim 10^{1.93}$ GeV. A bias of 0.16 would indicate that, on average, the predicted energy is about $10^{0.16} - 1 = 45\%$ bigger than the true energy in that bin. Therefore, the energy predicted by the network is, on average, almost twice the true energy or more for events with a true energy at $\sim 10^{1.93}$ GeV. For events with true deposited energies below $\sim 10^{1.93}$ GeV the bias is even larger.

For energies above $\sim 10^2$ GeV, the bias stays close to 0, with, e.g., ~ -0.01 at $\sim 10^6$ GeV. That means that the network could, on average, predict the logarithmic true deposited energy values almost perfectly.

The errorbars in Figure 19 show the standard error of the predicted energies in each bin and indicate the uncertainty of the calculated bias values. These uncertainties are smaller for higher deposited energies. This behavior is expected because lower energies are generally more challenging to reconstruct. The lower amount of Cherekov photons provides less information for the model. Additionally, the worse performance for low energies can partially be attributed to the usage of summary statistics in the model architecture and the associated shortcomings explained in section 5.4.

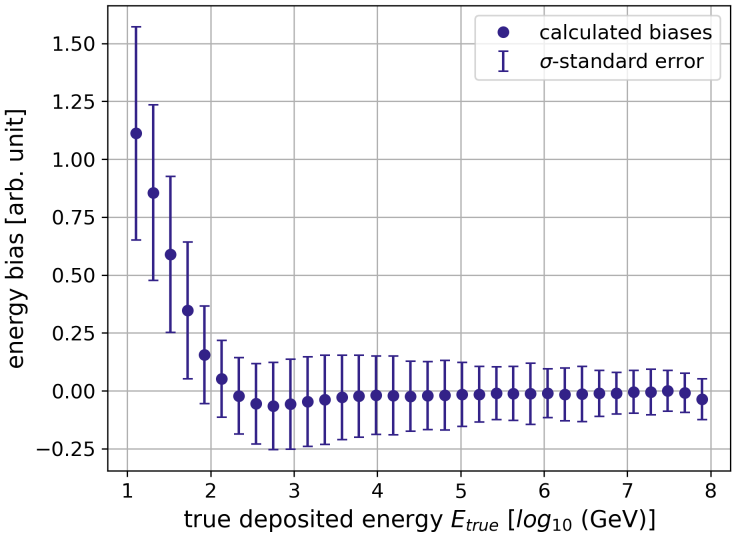
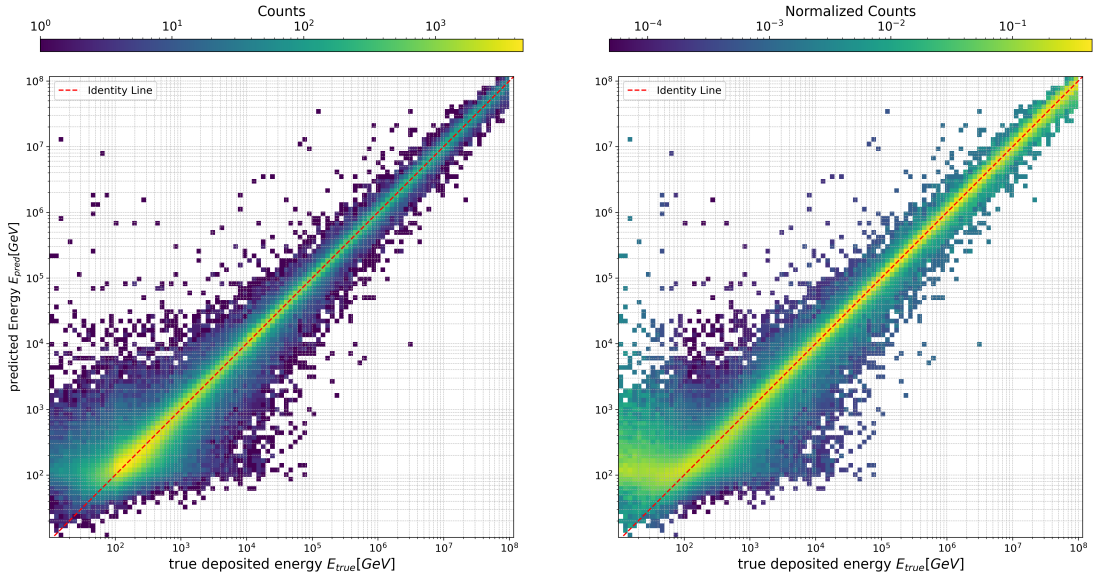


Figure 19: Plot of the calculated biases against the true deposited energy with errorbars.

Another way to visualize how well the network predicted the true values are heatmaps or 2D histograms. Figure 20a shows an unweighted and not normalized 2D histogram of the predicted energy against the true deposited energy. The "identity line" is the bisecting line and indicates a perfect prediction. The majority of events above about 10^2 GeV (true deposited energy) are concentrated along this identity line. For events under 10^2 GeV, the network shows large variation in its predicted energies and the majority of events seem to be above the identity line. This behavior agrees with the results of the energy bias shown in Figure 19. Additionally, Figure 20b shows the same unweighted data, but normalized along E_{true}. The normalized data can be interpreted as a measure of the probability to predict a certain energy E_{pred} for a given true energy. One can identify a clear trend for the network to predict values of $\sim 10^2$ GeV below a true energy deposition of 10^2 GeV. This can be attributed to the high amount of events in the databases that deposited 10^2 GeV. Due to the events being simulated over a spectrum, the overall distribution of events with respect to the true energy deposition is not flat. The observation for Figure 20b can be interpreted as the network being biased towards a prediction of 10^2 GeV, especially for events with a true energy deposition below 10^2 GeV. However, it is still crucial for the network to train on the full energy range to ensure reliable predictions throughout that range. Due to the highly non-linear mapping from input to output it may very well be that a network, that is trained only on higher energy events, performs badly on low energy events.

To further quantify the performance, the relative resolution, as described in section 5.6, can be applied. The relative resolution by itself does not give a statement about how good the network predicted any single event, but describes the overall spread for all predictions in a given energy bin. In contrast to the standard deviation shown in Figure 19, the relative resolution is more robust to outliers, thus yielding a better measure for the overall spread.



(a) Unweighted, not normalized

(b) Unweighted, normalized along E_{true}

Figure 20: Comparison of unweighted heatmaps of the predicted energy against the true deposited energy. The identity line corresponds to a perfect prediction.

For each true deposited energy bin in Figure 20a, the relative resolution is calculated according to Equation 20. Therefore, unweighted data is used. Furthermore, the relative resolution is computed only for energy bins with more than ten events. The upper plot of Figure 21 shows the relative resolution of the trained model across all event topologies together with different literature curves. A smaller relative resolution corresponds to a smaller spread in the predicted data and therefore a better performance. The lower part of the plot shows the number of events in each bin.

For the energy range between 10 GeV and $\sim 10^2$ GeV, the relative resolution falls with increasing energy from $\sim$66% to about 30%. Figure 21 shows a local minimum of the relative resolution at 10^2 GeV before reaching a local peak, or a worsening in resolution, just before 10^3 GeV. After that local peak, the relative resolution gradually falls and takes values between 16% and 19% above 10^4 GeV.

Although the event counts decrease between 10^4 GeV and 10^8 GeV over more than one order of magnitude, the relative resolution stays almost the same for the entire energy range. This observation can be attributed to the increasing number of photons with increasing energy deposition, leading to more information for the model to go off of.

Established algorithms on energy reconstruction in IceCube are using either a more traditional "template" method, as described in section 5.1, in combination with a maximum likelihood algorithm [20] or the same DynEdge GNN backbone [19] as in this thesis, but for low energy events (1 GeV to 100 GeV).

The traditional method in IceCube differentiates between ν_e and ν_μ neutrino flavors, while only briefly addressing ν_τ interactions. The literature shows a relative resolution values largely below $\sim$10% for ν_e and ν_μ interactions and energies above 10^4 GeV.

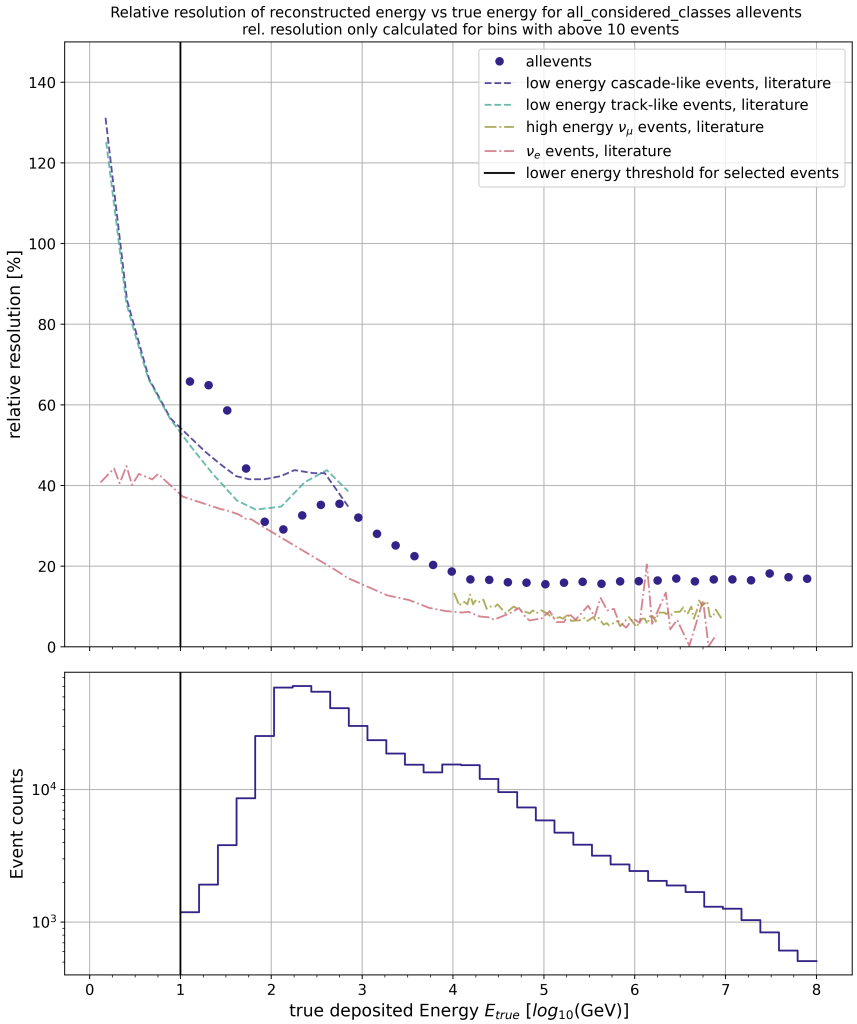


Figure 21: Plot of relative resolution for all events on the top. The plot was cut at a y-value of 150%. Events between the lower energy threshold and the first visible data point show resolutions above that cut value. The bottom shows how many events are in each bin on a logarithmic scale.

This can be seen in Figure 21, where the yellow-green dashed dotted line indicates ν_μ events and the pink-red dashed dotted line corresponds to the relative resolutions for ν_e. Traditional methods show indeed a slightly better performance than the algorithm developed in this thesis. However, the flavor-agnostic model of this thesis does not not differentiate between neutrino flavors to reconstruct the energy like traditional methods do. Furthermore, the traditional method is fairly time expensive and would not be feasible for real time analysis.

The DynEdge GNN model in the literature has a similar architecture to the model proposed in this thesis, but is trained on an equal amount of low energy events (1 GeV to 100 GeV) for all neutrino flavors [19]. The results of that study is represented by dark blue and cyan colored dashed lines in Figure 21. The literature work uses a

different definition of residuals for energy than the one outlined in Equation 19. Since the literature works in a lower energy regime, the residual was defined as

$$\mathcal{R}_{lit} = \frac{E_{pred} - E_{true}}{E_{true}} \cdot 100\,. \tag{22}$$

However the residual as defined in this thesis is more expressive to cover for the large energy range that the flavor-agnostic model is being trained on (between 10 GeV and 10^8 GeV). The differing definitions somewhat limit the comparability of the results. Two exemplary plots for the data used in this thesis with the definition in Equation 22 can be seen in the appendix (Figure 29 and Figure 30).

The established GNN model outperforms the model proposed in this thesis for energy depositions below $\sim 10^{1.72}$ GeV (the 4th bin from the left). A possible explanation for the worse performance may be that there are about 2 orders of magnitude less events in the bins below 10^2 GeV than in the bins with energy depositions at $\sim 10^2$ GeV for the flavor-agnostic model proposed in this thesis. This can be seen in the lower part of Figure 21. The low event count statistics leads to a better performance for the literature compared to the model proposed in this thesis. Furthermore, the events with energies between 10^2 GeV and 10^4 GeV were simulated using a spectral index of -1.5, resulting in a higher event count at about 10^2 GeV. Generally, low energy events, even muon tracks, have a short track length, thus potentially not reaching the detector. Additionally, low deposited energy, especially below 100 GeV [20], corresponds to less Cherenkov photons and therefore less signal. Those few photons may be absorbed in the ice before hitting detector or be so insignificant as to not trigger in the first place (except in DeepCore, where the threshold is approximately 10 GeV).

The decrease in relative resolution below 10^2 GeV and the local minimum at 10^2 GeV seem to be shifted towards the right compared to the literature values for both event types. Between 10^2 GeV and $\sim 10^3$ GeV, the model proposed in this thesis shows better performance, while both data display a peak in that energy range at roughly the same energy. The local peak is especially pronounced for the literature's track-like events, but was not mentioned in [19]. A possible explanation for the local peak, at least for muon tracks, could be that the model recognized the transition from primarily ionizing losses around 10^2 GeV to primarily radiative losses above 10^3 GeV [19][59]. The peak is not as pronounced for shower-like events. The reasons for the peak in the flavor-agnostic model proposed in this thesis are analyzed in the further sections.

Above $\sim 10^3$ GeV and below 10 GeV, a comparison of the literature models to the model developed in this thesis is not possible due to the different event energies used for training. The flavor-agnostic model in this thesis was strictly trained on events above 10 GeV, to only include events where DIS is the primary mechanism, whereas the literature work was focused on lower energy events.

Furthermore, the kind of data, or the pre-processing level, that was used is highly different when comparing this thesis to the literature. The traditional method did not specify which data level was used, but applies its algorithm on contained as well as on uncontained events [20]. The low-energy DynEdge uses Level7 data, with mostly contained events and practically no pure noise events [19]. As already discussed in section 5.2, the considered events in this thesis include at most Level2 data, but also non-Level2 data. Contained and uncontained events are included in the training and

testing databases. Despite training on potentially noisy or partially unfiltered events, the flavor-agnostic model shows comparable performance to previously proposed methods. Training on strictly Level2 data (or higher) may very well improve the network's performance and would have to be examined in further studies.

The model's performance on only Level2 data can be further analyzed by focusing on events that have passed specific filters. The filters considered for this thesis include the *CascadeFilter_13*, *MuonFilter_13*, *OnlineL2Filter_17*, *HESEFilter_15*, *MESEFilter_15*, and *HighQFilter_17*. Across all event topologies, the relative resolution for different filters can be seen in Figure 22. Every filter is encoded in a different color and the plot includes events that passed no filter (largely considered unusable data) as well. The lower part of the plot shows a histogram of the event counts for each filter in the corresponding color. Filters are not necessarily mutually exclusive. A single event can pass multiple filters, which is why a combined histogram of all filters would not add up the one at the bottom of Figure 21. Every filter covers a different energy range. While event that e.g. passed through HESE have energies above 10^3 GeV, the cascade filter's events stretch over the entire energy range from 10 GeV to 10^8 GeV. For each filter, the relative resolutions show generally a similar pattern to the combined plot in Figure 21.

Most filters with events below 10^2 GeV perform badly in that energy range with resolutions above 60%. There are not a lot of events in the mentioned low energy range, which is why low photon yield and low statistics may explain the sub par performance. The HESE and MESE filters show slightly better resolutions over their entire energy ranges compared to the other filters. The third best performing filter for energies below $\sim 10^5$ GeV is the HighQ filter. Above 10^5 GeV, HESE and MESE show resolutions of about 11% to 13%, while every other filter displays resolutions of about 16% to 18%.

In contrast to that, events that passed none of the listed filters show relative resolutions comparable to the overall picture below 10^2 GeV. The plot for data that passed no filter shows a local minimum at $\sim 10^2$ GeV, while possessing an event count of $\sim 5 \times 10^4$. The performance visibly worsens above 10^2 GeV, accompanied with a decline in event counts. The relative resolution of events that passed no filter probably largely dominates the performance of the network in that energy regime (below $\sim 10^3$ GeV). Events with low energy deposition yield only a few photons and are therefore less likely to trigger or pass any filter. This leads to a high event count of unfiltered data in the lower energy ranges. Since the data, or the histogram below, is not weighted, a high event count can be associated with dominating the overall performance. In that form, the argumentation is only true for data that passed no filter, since these data and filtered data describe exclusively different events. Between 10^3 GeV and 10^5 GeV, the resolution is almost three times as bad as the filtered or combined data, at between 45% and 68%. In the higher energy regime above 10^6 GeV, the events that passed no filter show relative resolutions at around 20%, but with larger variation than the filtered events. The data points corresponding to events that passed no filter display no immediately discernible pattern overall. Some high-energy events (above $\sim 10^6$ GeV) may have slipped through the filters because they were only partially contained or categorized as non starting events. Partially contained events, at, e.g., the edges of the detector can emit photons that travel outside the instrumented volume. Therefore, despite depositing energy in the ice, the detector cannot access a part the Cherenkov light pattern or any information about the photons that were not detected. Additionally,

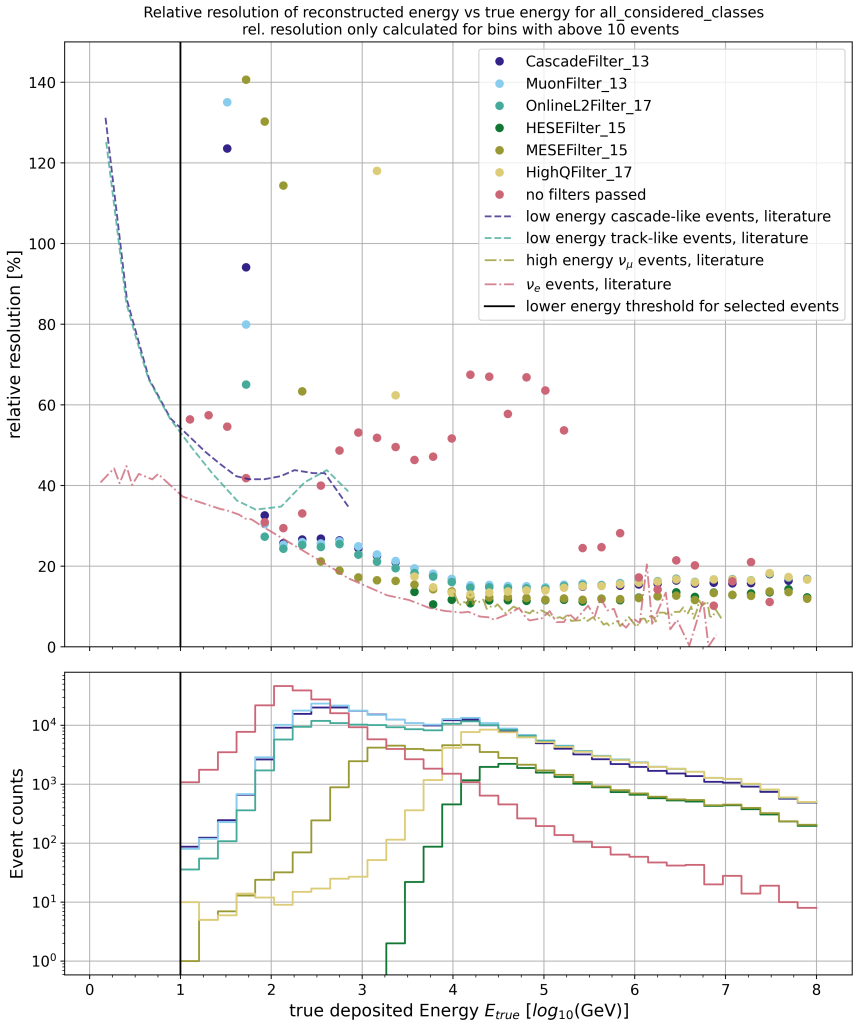


Figure 22: Plot of relative resolution for all events and different filters on the top. The plot was cut at a y-value of 150%. Events between the lower energy threshold and the first visible data point show resolutions above that cut value. The bottom plot shows how many events are in each bin on a logarithmic scale. The colors correspond to the filter of the plot above.

the filtering time constraints might have led the rather preliminary online filters to miss some events altogether. Despite that, these high energy events can be well resolved by the model due to their generally high photon count.

While the model shows a good overall performance, it was trained on ≈ 1.75 times more shower-like events than track-like events. That is because track-like events only result from ν_μ^{CC} interactions while all other possible interaction channels yield shower-like events. The equal flavor split, described in section 5.2, then leads to an under representation of track-like events. Therefore, the GNN performance on shower-like and track-like event topologies, respectively, is examined in the next sections.

6.3 Performance for shower-like events

Due to their more localized energy deposition, the deposited energy of shower-like events is generally easier to reconstruct when compared to track-like events. The signatures that can be classified as shower-like are the cascades and double bang event signatures described in Table 2. Even though the training and test databases contain for example about 6 times more *em_hadr_cascade*s than *double_bang_em* events, all shower-like events show a similar relative resolution profile to Figure 21. That profile is characterized as a decline in relative resolution from 10 GeV to 10^2 GeV, with a local minimum at 10^2 GeV. Afterwards, there is an increase in relative resolution percent points with a local peak just below 10^3 GeV, followed by and subsequent decline, i.e., an improvement in resolution until $\sim 10^4$ GeV and an approximately constant relative resolution for any higher energy depositions. The hadronic cascades and both double bang event signatures show a worsening in performance for energies above $\sim 10^7$ GeV. (For the plot for the hadronic cascade see appendix Figure 31)

Generally, all shower-like events lead to a similar detector response: A spherically extended pattern. While the pattern is alike, the total photon yield may differ between concrete event signatures. Despite that, the light yield for different signatures at the same energy is more similar than the light yields for the same signature at largely different energies. This explains why the performance is similar for a given true deposited energy across all shower-like events. For low energy double bang events, the tauon production and decay vertex may be spatially very close together due to the tauon's short lifespan. As a result, the vertices are hard to distinguish for the IceCube DOMs and the event signature looks similar to a hadronic shower, electromagnetic shower from a ν_e^{CC} or even a low energy ν_μ^{CC} interaction [24]. Double bang events start to form two well separated showers with spatial distance above 100 m at energies above $\sim 2 \times 10^6$ GeV [24]. The extend of such a double bang event may distort the energy reconstruction, especially since it is a rather rare phenomenon and the databases only contain a few hundred events in that energy regime. (as can be inferred from Figure 32 in the appendix) In addition to the $\sim$ 1.5 times higher number of electromagnetic cascade events in the databases, these facts might explain the worsening performance for double bang events at higher energies compared to the electromagnetic cascades. The electromagnetic cascades show below 10 events in the lower end of the investigated energy range. This is because the neutrino simulation is starting at 10^2 GeV and any ν_e^{CC} interaction would deposit the initial neutrino energy completely, unless the neutrino interacted beforehand (NC).

A plot distinguishing between different filters, exemplary for the double bang into hadronic cascade, can be seen in Figure 23. The different filters show great resolutions between 30% and 15%, even being able to compete with the ν_e template literature method at lower energies. The events that passed no filter, again, show a similar behavior in Figure 23 to the overall relative resolution in Figure 22, with a visible worsening in performance above a local minimum at 10^2 GeV. One could speculate that above the 10^2 GeV point more unfiltered data corresponds to partially contained events and therefore did not pass any filter. This increase in relative resolution leads to the apparent local peak even for events that are not track-like. Since that peak is to be expected for track like events, it might be an artefact of additionally training on

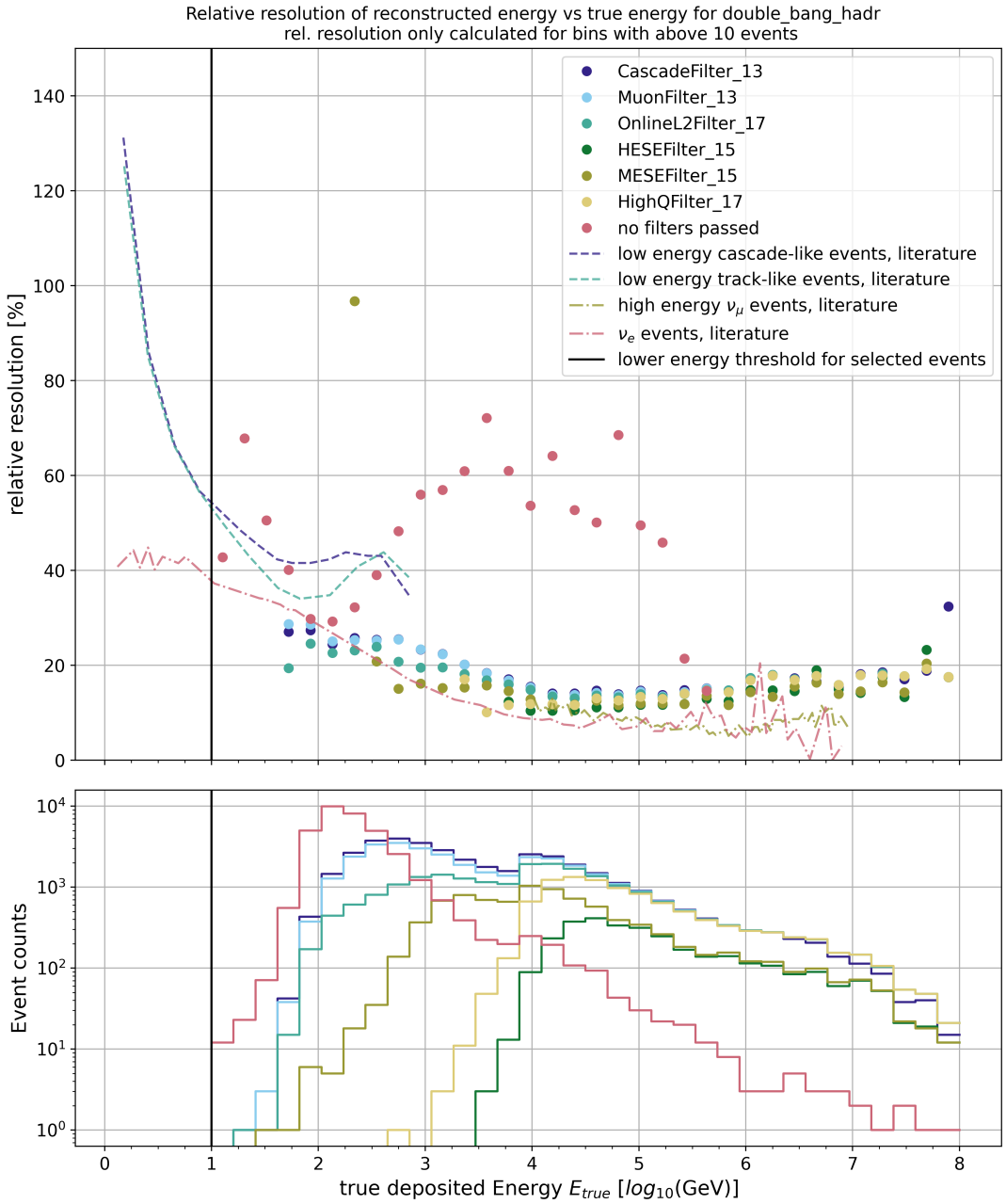


Figure 23: Relative resolution for double bang into hardronic cascade events for different filters. The bottom shows how many events are in each bin on a logarithmic scale. The colors correspond to the filter of the plot above.

track-like data. Due to the low photon yields for these unfiltered events, the network might treat them similar to the low photon yielding track-like events. The significant event counts for unfiltered events in that energy regime leads to a pronounced peak in the overall picture.

To investigate how well the network performed in dependence of the event's position inside the detector volume, the relative error with respect to the first vertex position was used for shower-like events. The relative error offers ease of interpretation. The IQR is not suitable since multiple events with vastly different energies could have interacted at spatially close points inside the detector. The relative error was calculated as follows:

$$\text{rel. error} = \frac{E_{pred} - E_{true}}{E_{true}} \cdot 100\% \,. \tag{23}$$

Firstly, the relative error with respect to the true first vertex r, in polar coordinates, is examined and an exemplary plot for the hadronic cascade is shown in Figure 24. The plot differentiates between two energy regions: Events between 10 GeV to 10^4 GeV and 10^4 GeV to 10^8 GeV. Additionally, the data points for the full energy range is shown. Events with a first vertex somewhat outside the detector are shown as well. These interactions are also simulated to account for uncontained events outside the actual detector. A clear trend of higher relative errors towards the detector edges is recognizable, with larger relative errors outside the detector volume. This is expected behavior. Additionally to the trend, while the event count is approximately an order of magnitude bigger for events below 10^4 GeV, the relative error fluctuates strongly, leading to error values between ∼40 and ∼70%. The structure that one can see in the data points for energies below 10^4 GeV, which resembles a zick-zack pattern is likely an effect of the used binning and the detector density. For r values close to a string the relative error is expected to be better. When changing r, one, on average, either approaches or withdraws from a string, resulting in this peculiar pattern. The higher energy regime, above 10^4 GeV, displays significantly less fluctuations.

While all other shower-like events show a similar behavior with respect to r, the double bang into electromagnetic cascade events display slightly better relative errors, but with similar fluctuations for low energy events, leading to values between 25% and 50%. The other two event signatures, electromagnetic cascade and double bang into hadronic cascade, show less fluctuations for low energies. The lower part of Figure 24 shows the event counts for the different energy ranges. The increase of event counts with increasing distance from the detector center is expected, as a larger value of r corresponds to a larger instrumented circular shell area. (see, e.g., appendix Figure 33)

Secondly, the relative error with respect to the true first vertex z is analyzed. Hereby, z refers to the detector height. The null point for z is approximately at the middle of the instrumented volume. An exemplary plot of the relative error against the z position can be seen in the upper part Figure 25 for the electromagnetic cascade. The red dash dotted line represents the absorption length of the antarctic ice [18]. Different colors encode different energy regimes. The relative error for the electromagnetic cascade increases for all energy ranges towards the bottom and the top of the detector, which is to be expected, due to the increase of partially contained events at the edges. Additionally, the model predictions show larger relative errors in the dust layer, where the photon absorption length is smaller. Therefore, energies of events with a first vertex inside the dust layer are harder to reconstruct accurately. For energies between 10 GeV to 10^4 GeV, the relative error is at about 20% both below and above the dust layer, except for the edges. The relative error values for this energy range exhibit a minimum at $z \approx -350\,\text{m}$. That minimum coincides with a maximum of the literature absorption

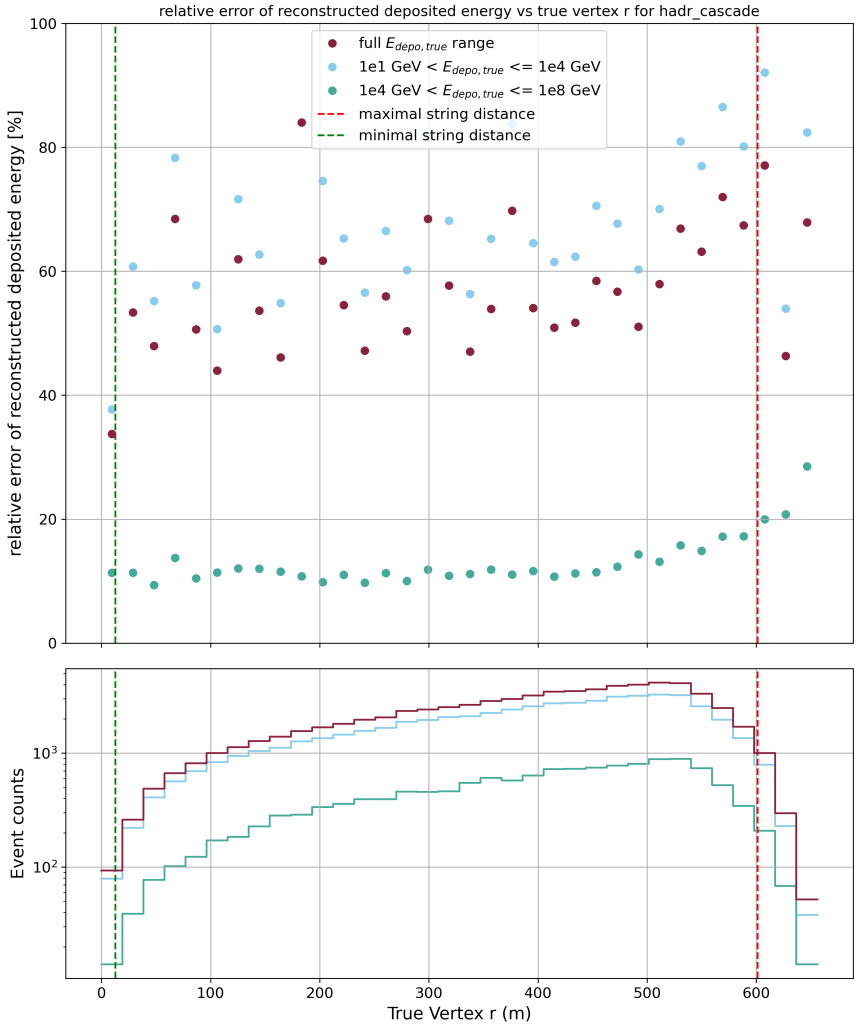


Figure 24: Plot of the relative error with respect to the true first vertex r for the hadronic cascade (upper). Different colors indicate different energy ranges. Event counts for these different energy ranges with matching colors (lower).

lengths in the instrumented volume. Above and below the dust layer, the events with an energy above 10^4 GeV show relative errors of $\sim$10%, increasing at the edges. The lower part of Figure 25 shows the event counts in dependence of the z position. The colors correspond to the same energy ranges as the upper part of the plot. While the event count for energy depositions above 10^4 GeV stays constant across the entire depth of the instrumented volume, event counts for energies below 10^4 GeV have a minimum at the dust layer. This can be explained by the lower amount of photons for lower energy events not being able to get past the dust layer impurities as easily. A higher photon count and the larger extend of higher energy shower-like events, on the other hand, result in IceCube DOMs still being able to reliably detect events in the dust layer.

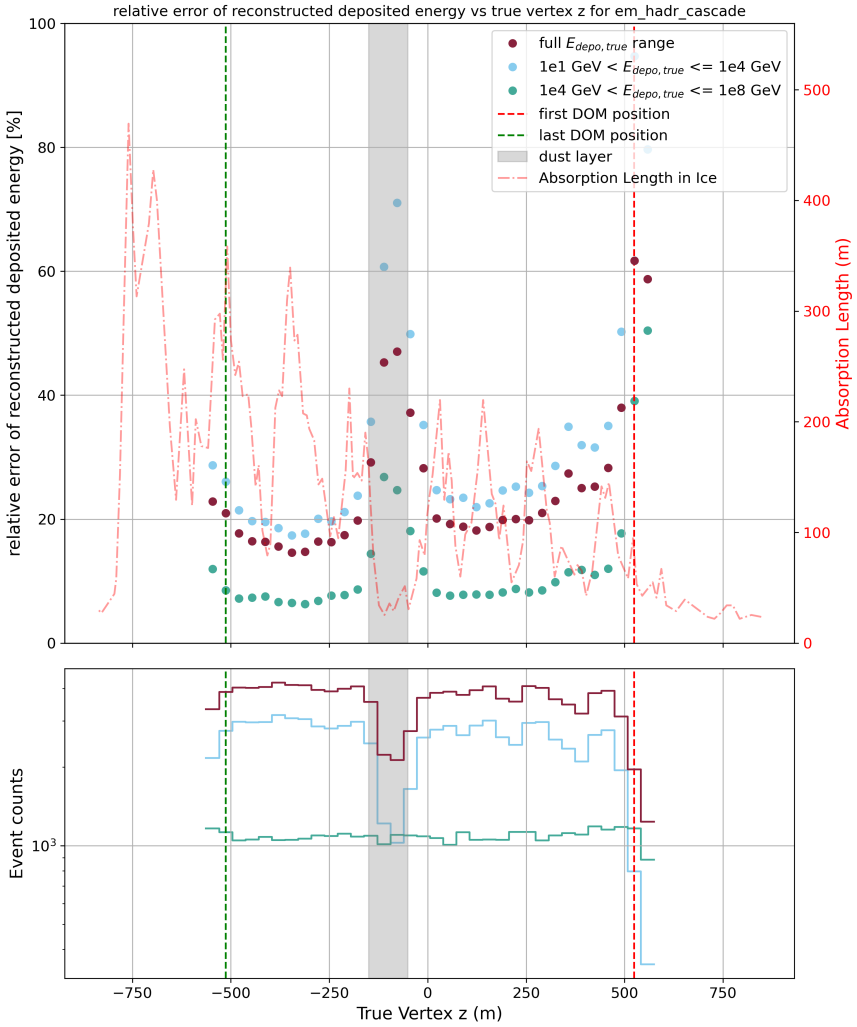


Figure 25: Plot of the relative error with respect to the true first vertex z for the electromagnetic cascade (upper). The red dash dotted line indicates the absorption length in the ice, taken from [18]. Different colors indicate different energy ranges. Event counts for these different energy ranges with matching colors (lower).

All the other shower-like event topologies display a similar behavior, while the hadronic cascade and the double bang into electromagnetic cascade show fluctuations. (see, e.g., appendix Figure 34)

The large fluctuations in relative error with respect to the first vertex coordinates suggest that the hadronic cascade and the double bang into electromagnetic cascade events may not be sufficiently represented in the training dataset.

In general, the flavor-agnostic GNN demonstrates good modeling of the dust layer and ice absorption characteristics. Additionally, it shows no unexpected detector geometry dependencies for the true first vertex positions.

6.4 Performance for track-like events

Due to the stochastic nature of the energy loss process, energy depositions of track-like events are generally harder to reconstruct than shower-like events. Track-like event signatures that were considered in this thesis include contained, starting, stopping and throughgoing tracks with different event counts (see Table 1).

There are no simulated contained or stopping tracks with an energy deposition above 10^5 GeV. Throughgoing and starting tracks cover the entire energy deposition range. Low-energy throughgoing events are likely tracks at the edges of the detector that do not cover as much of the instrumented volume. Unlike throughgoing events, energy depositions for starting track events include hadronic cascades. The energies of the muon and the associated cascade may vary greatly across different starting track events.

The relative resolutions for different filters are exemplary plotted for starting track events in Figure 26. Starting tracks display the expected local peak in relative resolution below 10^3 GeV for the Cascade, Muon, and OnlineL2 filters. The peak however is shifted towards slightly larger true deposited energy values. MESE also has events in that energy regime but does not show the characteristic local peak for some reason. For energies above $\sim 10^4$ GeV, the relative resolution for starting tracks is between $\sim$10% and $\sim$25%. One can see at slight worsening in performance above about 10^6 GeV. A possible explanation may include low statistics and the fact that the extend of a high energy muon track poses a special challenge to the network.

Furthermore, the events that passed no filters exhibit the same behavioral pattern that could already be observed for shower-like events: local minimum at $\sim 10^2$ GeV due to high statistics, despite low photon amount, with worse performance below $\sim 10^2$ GeV due to potentially low light yield. Additionally, events that passed no filter with energy depositions above $\sim 10^2$ GeV might show a worse performance due to their partially contained signature. The missing Cherenkov light can lead to less reliable predictions.

The other track-like event signatures show a similar behavior below $\sim 10^5$ GeV with different relative resolution values. Throughgoing tracks, however, show relative resolutions of mostly above $\sim$20% across all filters. (see appendix Figure 35)

The relative error with respect to the first vertex positions is not as expressive for track-like events since, e.g., throughgoing tracks have no interaction vertex in the instrumented volume. Instead, the relative error with respect to the directional properties of the detector primary was considered in this thesis. The detector primary is the particle that first entered the detector volume. In the case of throughgoing and stopping tracks, the detector primary would be the incoming muon, while for contained and starting tracks it would be the neutrino. The direction can be parameterized with azimuth and zenith angles. A zenith value of 0° would correspond to a track originating from directly above the detector.

The relative error plots with respect to cos(zenith) and azimuth are shown for throughgoing tracks in Figure 27 and Figure 28, respectively.

Throughgoing tracks show relative errors of 15% to 40% with outliers for events above $\sim 10^4$ GeV with respect to cos(zenith). Similarly, the plot for the relative error in

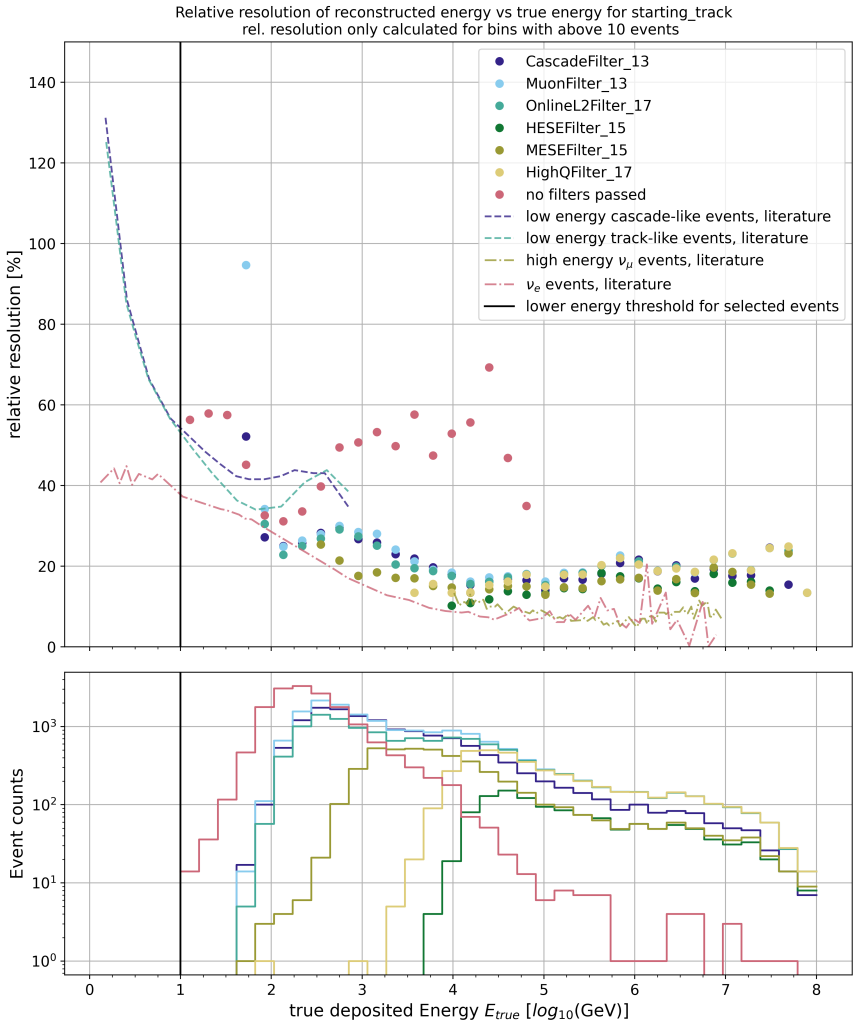


Figure 26: Relative resolution for starting track events for different filters. The bottom shows how many events are in each bin on a logarithmic scale. The colors correspond to the filter of the plot above

dependence of the azimuth angle in Figure 28 shows smaller errors between 20% and 40% for energies above 10^4 GeV. Events below $\sim 10^4$ GeV energy deposition display larger relative errors at about 100% with outliers for both azimuth and cos(zenith). The higher relative error for low energy throughgoing events can be attributed to the tracks being at the edges of the detector. For higher energies, as with the resolution above, the extend of the track may complicate the reconstruction.

The higher event counts for tracks originating from above the detector (Figure 27, lower plot) are because more of these track-like events have been simulated. This is not considered physical reality and would be accounted for when using the simulation weights.

Similar to throughgoing tracks, stopping tracks show large relative errors with respect to the direction: Between 100% and 10 000%, with outliers, for events below

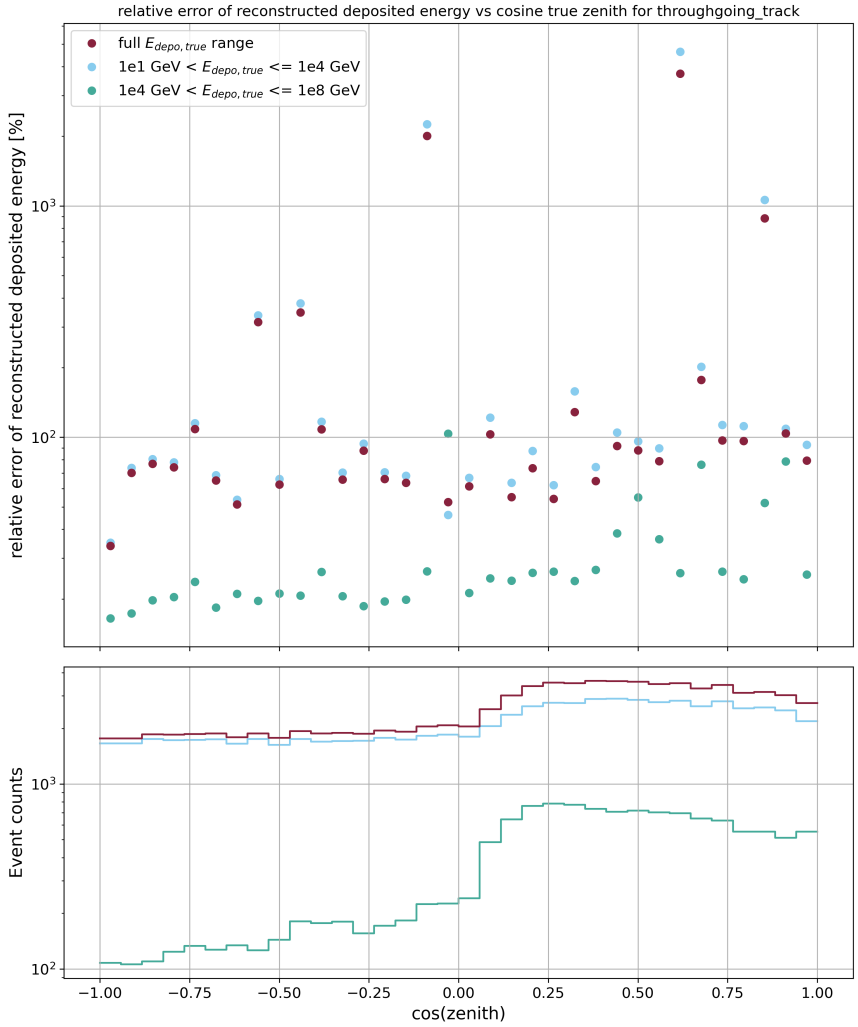


Figure 27: Plot of the relative error with respect to cos(zenith) for throughgoing track events (upper). Different colors indicate different energy ranges. Event counts for these different energy ranges with matching colors (lower).

10^4 GeV. This is because muons in stopping tracks are at the end of their lifetime and produce low amounts of Cherenkov photons.

Starting and contained tracks, on the other hand, show smaller relative errors, between 17% and ∼20% above 10^4 GeV. At lower energies below 10^4 GeV the relative error amounts to ∼30% to ∼40% with outliers. All these relative error values for starting and contained tracks are comparable to the values for shower-like events. This may be attributed to the associated cascades for these two event signatures being generally easier to reconstruct. (see appendix Figure 37 and Figure 36)

The differences in relative errors between the track-like signatures as a whole may also be due to the detector primary particle being different, despite the correlation between detector primary and the initial neutrino.

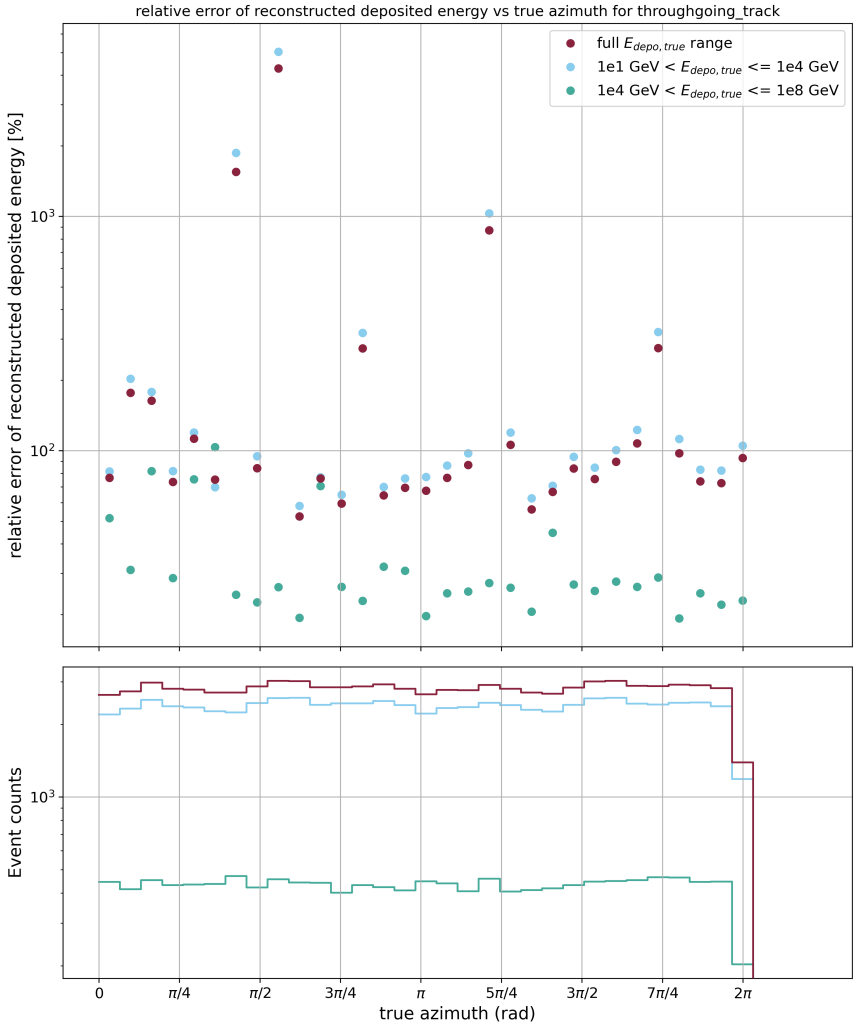


Figure 28: Plot of the relative error with respect to the true azimuth for throughgoing tracks events (upper). Different colors indicate different energy ranges. Event counts for these different energy ranges with matching colors (lower).

In general, track-like events have larger relative errors and a bigger spread in prediction. Despite that, the model shows no dependence on detector geometry for performance.

7 Conclusion and Outlook

In this thesis, a graph neural network (GNN) was used on IceCube simulation data to reconstruct the deposited energy of a neutrino interaction independent of the neutrino flavor. Therefore, the network was trained on 3.645×10^6 simulated events. After the training period, the network performed predictions on 4.5×10^5 simulated, unseen events. Both the training and testing databases contained shower-like and track-like events, with true deposited energy values between 10 GeV to 10^8 GeV. The databases included Level2 data, as well as events that passed none of the IceCube online filters.

The GNN was implemented using the GraphNet python framework. Each event was represented as one graph, whereby each IceCube digital optical module (DOM) constitutes a single node in the graph. Each node holds attributes, describing the event like the coordinates and the charge or the time distribution for example. The architecture that was used is called DynEdge and is included in the GraphNet installation.

In the analysis, it could be shown that the energy reconstruction of the flavor-agnostic GNN model developed in this thesis shows no bias towards any neutrino flavor. However, the model seems to prefer making predictions of 10^2 GeV for true deposited energy values below 10^2 GeV. This systematic bias was largely attributed to the high event counts for energy depositions of 10^2 GeV. To quantify the performance of the network, the relative resolution and the relative error of reconstructed energy were determined. For all considered events, the relative resolution shows a decline from ∼60% at ∼10 GeV to ∼30% at 10^2 GeV, with a local minimum at 10^2 GeV. Afterwards, there is an increase in relative resolution percent points with a local peak just below 10^3 GeV, followed by a subsequent decline, i.e., an improvement in resolution until ∼10^4 GeV, reaching a relative resolution just below 20%. For energy depositions above 10^4 GeV, a relative resolution of ∼16% to ∼19% was achieved.

Additionally, the relative resolution for different filters was investigated. Events that passed a filter showed good relative resolutions between 11% and 18% above ∼10^5 GeV depending on the filter, while events that passed no filters worsen the performance in the overall picture. The local peak just below 10^3 GeV was partially attributed to the impact of events that passed no filter, due to the high count of these events in for energies below 10^3 GeV. Another possible reason for this characteristic peak could be the transition region between primarily ionizing losses to primarily stochastical losses above 10^3 GeV for muons in ice. Generally, shower-like events are easier to reconstruct than track-like events. While both show similar relative resolution patterns, shower-like events have smaller resolution and track-like events show slightly worse performance. This was expected since shower-like signatures are localized, whereas track-like signatures are rather extended and have stochastical energy losses.

The results are at comparable values to established literature algorithms, despite the literature being either focused on a smaller energy range or differentiating between neutrino flavors.

Furthermore, the relative error with respect to the first vertex position for shower-like events, and with respect to arrival direction for track-like events were investigated. Shower-like events with respect to r (in polar coordinates) showed an increase in the relative error for larger distances from the detector center. This was expected since a

larger r corresponds to a larger included circular shell area. Shower-like events with respect to z showed a good modeling of the IceCube dust layer. Track-like events were inspected with respect to the azimuth angle and $\cos(\text{zenith})$. They showed generally larger errors than showers but with no dependence on the track-direction.

To sum up, the GNN model developed in this thesis shows no biases concerning neutrino flavor, neutrino direction, or detector geometry and is comparable to other literature methods.

With a prediction rate of $\sim$0.29 kHz on one GPU core, the flavor-agnostic model has potential as a low level IceCube filter when run in parallel on multiple filtering clients. As a filter the model can be used to reconstruct event energies, and trigger potential multi-messenger follow ups in the alert system without the need for a prior classification.

Further improvements on the model architecture can be made by increasing the amount of trainable parameters, preprocessing the data to simplify the loss landscape or by helping the network by weighting the data. Additionally, the bias towards 10^2 GeV predictions could be mitigated by choosing a flatter event distribution with respect to the true deposited energy.

The entire code used to implement the GNN and perform analysis can be found on github [60] or the ECAP gitlab [61].

Acronyms

AGN active galactic nucleus. 4

ANN artificial neural network. 17, 18, 22, 29

CνB cosmic neutrino background. 3

CC charged current. 5

CMB cosmic microwave background. 3, 4

DIS deep inelastic scattering. 4, 5, 6, 29, 47

DL deep learning. 17, 21

DOM digital optical module. 1, 9, 10, 11, 29, 30, 32, 34, 35, 36, 42, 50, 53, 59

GNN graph neural network. , 1, 17, 20, 22, 23, 29, 30, 32, 33, 34, 35, 39, 40, 41, 42, 45, 46, 47, 49, 54, 59, 60

IQR interquartile range. 39, 40, 52

KNN k-nearest neighbor. 23, 35, 37

ML machine learning. 17, 21, 29, 30, 32, 33, 39

MLP multilayer perceptron. 18, 19, 22, 23, 37, 38

NC neutral current. 5, 12, 30, 50

PMT photomultiplier tube. 9, 10, 15, 34, 36

ReLU rectified linear unit. 19, 37

SGD stochastic gradient descent. 25

SM Standard Model of particle physics. 1, 3

A Appendix

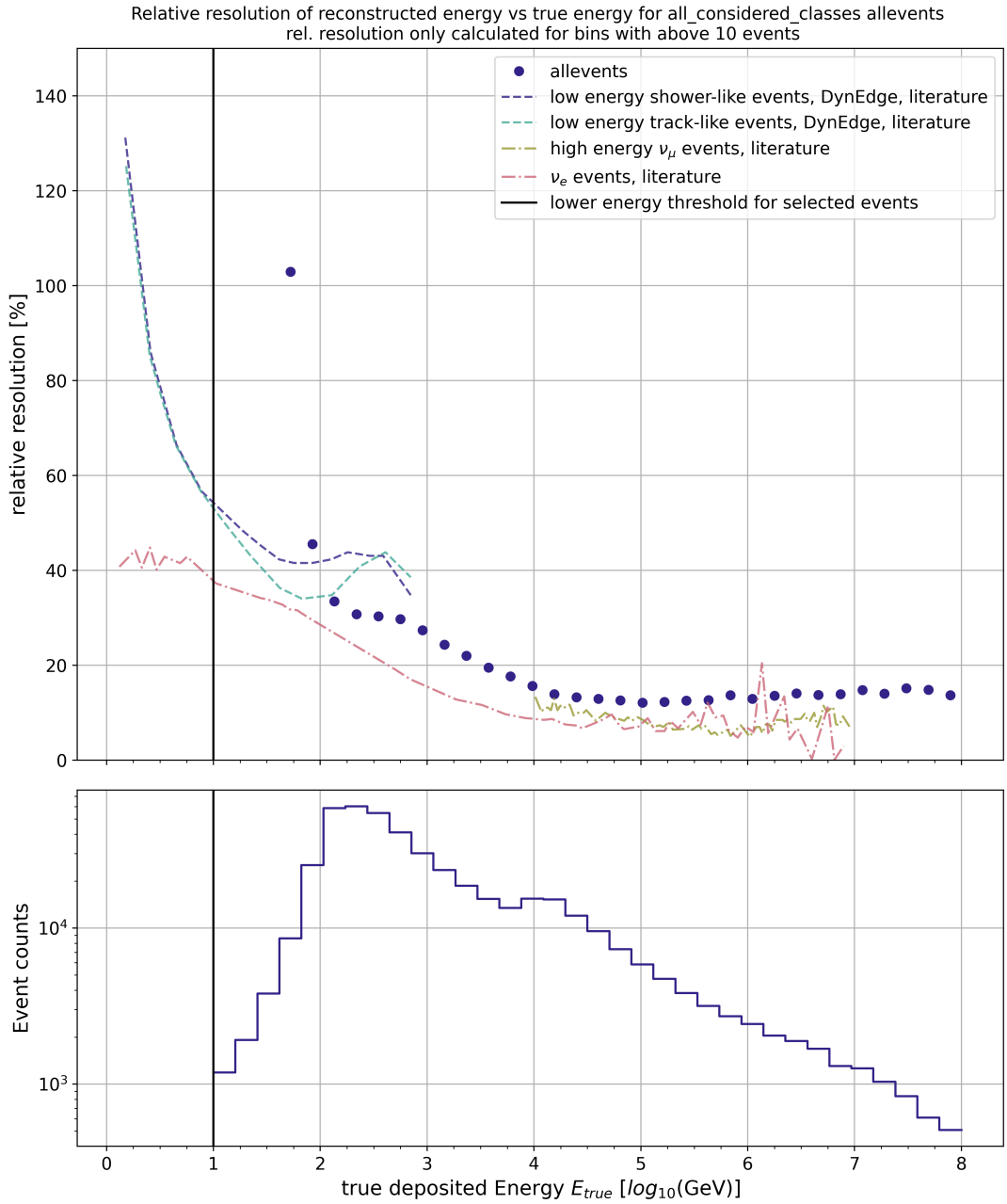


Figure 29: Plot of relative resolution for all events on the top. The bottom shows how many events are in each bin on a logarithmic scale. For this plot the literature definition as in Equation 22 was used. One can see a plateau instead of a peak below 10^3 GeV. The peak therefore also seems to be a result of the residual definition.

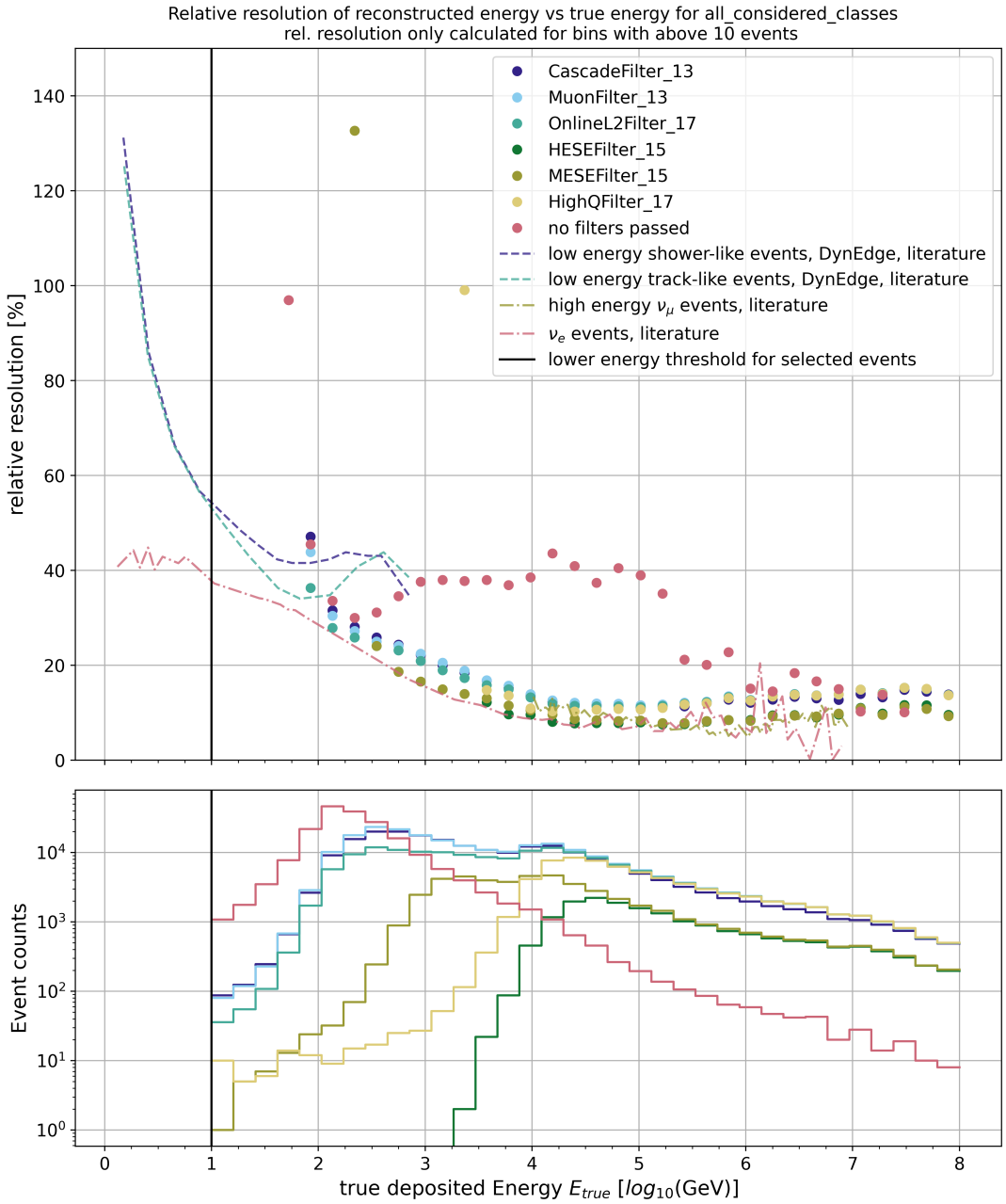


Figure 30: Plot of relative resolution for all events and different filters on the top. The bottom shows how many events are in each bin on a logarithmic scale. For this plot the literature definition as in Equation 22 was used. (All following plots use the definition as provided in the thesis)

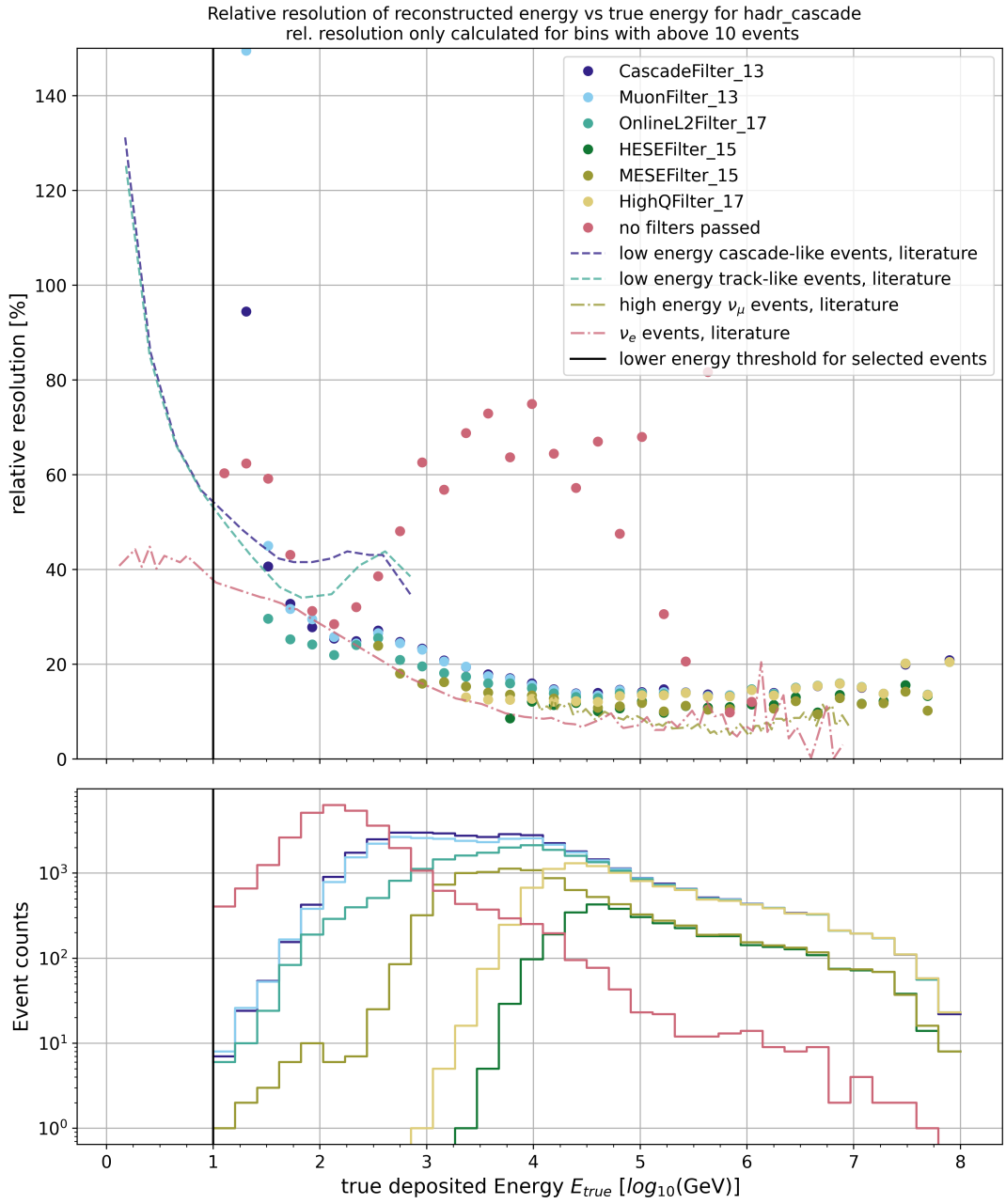


Figure 31: Plot of relative resolution for hadronic cascade on the top. The bottom shows how many events are in each bin on a logarithmic scale.

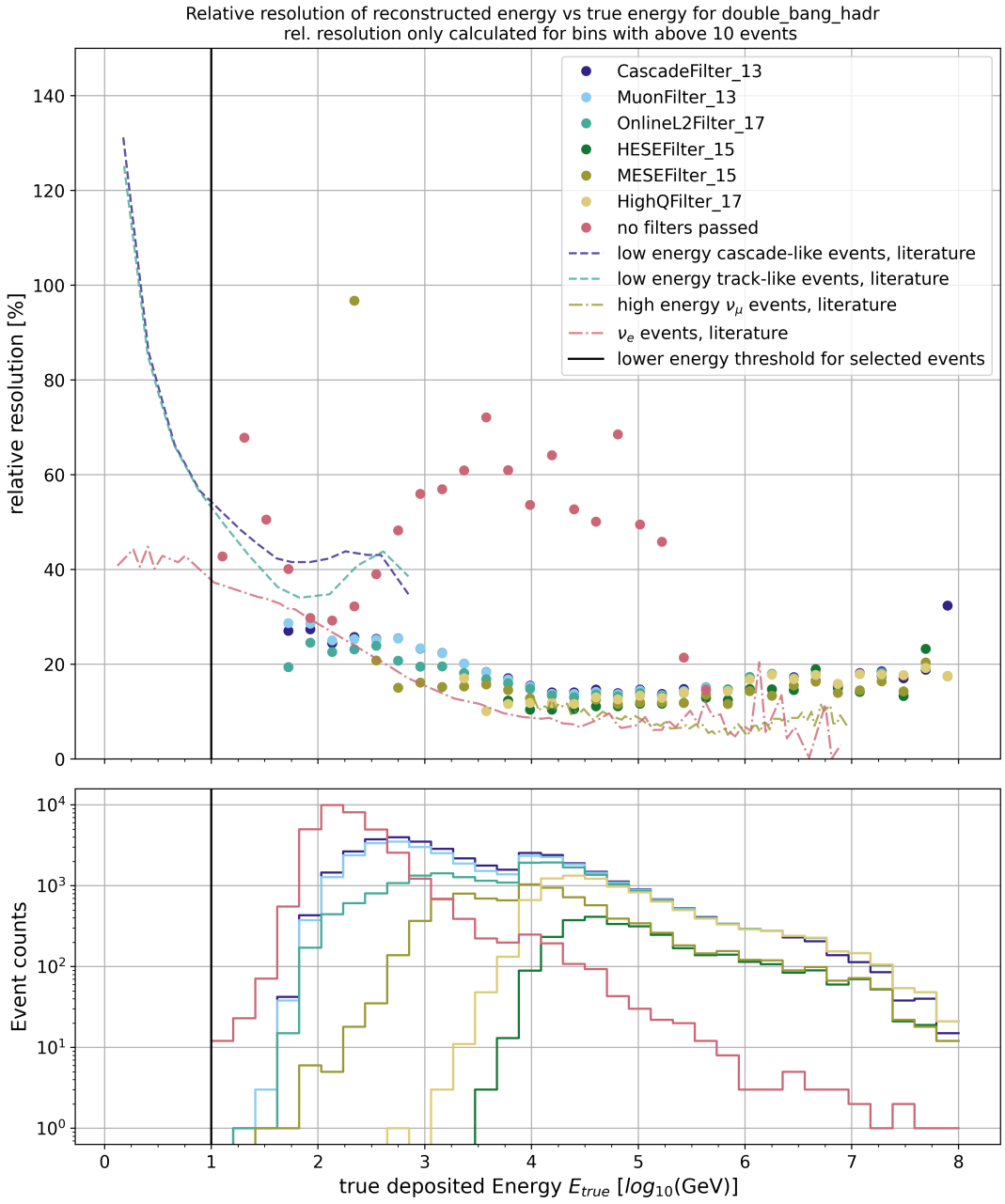


Figure 32: Plot of relative resolution for double bang into hadronic cascade on the top. The bottom shows how many events are in each bin on a logarithmic scale.

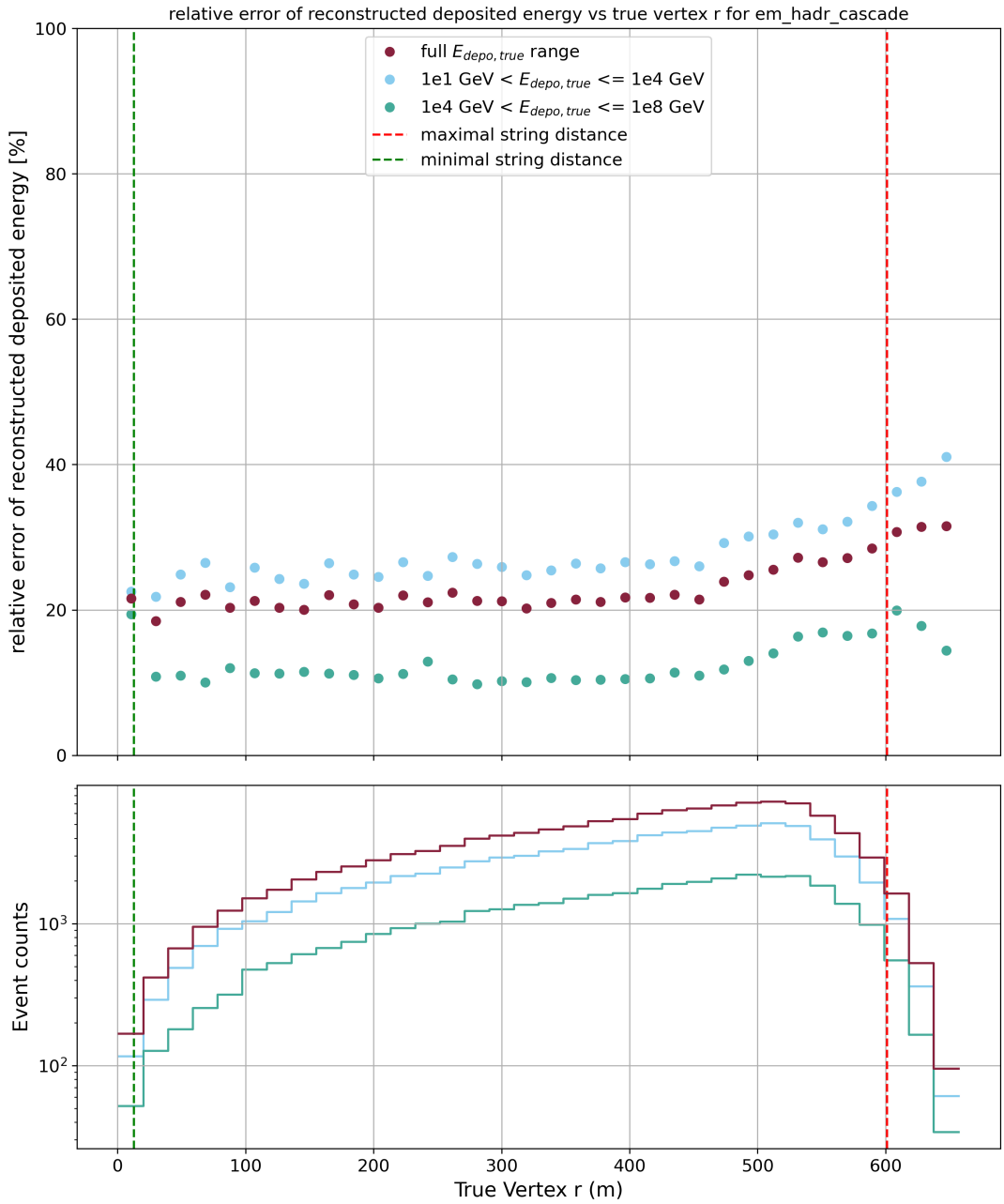


Figure 33: Plot of the relative error with respect to the true first vertex r for the electromagnetic cascade (upper). Different colors indicate different energy ranges. Event counts for these different energy ranges with matching colors (lower).

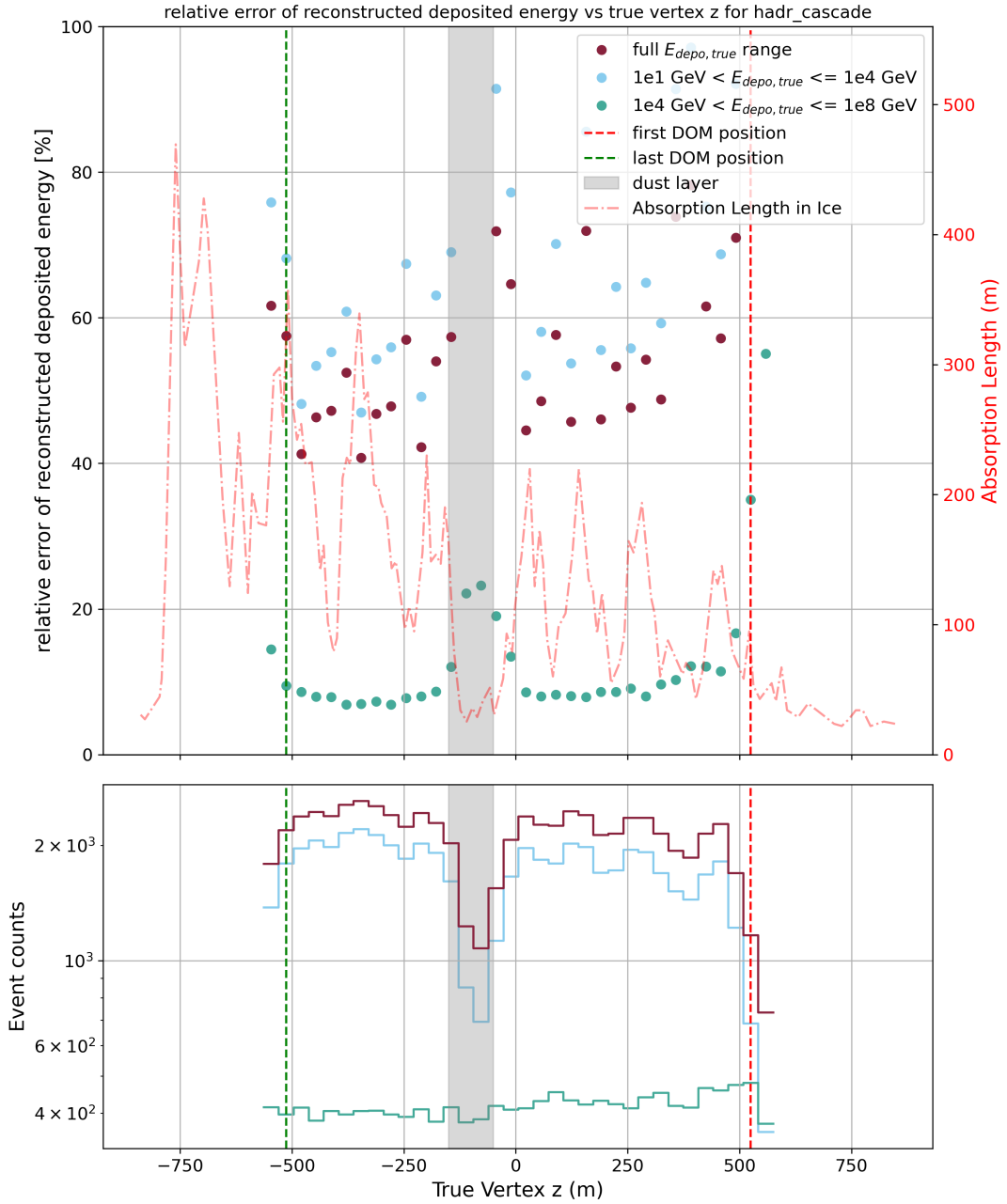


Figure 34: Plot of the relative error with respect to the true first vertex z for the hadronic cascade (upper). Different colors indicate different energy ranges. Event counts for these different energy ranges with matching colors (lower).

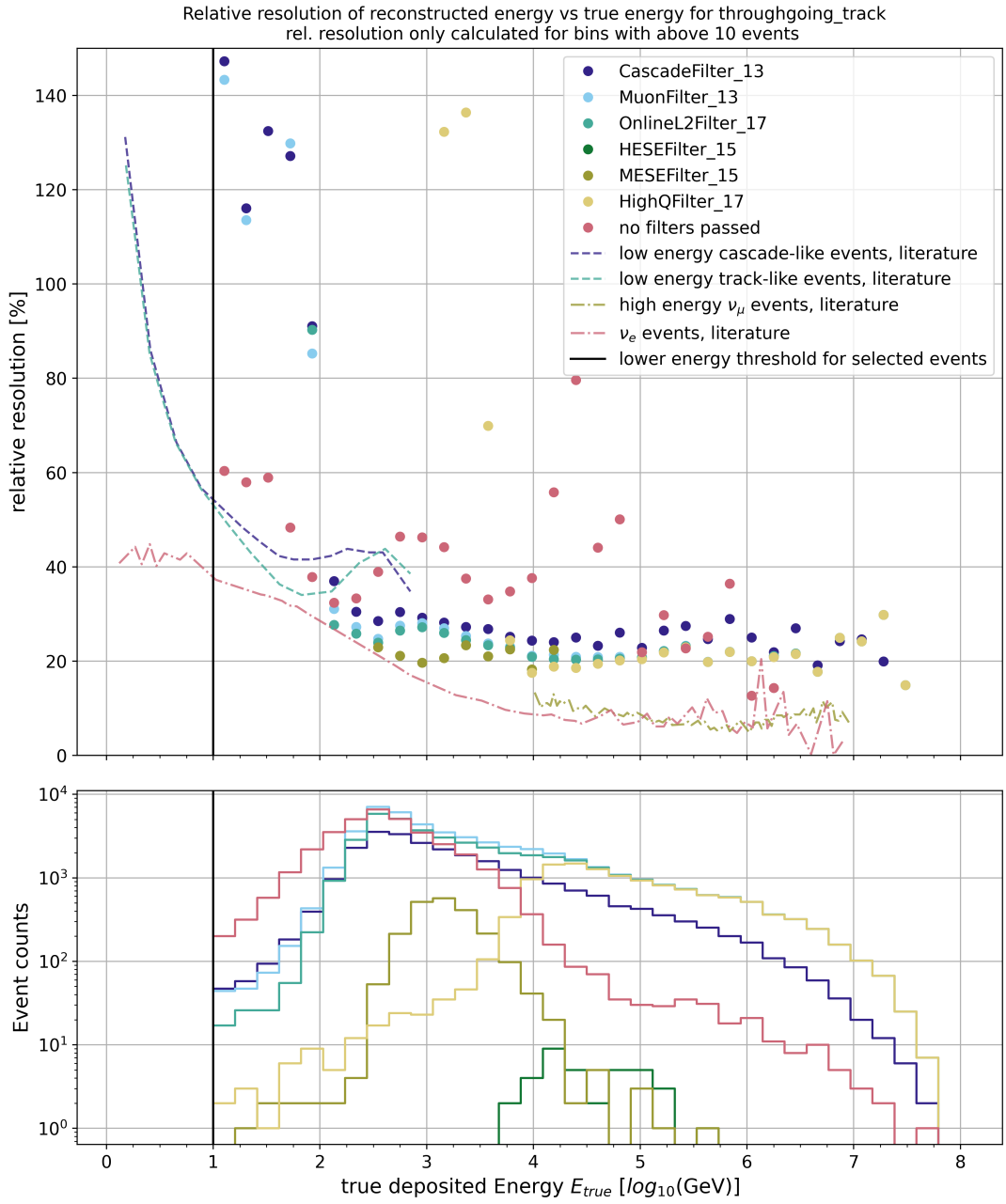


Figure 35: Plot of relative resolution for throughgoing track on the top. The bottom shows how many events are in each bin on a logarithmic scale.

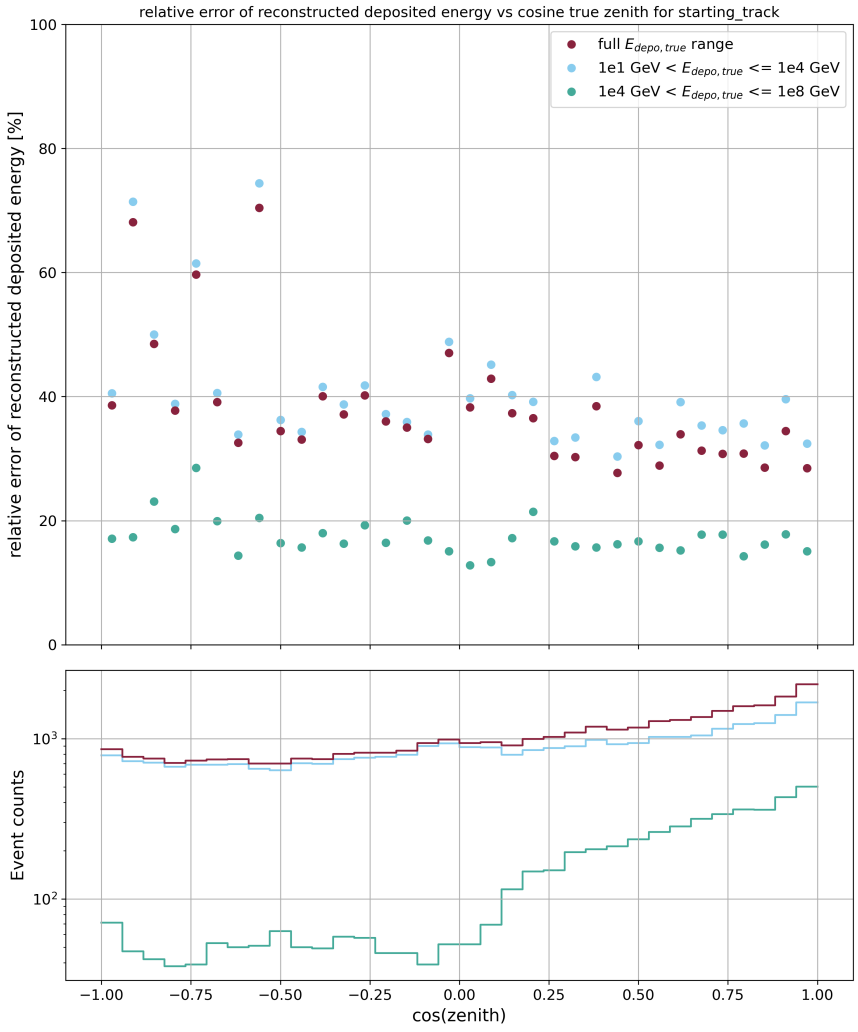


Figure 36: Plot of the relative error with respect to cos(zenith) for starting track events (upper). Different colors indicate different energy ranges. Event counts for these different energy ranges with matching colors (lower).

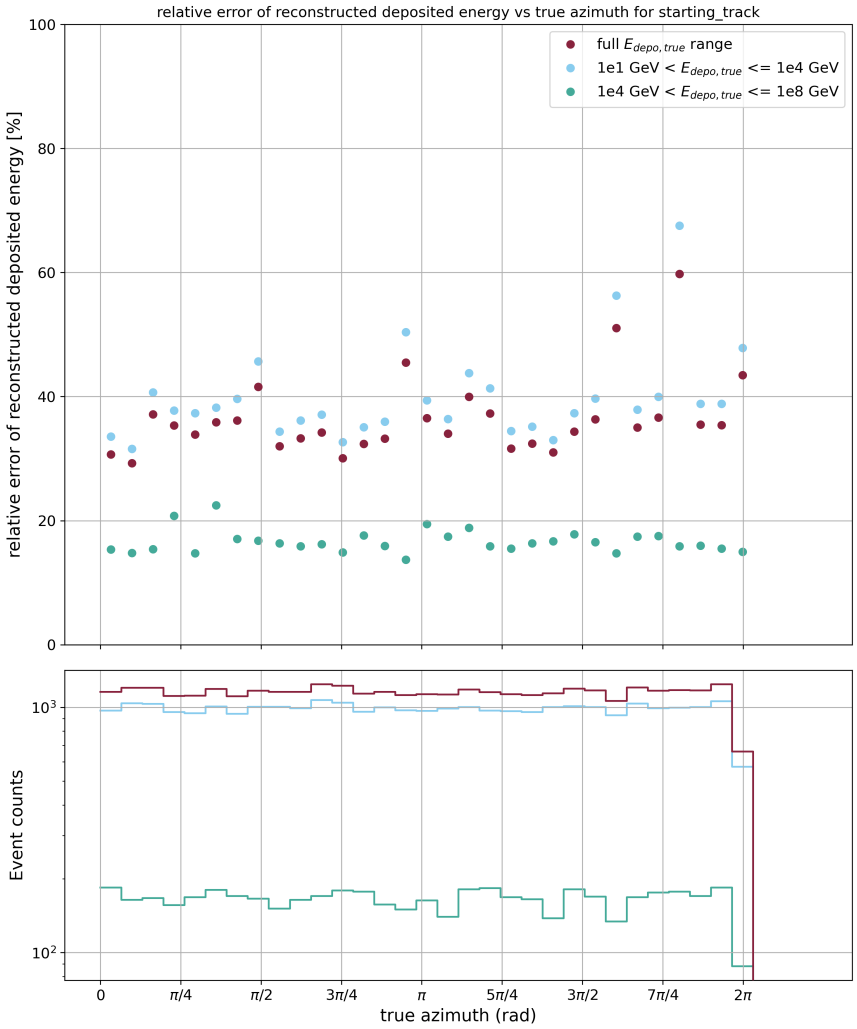


Figure 37: Plot of the relative error with respect to the true azimuth for starting tracks events (upper). Different colors indicate different energy ranges. Event counts for these different energy ranges with matching colors (lower).

Bibliography

1. Pauli, W. *Pauli letter collection: letter to Lise Meitner* Typed copy. https://cds.cern.ch/record/83282.
2. Gaisser, T. K., Engel, R. & Resconi, E. *Cosmic Rays and Particle Physics* (2016).
3. Zuber, K. *Neutrino Physics* (2020).
4. Aartsen, M. G., Ackermann, M., Adams, J., Aguilar, J. A., *et al.* The IceCube Neutrino Observatory: instrumentation and online systems. *Journal of Instrumentation* **12,** P03012. arXiv: 1612.05093 [astro-ph.IM] (Mar. 2017).
5. Vitagliano, E., Tamborra, I. & Raffelt, G. Grand unified neutrino spectrum at Earth: Sources and spectral components. *Reviews of Modern Physics* **92,** 045006. arXiv: 1910.11878 [astro-ph.HE] (Oct. 2020).
6. Katz, U. F. & Spiering, C. High-energy neutrino astrophysics: Status and perspectives. *Progress in Particle and Nuclear Physics* **67,** 651–704. arXiv: 1111.0507 [astro-ph.HE] (July 2012).
7. Greisen, K. End to the Cosmic-Ray Spectrum? *Phys. Rev. Lett.* **16,** 748–750. https://link.aps.org/doi/10.1103/PhysRevLett.16.748 (17 Apr. 1966).
8. Zatsepin, G. T. & Kuzmin, V. A. Upper limit of the spectrum of cosmic rays. *JETP Lett.* **4,** 78–80 (1966).
9. Formaggio, J. A. & Zeller, G. P. From eV to EeV: Neutrino cross sections across energy scales. *Reviews of Modern Physics* **84,** 1307–1341. arXiv: 1305.7513 [hep-ex] (July 2012).
10. Mahn, K., Marshall, C. & Wilkinson, C. Progress in Measurements of 0.1-10 GeV Neutrino-Nucleus Scattering and Anticipated Results from Future Experiments. *Annual Review of Nuclear and Particle Science* **68,** 105–129. arXiv: 1803.08848 [hep-ex] (Oct. 2018).
11. Schindler, S. *Comparison of Directional Reconstruction Algorithms of Muons using the Moon Shadow in IceCube* MA thesis (FAU Erlangen-Nürnberg, 2020). https://ecap.nat.fau.de/wp-content/uploads/2021/04/MSc_SebastianSchindler_2020.pdf.
12. Workman, R. L., Burkert, V. D., Crede, V., Klempt, E., *et al.* Review of Particle Physics. *Progress of Theoretical and Experimental Physics* **2022,** 083C01 (Aug. 2022).
13. Hofestädt, J. *Measuring the neutrino mass hierarchy with the future KM3NeT/ORCA detector* PhD thesis (FAU Erlangen-Nürnberg, 2017). https://ecap.nat.fau.de/wp-content/uploads/2017/04/Dissertation_JHofestaedt.pdf.
14. Schneider, J. *Characterization of the mDOM light detection and signal processing system for the IceCube Upgrade* PhD thesis (FAU Erlangen-Nürnberg, 2024). https://ecap.nat.fau.de/wp-content/uploads/2024/08/Dissertation_Judith_Schneider.pdf.

15. IceCube Collaboration. https://icecube.wisc.edu/science/icecube/. (accessed: 09.09.2024).

16. Aartsen, M. G., Abbasi, R., Abdou, Y., Ackermann, M., *et al.* Measurement of South Pole ice transparency with the IceCube LED calibration system. *Nuclear Instruments and Methods in Physics Research A* **711,** 73–89. arXiv: 1301.5361 [astro-ph.IM] (May 2013).

17. Ackermann, M., Ahrens, J., Bai, X., Bartelt, M., *et al.* Optical properties of deep glacial ice at the South Pole. *Journal of Geophysical Research (Atmospheres)* **111,** D13203 (July 2006).

18. Abbasi, R., Ackermann, M., Adams, J., Aggarwal, N., *et al.* In situ estimation of ice crystal properties at the South Pole using LED calibration data from the IceCube Neutrino Observatory. *The Cryosphere* **18,** 75–102 (Jan. 2024).

19. Abbasi, R., Ackermann, M., Adams, J., Aggarwal, N., *et al.* Graph Neural Networks for low-energy event classification & reconstruction in IceCube. *Journal of Instrumentation* **17,** P11003. arXiv: 2209.03042 [hep-ex] (Nov. 2022).

20. Aartsen, M. G., Abbasi, R., Ackermann, M., Adams, J., *et al.* Energy reconstruction methods in the IceCube neutrino telescope. *Journal of Instrumentation* **9,** P03009. arXiv: 1311.4767 [physics.ins-det] (Mar. 2014).

21. Aartsen, M. G., Abbasi, R., Ackermann, M., Adams, J., *et al.* IceCube-Gen2: the window to the extreme Universe. *Journal of Physics G Nuclear Physics* **48,** 060501. arXiv: 2008.04323 [astro-ph.HE] (June 2021).

22. Kajita, T. Discovery of Atmospheric Neutrino Oscillations. *International Journal of Modern Physics A* **31,** 1630047–880 (Sept. 2016).

23. Haack, Christian. personal correspondence. FAU Erlangen-Nürnberg, 2024.

24. Cowen, D. F. & Ice Cube Collaboration. *Tau Neutrinos in IceCube* in *Journal of Physics Conference Series* **60** (IOP, Mar. 2007), 227–230.

25. Aartsen, M. G., Abraham, K., Ackermann, M., Adams, J., *et al.* Characterization of the atmospheric muon flux in IceCube. *Astroparticle Physics* **78,** 1–27. arXiv: 1506.07981 [astro-ph.HE] (May 2016).

26. Aartsen, M. G., Abraham, K., Ackermann, M., Adams, J., *et al.* The Detection of a Type IIn Supernova in Optical Follow-up Observations of IceCube Neutrino Events. *Astrophys. J.* **811,** 52. arXiv: 1506.03115 [astro-ph.HE] (Sept. 2015).

27. Aartsen, M. G., Ackermann, M., Adams, J., Aguilar, J. A., *et al.* The IceCube realtime alert system. *Astroparticle Physics* **92,** 30–41. arXiv: 1612.06028 [astro-ph.HE] (June 2017).

28. IceCube Collaboration, Aartsen, M. G., Ackermann, M., Adams, J., *et al.* Multi-wavelength follow-up of a rare IceCube neutrino multiplet. *Astronomy & Astrophysics* **607,** A115. arXiv: 1702.06131 [astro-ph.HE] (Nov. 2017).

29. Glüsenkamp, T. *Muon Filter Proposal IC86-2012* IceCube-internal document. 2012.

30. Lukas Schulte, M. L.-B. *Request for the Online Cascade Filter and Additional Calculations at Level 2 for the IC86 2013 Cascade Filter Stream* IceCube-internal document. 2012.

31. Voge, M. *2012 TFT Proposal Request for a Level2 Online Muon Filter Revised Version 1* IceCube-internal document. 2012.

32. Keivani, A., Kopper, C. & Wandowsky, N. *TFT proposal for a "Starting Event Filter" for the 2015/16 season* IceCube-internal document. 2015.

33. Mase, K. *Request for the EHE filter (2013) (ver. 4)* IceCube-internal document. 2013.

34. SimWeights contributors. *SimWeights* https://software.icecube.wisc.edu/simweights/main/. (accessed: 26.10.2024).

35. Díaz-Vélez, J. C. *Neutrino and Air Shower Simulations in IceCube* https://events.icecube.wisc.edu/event/123/contributions/6472/attachments/5487/6310/Simulation_in_IceCube_2020.pdf. (accessed: 09.09.2024).

36. Kelleher, J. *Deep Learning* ISBN: 9780262354899. https://books.google.de/books?id=p_fTxQEACAAJ (MIT Press, 2019).

37. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* http://www.deeplearningbook.org (MIT Press, 2016).

38. Sanchez-Lengeling, B., Reif, E., Pearce, A. & Wiltschko, A. B. A Gentle Introduction to Graph Neural Networks. *Distill.* https://distill.pub/2021/gnn-intro/ (2021). (accessed: 30.09.2024).

39. Neutelings, I. https://tikz.net/neural_networks/. (accessed: 30.09.2024).

40. Jaiswal, S. *Multilayer Perceptrons in Machine Learning* https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning. (accessed: 30.09.2024).

41. Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., *et al. Relational inductive biases, deep learning, and graph networks* 2018. arXiv: 1806.01261 [cs.LG]. https://arxiv.org/abs/1806.01261.

42. McBride, M. *Adjacency matrices* https://graphicmaths.com/computer-science/graph-theory/adjacency-matrices/. (accessed: 27.10.2024).

43. Verma, N. *Understanding K-Nearest Neighbors (KNN) Regression in Machine Learning* https://medium.com/@nandiniverma78988/understanding-k-nearest-neighbors-knn-regression-in-machine-learning-c751a7cf516c. (accessed: 08.10.2024).

44. IBM. *What is the k-nearest neighbors (KNN) algorithm?* https://www.ibm.com/topics/knn. (accessed: 08.10.2024).

45. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323,** 533–536 (Oct. 1986).

46. Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)* ISBN: 0387310738 (Springer-Verlag, 2006).

47. Scikit-learn developers. *Stochastic Gradient Descent* https://scikit-learn.org/1.5/modules/sgd.html. (accessed: 27.10.2024).

48. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* 2017. arXiv: 1412.6980 [cs.LG]. https://arxiv.org/abs/1412.6980.

49. Laub, P. *Machine-Learning-based Background Identification for the Radio Neutrino Observatory Greenland* MA thesis (FAU Erlangen-Nürnberg, 2020). https://ecap.nat.fau.de/wp-content/uploads/2024/01/Master_thesis_final_Philipp_Laub.pdf.

50. Pykes, K. *The Difference Between Classification and Regression in Machine Learning* https://towardsdatascience.com/the-difference-between-classification-and-regression-in-machine-learning-4ccdb5b18fd3. (accessed: 08.10.2024).

51. Qu, H. & Gouskos, L. Jet tagging via particle clouds. *Phys. Rev. D* **101,** 056019. arXiv: 1902.08570 [hep-ph] (Mar. 2020).

52. Wohlfahrt, F. *GNNs for low-level filters in IceCube using GraphNeT* Bachelor's Thesis (FAU Erlangen-Nürnberg, 2024). https://ecap.nat.fau.de/wp-content/uploads/2024/08/Bachelorarbeit_Fabian_Wohlfahrt.pdf.

53. Ørsøe, R. & GraphNeT team. *GraphNeT - A Deep Learning Library for Neutrino Telescopes.* 2024. https://graphnet-team.github.io/graphnet/index.html (2024). (accessed: 23.09.2024).

54. Ørsøe, Rasmus. personal correspondence. TUM, 2024.

55. Zhu, J., Yan, Y., Zhao, L., Heimann, M., *et al. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs* 2020. arXiv: 2006.11468 [cs.LG]. https://arxiv.org/abs/2006.11468.

56. GraphNeT team. *DynEdge* https://graphnet-team.github.io/graphnet/api/graphnet.models.gnn.dynedge.html#module-graphnet.models.gnn.dynedge. (accessed: 08.10.2024).

57. Erlangen National High Performance Computing Center (NHR@FAU). *Woody* https://doc.nhr.fau.de/clusters/woody/. (accessed: 10.10.2024).

58. Erlangen National High Performance Computing Center (NHR@FAU). *TinyGPU* https://doc.nhr.fau.de/clusters/tinygpu/. (accessed: 10.10.2024).

59. Particle Data Group. *Atomic and nuclear properties of water (ice) (H2O)* https://pdg.lbl.gov/2015/AtomicNuclearProperties/HTML/water_ice.html. (accessed: 20.10.2024).

60. Hoffmann, M. *Bachelorthesis* https://github.com/marvinhfmn/Bachelorthesis. (accessed: 28.10.2024).

61. Hoffmann, M. *BachelorthesisHoffmannMarvin* https://git.ecap.work/icecube/graphnet/BachelorthesisHoffmannMarvin#s. (accessed: 31.10.2024).

Acknowledgments

I want to voice my gratitude for everyone who supported me during the process of writing this thesis. Thank you to everyone at ECAP, and the ICeCube group for the welcoming atmosphere and, in particular, for the insights into the IceCube collaboration. Special thanks go to the following persons:

- Dr. rer. nat. Christian Haack, for giving me the opportunity to work on this project and being always helpful throughout this thesis, regardless of the issue.
- Oliver Janik, for helping me with conceptual and coding related problems, proof-reading and always being available to answer my questions.
- Judith Schneider, for reliably proof reading the thesis at various stages.
- Prof. Dr. Claudio Kopper, for sharing his expertise in neural networks and physics research.
- Fabian Wohlfahrt, for being open about discussing coding related issues and working on similar projects together.
- My parents, who have been, and continue to be the best support I could hope for.

Eigenständigkeitserklärung

Hiermit versichere ich, Marvin Hoffmann (22811608), die vorgelegte Arbeit selbstständig und ohne unzulässige Hilfe Dritter sowie ohne die Hinzuziehung nicht offengelegter und insbesondere nicht zugelassener Hilfsmittel angefertigt zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen und wurde auch von keiner anderen Prüfungsbehörde bereits als Teil einer Prüfung angenommen.

Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Ort, Datum	Unterschrift