UNIVERSITY OF COPENHAGEN

Angular reconstruction of high energy neutrinos using machine learning

Applying transformers to astrophysical neutrino events from the IceCube observatory

L.H. Voorend

Master Thesis

Department of experimental particle physics

Niels Bohr Institute

Supervisor
T.C. Petersen

First version

Copenhagen, May, 2025

Luc Henricus Voorend
Department of experimental particle physics
Niels Bohr Institute
University of Copenhagen
Jagtvej 155 A, 2200 Copenhagen N
May 2025

This document was typeset using LaTeX

Abstract

The IceCube Neutrino Observatory, a Cherenkov detector instrumenting a cubic kilometer of ice at the South Pole, has discovered astrophysical neutrinos of TeV − PeV energy originating from outside our Galaxy. Barely absorbed and undisturbed by magnetic fields, these high energy neutrinos can give the only unhampered view of cosmic ray accelerators far out in the universe. An accurate angular reconstruction of high energy tracks in IceCube is crucial for neutrino astronomy. Traditionally, this reconstruction has been based on likelihood optimizations. This work explores a machine learning approach using a transformer model developed and trained for the angular reconstruction of muon (anti)neutrinos in the 100 GeV − 100 PeV energy range in IceCube. A novel pulse aggregation method was introduced, reducing the sequence length of events by summarizing pulses on a PMT basis. Improvements in angular resolution were found compared to SplineMPE for starting tracks with a track length of up to 700 m in the detector, as well as for downgoing events across the entire energy spectrum. At PeV energies, the transformer achieved a median opening angle of 0.61° and 0.66° for through-going and starting tracks, respectively. Although the machine learning model did not match traditional methods for through-going tracks, it lays the groundwork for future advances in transformer-based angular reconstruction.

Acknowledgments

I would like to take the opportunity to thank the many people who have supported me in the writing of this master thesis in the past nine months.

First, I want to express my appreciation for the guidance from my supervisor, Troels Petersen. His endless supply of ideas and unconditional enthusiasm played an important role in the success of this work. Moreover, I am sincerely grateful for his scientific support beyond the scope of this thesis, allowing my development as an academic researcher. But most importantly, I am happy that he prioritized the thesis being an enjoyable experience.

Next, I would like to thank Inar Timiryasov and Jean-Loup Tastet for sharing their knowledge and experience of machine learning with me and for providing the technical support for developing the transformer. Furthermore, I would like to thank Jason Koskinen and Markus Ahlers for always allowing me to discuss my results with them and placing these results in the bigger context of other research done in the IceCube collaboration.

Then I would like to express my warmest thanks to Cyan Jo and Janni Nikolaides, with whom I had to pleasure to work together daily for most of the project. Their feedback, ideas, and efforts in executing those ideas have contributed significantly to the results of this thesis. Their admirable work ethic led to a great collaboration, which made me go to the office with joy every day. Of course, I would also like to thank the other master students, Simon Ørgaard, Jack Parkinson, Frederikke Rasmussen, and Jens Kinch, for contributing to this.

Lastly, I would like to acknowledge the support from my friends and family, both here in Denmark and from a distance back in the Netherlands, for encouraging me to do the things I love and for withstanding my endless talking about neutrino physics. I could not have done this without you.

Table of Contents

1 Introduction **1**

Part I Theoretical background **3**

2 Particle Physics **4**
2.1 The Standard Model of particle physics . . . 4
2.2 Neutrinos . . . 6
2.2.1 Flavor and mass eigenstates . . . 7
2.2.2 Antineutrinos . . . 8
2.2.3 Neutrino sources . . . 8
2.2.4 Multimessenger astronomy . . . 11
2.3 High energy neutrino detection . . . 11
2.3.1 Neutrino interactions . . . 11
2.3.2 Cherenkov radiation . . . 14

3 The IceCube neutrino observatory **15**
3.1 The detector architecture . . . 15
3.1.1 The IceCube Coordinate System . . . 16
3.1.2 Digital optical modules . . . 17
3.2 Ice properties . . . 18
3.3 The data . . . 19
3.3.1 Data acquisition . . . 19
3.3.2 Hit cleaning and event selection . . . 21
3.4 Simulations . . . 22
3.4.1 The simulation chain . . . 22
3.4.2 The SnowStorm method . . . 23

4 Machine learning **24**
4.1 A mathematical introduction . . . 24
4.1.1 Supervised learning . . . 24
4.1.2 Neurons and activation functions . . . 25
4.1.3 Neural networks . . . 27
4.1.4 Training a neural network . . . 28
4.2 Transformers . . . 32
4.2.1 Model architecture . . . 32
4.3 IceCube Kaggle competition 2023 . . . 36

5 Traditional reconstruction methods **37**
5.1 Muon track reconstruction . . . 37
5.1.1 Line-fit . . . 38
5.1.2 SplineMPE . . . 38
5.2 Cascade reconstruction . . . 40

Part II Methods and results **41**

6 Methods **42**
6.1 The dataset . . . 42
6.1.1 Data cleaning . . . 42
6.1.2 Event selection . . . 43
6.1.3 PMT-fication . . . 45
6.2 Dataloading . . . 47
6.2.1 Data storage . . . 47
6.2.2 PyTorch dataloader . . . 48
6.2.3 Data pre-processing . . . 50
6.3 Model architecture . . . 50
6.3.1 Input embedding and positional encoding . . . 50
6.3.2 Encoder block . . . 51
6.3.3 Aggregation and linear layer . . . 52
6.4 Training procedure . . . 53
6.4.1 Hardware . . . 53
6.4.2 Model hyperparameters . . . 53
6.4.3 Loss function . . . 54
6.4.4 Optimizer . . . 55
6.4.5 Learning rate scheduler . . . 55
6.4.6 Train / validate / test . . . 56

7 Results and discussion **57**
7.1 Event selection . . . 57
7.1.1 Performance on raw data . . . 58
7.1.2 Binned performance . . . 59
7.1.3 Performance on selected set . . . 60
7.2 Transformer scaling laws . . . 62
7.2.1 Size of the training set . . . 62
7.2.2 Model size . . . 63
7.2.3 Number of heads . . . 65
7.3 Pulse maps vs. PMT-fication . . . 66
7.3.1 Neutrino energy of 10 TeV - 1 PeV . . . 66
7.3.2 Neutrino energy of 100 GeV - 10 TeV . . . 68
7.4 Directional reconstruction . . . 71
7.4.1 Level 3 muon (anti)neutrinos . . . 71
7.4.2 Starting and through-going tracks . . . 74
7.4.3 Track length . . . 75
7.4.4 Up- and downgoing events . . . 78
7.4.5 Systematic checks . . . 79

8 Conclusion **82**
8.1 Summary . . . 82
8.2 Recommendations for future work . . . 83

Appendix **85**
A. Model details . . . 85
B. Statistical uncertainty on the median . . . 90
C. List of file locations . . . 94
D. Additional plots . . . 96

Glossary **97**
Bibliography **116**

1

Introduction

Over the last century, significant advancements in physics have been made through the investigation of elementary particles. With the gradual development of the Standard Model of particle physics, many questions on 'what matter is made of' have been answered. However, as is often the case in physics, each answer has led to the emergence of various new questions. Simultaneous research into the building blocks of nature at both subatomic and astronomical scales has revealed that these two realms are more interconnected than previously anticipated. A detailed understanding of hadronic and leptonic interactions is essential to interpret the high-energy processes occurring in active galactic nuclei, among the most powerful and persistent sources in the universe. The detection of high energy astrophysical particles on Earth uncovers the existence of extremely powerful cosmic particle accelerators, yet very little is known about their nature and location in the universe.

In the quest to better understand the origin of matter and the evolution of the universe, one particle plays a particularly important role: *the neutrino*. These particles are sometimes referred to by physicists as 'ghost particles': they have no charge, they have an extremely small (yet still unknown) mass, and they barely interact, making them exceptionally hard to detect. Nevertheless, they come in enormous numbers in the universe, with about 100 trillion neutrinos passing through our bodies each second. Neutrinos do not participate in electromagnetic or strong interactions but interact only via the weak force and gravity. While γ-rays above 10 TeV are strongly absorbed by the cosmic microwave background, and cosmic rays under 10 EeV are significantly deflected by magnetic fields, neutrinos can traverse the cosmos largely undisturbed [1]. This makes them ideal messenger particles for probing astronomical phenomena.

Multi-messenger astronomy and the search for point sources of high energy neutrinos have been the major motivations for building the *IceCube Neutrino Observatory*, a cubic-kilometer-scale Cherenkov detector embedded in the Antarctic ice. In November 2013, IceCube announced the detection of its first neutrinos originating outside our solar system, including two events with energies in the PeV range [2]. In subsequent years, IceCube reported strong evidence of high energy neutrinos associated with a blazar located 5.7 billion light-years away [3], as well as an active galactic nucleus [4]. Furthermore, the IceCube collaboration identified neutrino emissions from the Galactic plane at a 4.5σ significance level [5], and other research has linked two high energy neutrino events detected by IceCube to tidal disruption events [6, 7].

Neutrino astronomy is a rapidly growing field within astroparticle physics, providing insights into the high energy and non-thermal processes in the universe. The field primarily targets astrophysical muon neutrinos (ν_μ) and anti-neutrinos ($\bar{\nu}_\mu$), as their track-like signature in detectors allows for reconstruction with better angular resolution. Over the years, many developments have been made in the analysis methods used in IceCube. Advanced algorithms help distinguish signal from background, and improved reconstruction techniques yield more precise estimates of the neutrino energy and direction. The angular resolution is particularly critical for neutrino source searches, which are typically dominated by background events. Improvements in directional reconstruction accuracy reduce the number of background events included, thereby directly enhancing the statistical significance of the observations.

Traditionally, the directional reconstruction of neutrino events in IceCube has relied on likelihood optimization techniques. Track-like events at $\sim$ TeV energies can typically be reconstructed with an angular resolution of $\lesssim 1^\circ$ [8], while at $\sim$ PeV energies, the resolution can improve to $\lesssim 0.2^\circ$ [9].

The reconstruction algorithms range from general-purpose, fast first-guess algorithms [10] to more sophisticated but slower full-likelihood reconstructions, often tailored to a specific type of event [9]. Recently, new reconstruction algorithms leveraging the potential of machine learning have made an appearance. Once trained, neural networks can reconstruct neutrino events orders of magnitude faster than traditional methods. Graph neural networks, such as those used in *GraphNeT*, have been proven useful for several tasks in IceCube, including event selection, energy reconstruction, and directional reconstruction [11]. However, current machine learning methods still struggle to outperform the traditional reconstruction methods beyond the GeV energy range [12]. To address this, the IceCube collaboration organized a *Kaggle* competition in 2023, challenging participants to develop machine learning algorithms capable of angular reconstruction of high energy IceCube events [13]. Promising results of the competition put a new machine learning architecture in the spotlight: the *transformer*.

This thesis builds upon the results of the Kaggle competition. By employing a novel approach that aggregates pulses to reduce the maximum event sequence length, it aims to develop a transformer model capable of angular reconstruction of high energy neutrino events in IceCube. This work focuses specifically on the reconstruction of tracks from muon (anti-)neutrinos within the $100\,\mathrm{GeV} - 100\,\mathrm{PeV}$ energy range. In doing so, it also explores fundamental aspects of the transformer behavior on IceCube data, such as the effects of event selection, model and training set size, and input data representation. The angular resolution achieved by the transformer is compared to state-of-the-art likelihood-based reconstruction methods for various types of IceCube neutrino events, including both starting and through-going tracks.

This manuscript is divided into two parts. Part I introduces the theoretical background required for this research. It contains general information and does not include research-specific details. It begins with an overview of the Standard Model of particle physics and the key aspects of high energy neutrino physics, including detection mechanisms and detector signatures. Next, it presents the IceCube neutrino observatory, covering the detector architecture, relevant ice properties, and the data, including Monte Carlo simulations. This is followed by an introduction to machine learning, starting with the mathematical foundations of supervised learning using simple neural networks, including topics such as optimization, convergence, and generalization. The components of the transformer architecture are then explained, followed by a brief discussion of the IceCube Kaggle competition. The first part concludes with a discussion of traditional directional reconstruction algorithms in IceCube, such as *line fit*, *SplineMPE* and *SegmentedSplineReco*. Readers unfamiliar with these topics, or wishing to refresh their knowledge, are encouraged to consult the first part.

Part II presents the methods and results of this thesis. It includes all research-specific details and can, in principle, be read as a stand-alone part. It begins by outlining the methods used in the analyses, including the datasets, data cleaning procedures and event selection criteria. A new method of representing IceCube events - by aggregating pulses at the photomultiplier tube (PMT) level, termed *PMT-fication* - is introduced. Additionally, the importance of dataloading is discussed, covering the storage of data, pre-processing of input data and the development of a PMT-fied dataloader. The architecture of the used transformer is detailed, along with the training procedure. The results explore how factors such as training event selection, model width and depth, training set size, model shape, and data representation influence the transformer's ability to perform angular reconstruction of high energy neutrino events. These results inform the design of a final large-scale model, whose angular resolution is evaluated against IceCube's SplineMPE reconstruction method. The thesis will conclude with recommendations for future work.

Part I

Theoretical background

2

Particle Physics

Matter at the subatomic level consists of tiny chunks. These chunks come in different types, and when replicated in astronomical quantities, they form everything around us. Interestingly, these replicas are indistinguishable copies from one another; if you have seen one electron, you have seen all electrons. No electron is bigger or heavier than another. Particle physics is the study of these chunks and the forces between them. The history of particle physics goes back a long way, from the discovery of the electron (e) in 1897 [14], followed by the discovery of many more particles like the photon (γ) [15, 16], pion (π) [17] and muon (μ) [18] in the first half of the 20^{th} century. After 1950, the first of the modern particle accelerators were built, allowing researchers to smash particles together, producing different particles in the process. Ultimately, the curiosity for these fundamental building blocks of the universe has led to the construction of the largest and highest energy particle accelerator to date, the Large Hadron Collider (LHC) at CERN, and the first cubic-kilometer neutrino detector, IceCube, at the start of this century. Many of the discoveries in particle physics have been awarded a Nobel Prize in Physics. Although there is no official category for particle physics within the prize, at least 30-40 of the 118 awarded prizes are directly related to the field.

In this thesis, the neutrino is the central particle of interest. However, to thoroughly understand the behavior of this particle and its interactions with the IceCube detector, some general knowledge about particle physics is required. This chapter aims to give an introduction to the part of particle physics relevant to high energy neutrino physics. First, a general introduction to the Standard Model of Particle Physics will be presented. This will be followed by an introduction to the neutrino and the possible sources of high energy neutrinos, cosmic rays, and the relevant particle interactions for neutrino detection in ice.

2.1 The Standard Model of particle physics

The Standard Model of particle physics is the theory explaining fundamental particles and fields, together with their dynamics [19]. A graphical representation is given in Figure 2.1. It includes twelve spin$-\frac{1}{2}$ particles, the *fermions*, also known as matter particles. Each of these fermions has a corresponding antiparticle, with the same mass but opposite charge. Next to the fermions, the Standard Model includes four *gauge bosons* of spin-1. These particles act as force carriers, responsible for mediating the fundamental interactions. The last particle described by the Standard Model is the Higgs boson, a *scalar particle* with spin-0. The Higgs boson is not a force mediator, however, it does play an important role in the Standard Model. The Higgs boson couples to mass, and through the *Higgs mechanism* it generates the mass of the gauge bosons [20–22].

The fermions are divided into *quarks* and *leptons* (and antiquarks and antileptons). There are six types, or so-called *flavors*, of quarks [19]. They are categorized into three *generations* of increasing mass. The up and down quarks have the smallest mass. The heavier quarks quickly decay into up and down quarks. Therefore, all commonly observable matter is composed of up and down quarks (and electrons). Due to a phenomenon called *color confinement*, quarks can never be found on their own. They combine to form *composite particles*, the so-called *hadrons*. The hadrons themselves are split into *baryons*, made of an odd number of quarks, and *mesons*, made of an even number of quarks (equal amount of quarks and antiquarks). The most common baryons are the proton

and the neutron, together composing the nucleus of every atom. Mesons include particles like the pion and the kaon. All mesons are unstable, with heavier mesons quickly decaying into lighter mesons, and light mesons decaying into leptons, neutrinos, and photons. Quarks are the only particles that experience all four fundamental forces: the electromagnetic force, gravitation, the strong interaction and the weak interaction.

There are two main classes of leptons: *charged leptons*, constituted of the electron, muon, and tau, and neutral leptons, also known as *the neutrinos* [19]. The six total types of leptons are once again grouped in three generations. Within the charged leptons, the electron has the lowest mass. For this reason, the electron is stable and the most common charged lepton in the universe, whereas the heavier muons and taus quickly decay into electrons and neutrinos. A similar decay behavior is not seen for the neutrinos. Rather, a neutrino with a specific flavor is in a specific quantum superposition of the three *neutrino mass eigenstates*. This allows the neutrino to *oscillate* between the different flavor states [23]. More on this will be discussed in Section 2.2.1. Leptons are not subject to the strong interaction, but experience the three other fundamental forces (where the electromagnetic interaction is proportional to charge, so absent for electrically neutral neutrinos).

The gauge bosons act as force carriers for the elementary fermions [19]. The Standard Model knows four types of gauge bosons:

- *Photons*, a massless particle mediating the electromagnetic force, the force between electrically charged particles.

- *Gluons*, another massless particle, which mediates the strong interactions. This is the force binding quarks together into hadrons. Their theory is described in quantum chromodynamics (QCD), and there exists a total of eight different gluons.

- *Z bosons*, an electrically neutral particle mediating the weak interactions. The Z boson has a rather large mass compared to most other particles in the Standard Model. The weak interaction is the mechanism responsible for radioactive decay. It is called weak, since the field strength is several orders of magnitude less than the electromagnetic and strong forces. Interactions mediated by the Z bosons are called *neutral current* interactions.

- *W bosons*, a particle with either a positive or negative electric charge of 1 mediating the weak force, respectively represented as W^+ and W^-. Also the $W^\pm$ has a rather large mass, though slightly less than the Z boson. Weak interactions mediated by the W boson are called *charged current* interactions.

When interacting, elementary particles do this by the exchange of these gauge bosons, often in the form of *virtual particles*. The theory of virtual particles is far beyond the scope of this thesis, but it is important to know that it allows virtual particles to spontaneously emerge from vacuum at short time and distance scales.

The Standard Model is not a complete theory and leaves some phenomena unexplained. For example, the fourth fundamental interaction, gravity, is not explained [24]. Another limitation of the Standard Model is its inability to explain the matter-antimatter asymmetry [25]. Specifically for neutrinos, the theory of the Standard Model falls short in explaining the flavor oscillations. According to the Standard Model, neutrinos do not oscillate. Yet, the occurrence of this behavior has been observed by various experiments [26]. Although neutrino oscillations are not the main topic of this thesis, they are of great interest for neutrino research in general [27], even awarded with the 2015 Nobel Prize in Physics [28].

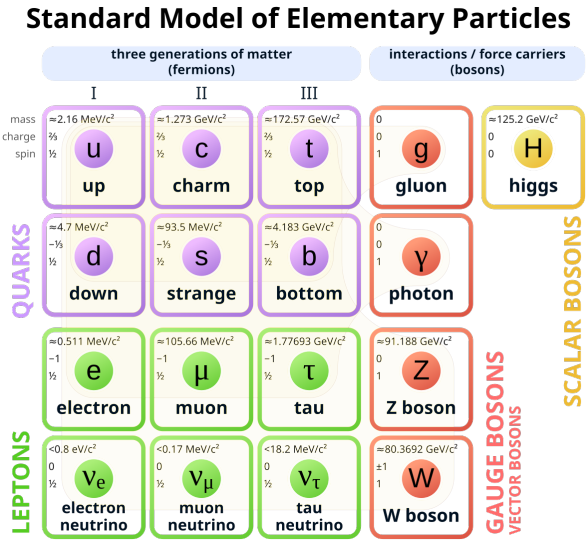


Figure 2.1: Graphical representation of the Standard Model of particle physics. Source: Wikipedia Commons.

2.2 Neutrinos

Since this thesis is about neutrino reconstruction, these particles deserve a slightly more elaborate introduction. Back in the 1930s, a problem emerged when studying the energy of the electron in nuclear beta decay. In this decay process, a neutron in nucleus A transforms into a proton, creating a lighter nucleus B with the emission of an electron, as described in Equation 2.1:

$$A \to B + e^- \tag{2.1}$$

When studying the energies of the outgoing particles in the rest frame of A, particles B and e should come out with equal and opposite momenta, resulting in an expected electron energy depending on the particle masses m by

$$E = \left(\frac{m_A^2 - m_B^2 + m_e^2}{2m_A} \right) c^2 \tag{2.2}$$

according to the law of conservation of energy. The problem was, when measuring the energy of the electron, not a constant energy but rather a continuous spectrum of energies was measured, where the energy of Equation 2.2 formed the upper bound [29]. Wolfgang Pauli suggested a solution to this problem, namely the existence of a third particle emitted during beta decay, a light and neutral particle. A couple of years later, this idea was adopted by Enrico Fermi in *Fermi's theory of beta decay* [30], describing the decay process as a large neutral particle decaying into a large charged particle, an electron and a small neutral particle. Fermi called Pauli's particle the neutrino, Italian for *little neutral one*. The particle that was actually produced was an antineutrino, and nowadays we know this process as

$$n^0 \to p^+ + e^- + \bar{\nu}_e. \tag{2.3}$$

The first experimental detection of the neutrino took a while. In 1956, Clyde Cowan et al. published their confirmation of detecting the neutrino in what is now known as the *Cowan-Reines neutrino experiment* [31]. They reacted the antineutrino from beta decay with a proton, producing a neutron and a positron:

$$p^+ + \bar{\nu}_e \to n^0 + e^+. \tag{2.4}$$

This reaction produced a detectable gamma ray signature, unique to an antineutrino interaction.

2.2.1 Flavor and mass eigenstates

The particle detected by the Cowan-Reines experiment was the electron antineutrino. There exists a total of three neutrino flavors: the electron neutrino (ν_e), the muon neutrino (ν_μ) and the tau neutrino (ν_τ). When neutrinos are absorbed or emitted during a weak interaction, they do so in one of these *flavor eigenstates*. For a long time, neutrinos were believed to be massless particles, but it is now known that there are three distinct neutrino masses. Interestingly, these three neutrino masses do not belong to the three flavor eigenstates. Rather, each flavor eigenstate is a linear combination of three *mass eigenstates*. These mass eigenstates are simply labeled as '1', '2' and '3'. The exact masses of these mass eigenstates are unknown; all that is known is that they are distinct and tiny compared to any other particles. Additionally, it is not even known which of the three is the heaviest. Current observations still allow for two different mass configurations, an unanswered question known as the *neutrino mass hierarchy*, see Figure 2.2. Observations require $\Delta m^2_{21} \equiv m_2^2 - m_1^2$ to be positive, while $\Delta m^2_{31} \equiv m_3^2 - m_1^2$ is still allowed to be positive (normal hierarchy) or negative (inverted hierarchy) [32].

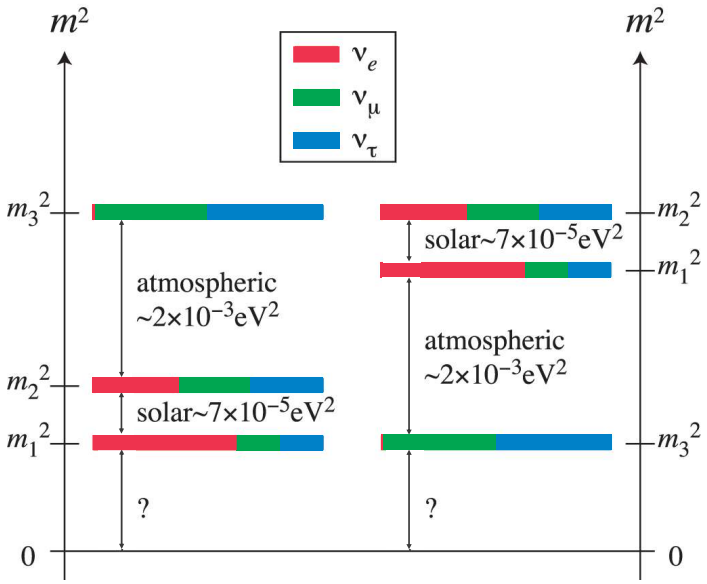


Figure 2.2: Neutrino oscillation measurements still allow for two different orderings of the mass eigenstates. The left shows the normal mass hierarchy, while the right shows the inverted mass hierarchy. Colors indicate the flavor eigenstate composition of each of the mass eigenstates. Source: adjusted from [32]

Curiously, a neutrino created in a specific flavor can later be measured to have a different flavor. While interacting in one of their flavor eigenstates, neutrinos propagate through space as mass eigenstates. More interestingly, because of the mass difference between the mass eigenstates, a relative phase shift is introduced between them while propagating. This results in a changing superposition mixture of mass eigenstates for the neutrino, but since the flavor eigenstates are a mixture of mass eigenstates, this results in a change in flavor state. This phenomenon is called *neutrino oscillation* [23]; e.g. a neutrino created as an electron neutrino will be in some mixture of an electron, muon, and tau neutrino after traveling some distance. This oscillation is periodic, and at some point, the neutrino will return to almost the original mixture, a pure electron neutrino. The oscillation parameters are described in the so-called PMNS matrix, but many of its values are still uncertain [33]. Importantly, neutrino oscillations depend on Δm^2, so the sheer fact that we observe these oscillations proves that the mass eigenstates must have distinct masses.

2.2.2 Antineutrinos

For each of the neutrino flavors, there exists a corresponding antineutrino. These antineutrinos are also spin$-\frac{1}{2}$ particles with no electric charge. Currently, two viable models exist for neutrinos and antineutrinos. The Dirac model claims that neutrinos are *Dirac fermions* (like all other Standard Model elementary fermions), which would mean that the neutrino and antineutrino are different particles. A second theory suggests that neutrinos could be *Majorana fermions*, which would make the neutrino and antineutrino the same particle with two possible chiralities [34]. Extensive experimental efforts are invested into distinguishing the two, for example by studying neutrinoless double-beta decay or by using astrophysical probes like the cosmic microwave background [35, 36], but their details exceed the purpose of this section. Importantly for this research, all antineutrinos observed so far had right-handed *helicity* (projection of the spin state onto the direction of momentum), while all neutrinos were left-handed. This causes a difference in interaction cross section between neutrinos and antineutrinos through a phenomenon called *helicity suppression*, making antineutrinos about a factor 3 less likely to interact [37]. More details on neutrino cross sections are given in Section 2.3.1.

2.2.3 Neutrino sources

Neutrinos are among the most abundant particles in the universe. The sources of the neutrino flux are a combination of natural and artificial sources. Figure 2.3 shows the expected neutrino flux as a function of energy for the dominant sources. A short description of the natural sources:

- **Cosmological neutrinos:** The range of μeV to meV is populated by cosmological (or 'relic') neutrinos. These neutrinos are a relic of the Big Bang, but so far they have not been detected because of their small energies.
- **Solar neutrinos:** The sun is a source of electron (anti)neutrinos in the keV $-$ MeV range, produced during fusion.
- **Supernova neutrinos:** During a supernova, the massive star radiates most of its binding energy in the form of neutrinos [38]. Since the neutrinos are released from the supernova before the burst of photons, detection of these neutrinos can be used as the first indication of a supernova event. Therefore, many neutrino detectors are part of the SuperNova Early Warning System (SNEWS) [39] to provide an indication to astronomers that a supernova may soon be visible.
- **Terrestrial/Geo antineutrinos:** These are electron antineutrinos produced during radioactive beta decay in the Earth, which may help us understand the radioactive nature of the Earth [40].
- **Atmospheric neutrinos:** When cosmic rays interact with the atmosphere of the Earth, they create a shower of neutrinos in the MeV $-$ PeV range. More details on atmospheric neutrinos are given in the following.
- **Astrophysical neutrinos:** The highest energy neutrinos detected come from astrophysical sources like blazars or active galactic nuclei (AGN). A more elaborate description of these neutrinos is given at the end of this section.
- **Cosmogenic neutrinos:** The expected source of the highest energy neutrino flux is interactions of ultra high energy (UHE) cosmic rays with the cosmic microwave background. KM3NeT has reported the potential observation of the first cosmogenic neutrino [41]. However, the current statistical significance and available data are insufficient to confirm this detection with confidence.

Since this thesis considers high energy neutrinos in the GeV − PeV range, a more elaborate description of atmospheric and astrophysical neutrinos is given below.

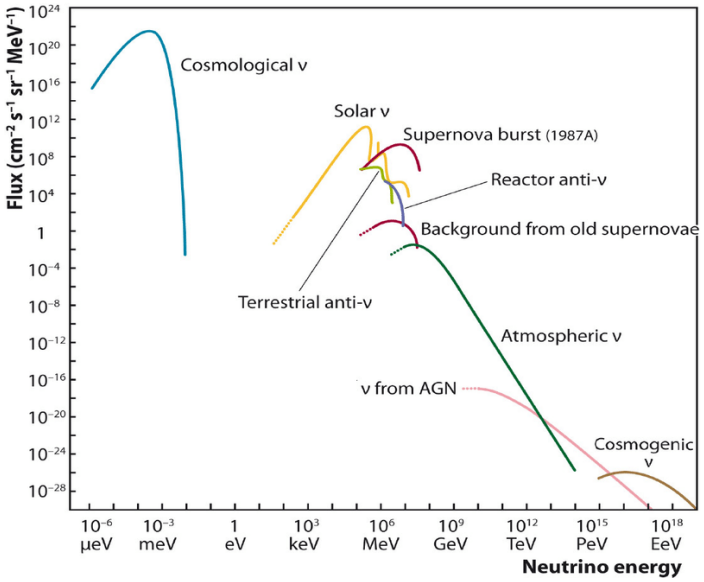


Figure 2.3: Expected neutrino flux as a function of energy for various natural and artificial sources. See text for a more detailed explanation. Source: [38]

Atmospheric neutrinos

The vast majority of the neutrinos detected by IceCube in the GeV range are atmospheric neutrinos. They originate from cosmic rays, highly energetic particles of which about 90% are protons, and the bulk of the rest are alpha particles. The cosmic rays cause *air showers* when interacting with molecules in the atmosphere, as depicted in Figure 2.4. Mostly light mesons, such as pions and kaons, are produced in the interaction. These showers subsequently develop as a combination of three components: the electromagnetic, hadronic and muonic components. The electromagnetic component gets stopped by the surface of the Earth due to energy loss through ionization [42]. The hadronic component forms the core of the air shower with secondary hadronic particles. Neutral pions decay almost instantly into two high energy gamma rays (mostly):

$$\pi^0 \rightarrow \gamma + \gamma. \tag{2.5}$$

The charged pions travel longer, undergoing additional hadronic interactions, producing even more particles. Eventually, they decay into muons and muon neutrinos, feeding the muonic component of the shower:

$$\pi^+ \rightarrow \mu^+ + \nu_\mu, \tag{2.6}$$

$$\pi^- \rightarrow \mu^- + \bar{\nu}_\mu. \tag{2.7}$$

Some of the muons decay further into an electron/positron, an electron (anti)neutrino and a muon (anti)neutrino

$$\mu^+ \rightarrow e^+ + \nu_e + \bar{\nu}_\mu, \tag{2.8}$$

$$\mu^- \rightarrow e^- + \bar{\nu}_e + \nu_\mu, \tag{2.9}$$

while many of the high energy muons do not decay, but penetrate Earth's surface, resulting in a lot of background muon tracks in the IceCube detector. Atmospheric neutrinos are very valuable for neutrino oscillation measurements; however, for neutrino astrophysics, they are considered background.

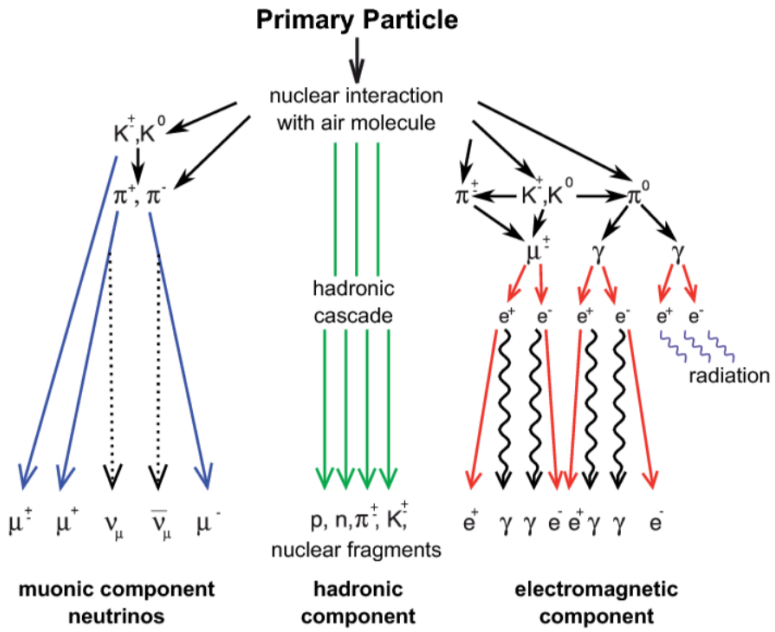


Figure 2.4: Graphical representation of a cosmic ray atmospheric air shower. The shower develops as three parts, an electromagnetic (red), hadronic (green) and muonic component (blue). This muonic component results in the detection of atmospheric neutrinos in IceCube. Source: [42]

Astrophysical neutrinos

The high energy astrophysical neutrinos in the energy range $100\,\text{GeV} - 100\,\text{PeV}$ are the focus of this thesis. Just like atmospheric neutrinos, astrophysical neutrinos originate mainly from the decay of pions and kaons, this time not interacting at the Earth's atmosphere but near galactic or extragalactic accelerators. In this way, astrophysical neutrinos trace cosmic ray accelerators. How these particles are accelerated to such extremely high energies is still unknown, but researching high energy neutrinos might help answer this question. The idea is that near neutron stars or black holes, gravitational energy released during accretion can accelerate cosmic ray particles [1]. Next, these particles interact with ambient gas or radiation, in which charged pions and kaons are produced, which decay to astrophysical neutrinos similar to what is described in Equations 2.6, 2.7, 2.8 and 2.9.

Assume the astrophysical neutrinos are produced in a flavor ratio of $\nu_e : \nu_\mu : \nu_\tau \approx 1 : 2 : 0$. Once produced, these neutrinos travel at the speed of light through space, without participating in further interactions. However, while traveling, they do experience flavor oscillation as described in Section 2.2.1. This averages out their flavor composition upon arrival at the detector, resulting in a detected flavor ratio of $\nu_e : \nu_\mu : \nu_\tau \approx 1 : 1 : 1$ [43].

The first evidence for a high energy astrophysical neutrino flux was found by IceCube in 2013 [44], with 37 neutrino events ranging from 30 TeV to 2 PeV. It is hard to distinguish astrophysical neutrinos from atmospheric neutrinos on an event-by-event basis. However, a spectral analysis enables us to measure the astrophysical flux of $\propto E^{-2.3}$ as an access to the steeper atmospheric spectrum of $\propto E^{-3.7}$ [1]. Moreover, almost no atmospheric neutrinos are detected with an energy above $\sim 300\,\text{TeV}$, so any more energetic neutrino event is likely of astrophysical origin.

2.2.4 Multimessenger astronomy

Many cosmological events emit signals of different 'messenger' types: radio waves, infrared, visible light, X-rays, gravitational waves, cosmic rays or neutrinos. Independent detectors receive these signals on Earth. If it can be identified that all of these signals originate from the same cosmological event, this could allow for a better understanding and reconstruction of the event. This concept is called *multimessenger astronomy*. Neutrinos are the newest addition to the team of messengers. This addition could bring a lot of new, valuable information, since neutrinos are the ideal messenger particles. Having no charge and only interacting weakly, they can reach the Earth from the edge of the universe without adsorption or deflection. They are not affected by magnetic fields like charged particles, and unlike photons, they barely interact with matter. Especially in the energy range from $10\,\mathrm{TeV} - 10\,\mathrm{EeV}$, neutrinos play an important role. High energy photons above 10 TeV are strongly absorbed by interactions with the cosmic microwave background [1]. Moreover, the arrival directions of cosmic rays below 10 EeV are heavily deflected by cosmic magnetic fields. This way, six orders of magnitude in energy are only covered by neutrinos.

In the past decade, various astrophysical sources of high energy neutrinos have been identified by the IceCube detector. In 2018, IceCube announced a high energy neutrino event pointing back to the blazar TXS $0506 + 056$ with a 3σ significance [3]. In 2021 and 2022, two candidate tidal disruption events (TDEs) were identified as a neutrino source [45, 46]. In late 2022, IceCube announced evidence for a high energy neutrino emitted by the active galactic nucleus (AGN) of NGC 1068 (also known as Messier 77). [4].

A good angular reconstruction method is vital for neutrino astronomy. Astrophysical neutrino events are rare and embedded in a background of atmospheric neutrinos as misreconstructed muons from cosmic ray air showers. A high angular resolution narrows the search area, reducing the expected number of background events. The statistical significance of the correlation between neutrino direction and known astrophysical sources scales with the number of signal and background events like

$$Significance \propto \frac{N_{signal}}{\sqrt{N_{background}}}.$$

Therefore, as precise angular reconstructions increase the signal-to-background ratio, they directly enhance the discovery potential of neutrino sources.

2.3 High energy neutrino detection

Their feeble interactions with matter make neutrinos good messenger particles, but unfortunately, also make them very hard to detect. Already in the 1970s, it was known that extremely large detectors would be required to detect astrophysical neutrinos in statistically significant numbers [1]. There exist neutrino detectors - some still being under construction - underwater (like BDUNT and KM3NeT), underground (like Super/Hyper-Kamiokande, JUNO or DUNE) and in-ice (like IceCube). The two most common neutrino detection methods are the use of scintillation or Cherenkov light [47]. This section will limit itself to the physics of neutrino detection in IceCube, an in-ice Cherenkov detector. More details on the IceCube detector are given in Chapter 3.

2.3.1 Neutrino interactions

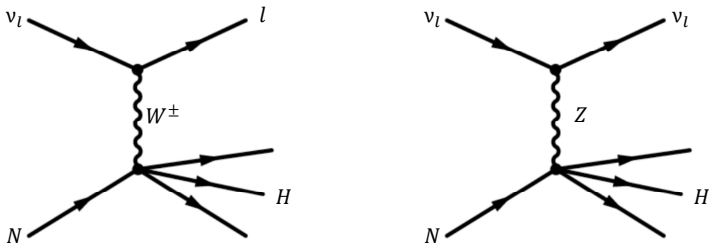
When a neutrino is detected, it is actually its reaction products of an interaction close to or in the detector that are measured. It is not possible to detect a neutrino traveling through the detector directly. The main channels for detecting high energy neutrinos in IceCube are the charged current (CC) and neutral current (NC) interactions through deep inelastic scattering with

an atomic nucleus N in the ice:

$$\mathrm{CC} : \nu_l + \mathrm{N} \rightarrow l + \mathrm{H}, \tag{2.10}$$

$$\mathrm{NC} : \nu_l + \mathrm{N} \rightarrow \nu_l + \mathrm{H}, \tag{2.11}$$

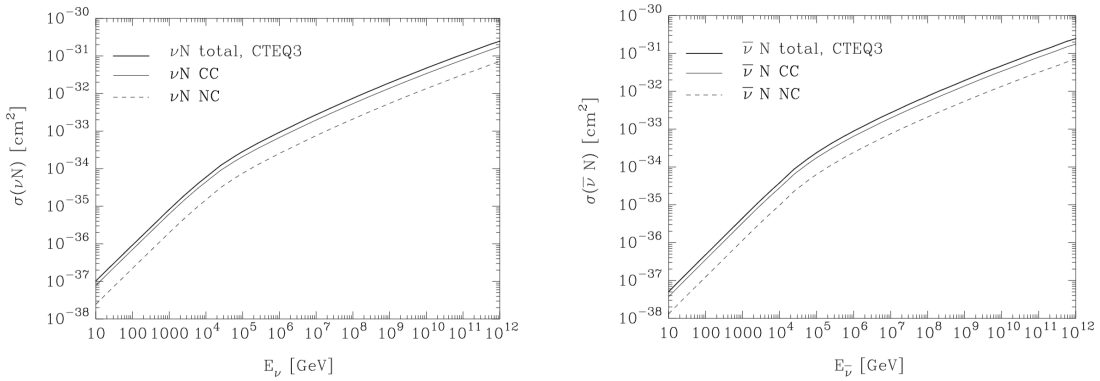
with l the lepton/neutrino flavor and H a hadronic cascade [37]. During the CC interaction, a $W^{\pm}$ boson is exchanged, and the neutrino is converted into its corresponding lepton. The NC interaction takes place under the exchange of a Z boson. Here, the neutrino state is left intact. The struck nucleus does not remain intact and its contents cause a hadronic cascade. Figure 2.6 shows the Feynman diagram of each of the interactions.



(a) Charged Current (CC) **(b)** Neutral Current (NC)

Figure 2.5: Feynman diagrams of the deeply inelastic scattering Charged Current (CC) and Neutral Current (NC) interactions of a neutrino ν with an atomic nucleus N, producing a hadronic shower H.

The likelihood of each of these reactions taking place is called the interaction *cross section* σ. These cross sections depend on energy, and in general, the higher the energy, the more likely the neutrino is to interact. The full spectra for the muon neutrino and muon antineutrino CC and NC deep inelastic scattering interaction cross sections are given in Figures 2.6a and 2.6b respectively. The interaction length scales with $1/\sigma$. As a result of the increasing cross section with energy, the Earth's diameter exceeds the interaction length of neutrinos with an energy above 40 TeV [48]. As a result, the Earth becomes opaque to ultra high energy neutrinos. Moreover, the interaction length of electron antineutrinos is suppressed orders of magnitude around 6.3 PeV due to a resonance reaction of $\bar{\nu}_e + e \rightarrow W^-$, the so-called *Glashow resonance* [49].



(a) Neutrino **(b)** Antineutrino

Figure 2.6: Cross sections for deeply inelastic scattering interactions for high energy neutrinos and antineutrinos as a function of energy: dotted line, NC interaction; thin line, CC interaction; thick line, total cross section. The increase in cross section with energy results in the Earth becoming opaque for neutrinos at ultra high energies. Source: [48]

The charged particles that are created with the interaction emit Cherenkov radiation (see Section 2.3.2) that can be detected by detector modules. Depending on the neutrino flavor and the type of interaction, different detector signatures exist. Figure 2.7 shows an overview of the signatures for the different interaction types and flavors. Hadronic cascades travel a maximum of $\sim 10\,\mathrm{m}$ in ice [50]. Therefore, all NC interactions are almost point-like light sources, resulting in a spherical blob of light. For the CC interactions, the detector signature depends on the neutrino flavor. The electron created in a CC interaction of an electron neutrino loses energy quickly and causes an electromagnetic cascade with a range of the same order as the hadronic cascade [50]. For the ν_τ CC interaction, the detector signature strongly depends on the energy of the neutrino. The created τ lepton quickly decays. The τ sometimes decays to an electron, which then causes an electromagnetic cascade like a CC interaction of ν_e. During its short life, the τ travels about $50\,\mathrm{m} \cdot (\mathrm{E}_{\nu_\tau}/\mathrm{PeV})$ [51]. Below 1 PeV, the traveled distance is so small that the spatial separation of both cascades cannot be observed. At higher energies, the CC interaction vertex and tau lepton decay vertex can be resolved, known as a so-called *double bang* signature. In approximately 17% of the cases, the τ decays to a muon, leaving a track-like signature.

The CC muon neutrino interactions are central in this thesis. The muons survive for a long time, creating elongated tracks of detector hits that can cross the entire IceCube detector, even at energies in the low TeV range [50]. Their long tracks make them ideal for angular reconstruction of the neutrino. The mean interaction angle ψ between the muon neutrino and the muon depends on the neutrino energy E_{ν_μ} and is given by [52]

$$\psi = 0.7^\circ \cdot \left(E_{\nu_\mu}/\mathrm{TeV}\right)^{-0.7}. \tag{2.12}$$

This introduces an error in the reconstructed neutrino angle if just the muon track is used. For muon neutrinos with their interaction vertex in or close to the detector, the hadronic cascade can be used to correct for ψ.

Interaction	Secondary particles	Detector signature
CC ν_μ	μ track and hadronic cascade	Track with cascade
CC ν_τ	τ decays into μ ($\sim 17\,\%$ b.r.)	
	τ decays into e / hadrons	
CC ν_e	Hadronic and EM cascades	Cascade
NC ν_α	Hadronic cascade	

Figure 2.7: Overview of the different neutrino interaction signatures, depending on interaction type and neutrino flavor. The interaction signatures are divided into two main classes: tracks and cascades. Source: [53]

2.3.2 Cherenkov radiation

When particles travel through a medium faster than the phase velocity of light in that medium, they emit visible radiation called *Cherenkov radiation*, named after Pavel Cherenkov, who first described this phenomenon in 1937 [54]. This mechanism is used in IceCube to detect neutrinos. Cherenkov photons from the charged neutrino interaction products are detected, since these particles travel faster in the ice than light does. The photon emission angle depends on the velocity of the charged particles and the refractive index n_p of the medium, described by

$$\cos(\theta_C) = \frac{1}{\beta\, n_p}, \tag{2.13}$$

with $\beta^2 = v^2/c^2$. For charged particles with $\beta \approx 1$ and a refractive index of ice of $n_p = 1.32$, this angle is $\theta_C \approx 41°$ [50]. For track-like particles, this results in a cone of Cherenkov light, while for the cascade events, the Cherenkov front looks more spherical, see Figure 2.8. The detection depends strongly on the propagation of photons through the ice, since the photons can be scattered or absorbed before detection. These properties are discussed in IceCube's ice models, which will be discussed in Chapter 3.

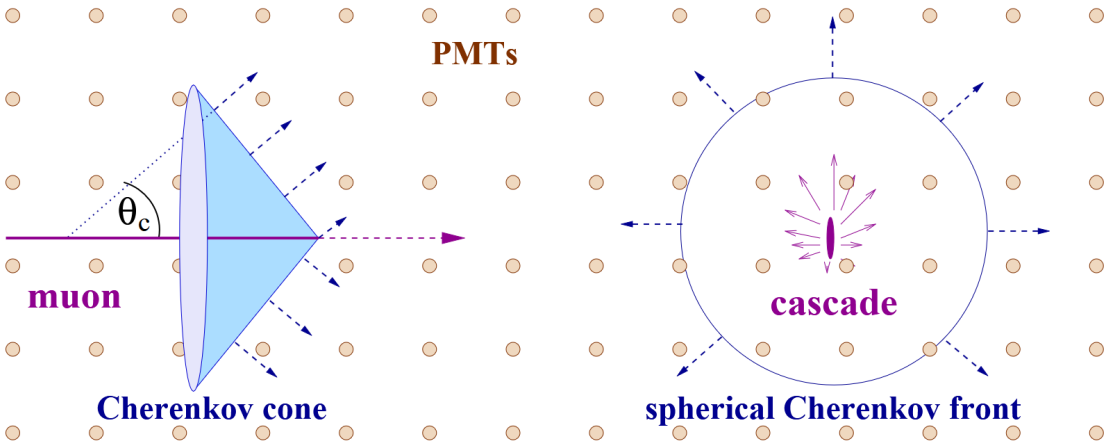


Figure 2.8: Cherenkov light signatures for muon track events (left) and cascade events (right). Source: [55]

3

The IceCube neutrino observatory

The IceCube neutrino observatory (or IceCube for short) is a neutrino detector located at the Amundsen-Scott South Pole Station in Antarctica. It is developed by the University of Wisconsin-Madison and is the successor of the Antarctic Muon and Neutrino Detector Array (AMANDA), which is now a part of the IceCube detector. Already in 1960, Moisey Markov proposed the idea to build a neutrino detector based on Cherenkov radiation in a clear medium like a lake or a sea [56]. This concept led to a series of underwater and in-ice detectors, eventually resulting in the completion of IceCube in 2010, the biggest neutrino detector in the world to date. With its cubic kilometer size, IceCube was primarily designed to observe astrophysical neutrinos in the energy range of about 100 GeV to several PeV. However, next to neutrino astronomy, IceCube's scientific goals include researching neutrino oscillations with DeepCore [57], a search for sterile neutrinos [58], indirect dark matter searches [59], cosmic ray physics [60] and glaciology [61].

This chapter will introduce the IceCube detector architecture, including a detailed description of its digital optical modules (DOMs). It will discuss the important optical properties of the Antarctic ice influencing the data for this research. Furthermore, it will give an in-depth description of the IceCube data. This encompasses the data acquisition, IceCube triggers, background cleaning, filters, and simulation methods.

3.1 The detector architecture

IceCube is deployed into the 3 km thick layer of ice at the South Pole. The main array features 78 holes of a 60 cm diameter in a hexagonal grid spaced 125 m apart, see Figure 3.1. Each hole contains a support cable (*string*) with 60 DOMs (see Section 3.1.2) at depths from $1450 - 2450$ m, spaced equidistantly at a vertical distance of 17 m. In the center of the main array, an additional eight strings are installed, creating a denser detector region to research lower energy neutrinos, called *DeepCore*. The DeepCore strings have a typical horizontal spacing of 55 m, with a vertical DOM spacing of 7 m [62]. The energy threshold of DeepCore is approximately 10 GeV. Besides the in-ice DOMs, IceCube instruments some detectors at the surface of the ice, called *IceTop*. IceTop consists of two tanks with ice, containing two DOMs each, at the surface of 81 of the holes [50]. The IceTop array is used for the detection of cosmic ray air showers, and can be used as a veto for atmospheric neutrinos coming from above [63].

The partial detector configurations during building are named according to their number of strings at that time (IC22, IC40, IC59, IC79 and most recent IC86). Since these partial detector configurations were already taking data, their corresponding datasets are labeled with the same names. Soon, IceCube plans to install 700 new DOMs on 7 strings within the current DeepCore array, a project called *IceCube Upgrade* [64]. Its goal is to bring the detector sensitivity down to a few GeV. Moreover, the *IceCube-Gen2* collaboration hopes to add another 120 strings by 2033, resulting in a total instrumented volume of $\sim 8\,\mathrm{km}^3$ [65]. This would allow us to resolve the high energy neutrino sky from TeV to EeV energies.

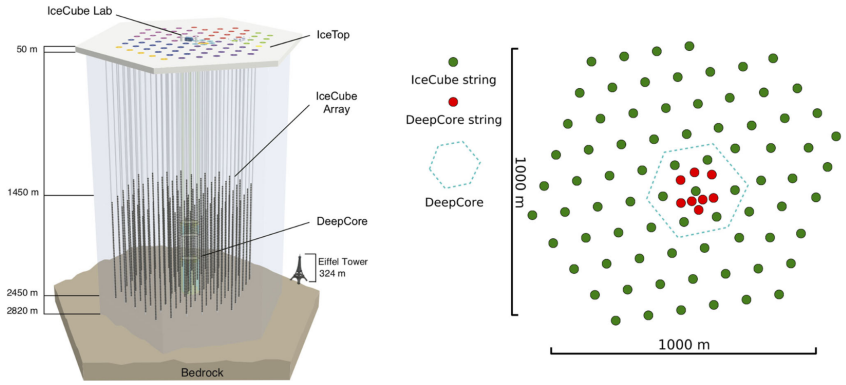


Figure 3.1: Graphical representation of the IceCube detector. The left shows its three-dimensional configuration, while the right shows the surface footprint of the IceCube main array (green) and DeepCore (red) strings. Source: [62]

3.1.1 The IceCube Coordinate System

The IceCube detector geometry and the IceCube data are commonly described in the *IceCube Coordinate System*. Positions (x, y, z) are described in meters, with the positive y-axis pointing in the northern direction (parallel to the prime meridian, pointing towards Greenwich, UK), the positive x-axis in the eastern direction (parallel to the 90^{th} Meridian East, pointing towards Nepal), the positive z-axis pointing towards the ice surface, away from Earth. The origin is at the center of the IceCube detector, about 1500 m below the surface of the ice. The direction of particles traveling through IceCube is described in spherical coordinates using the standard zenith and azimuth angles. Importantly, the direction of particles is defined as the vector from the origin of the coordinate system to the neutrino source, opposite to the direction of the neutrino velocity vector, see Figure 3.2. This means that zenith angle $\Theta = 0°$ described a vertically downgoing particle, traveling from the ice surface towards the core of the Earth, while $\Theta = 180°$ describes a perfectly upgoing particle, traveling from the core of the Earth towards the ice surface. Neutrinos at $\Theta > 90°$ are said to come from the *Northern sky*, while neutrinos at $\Theta < 90°$ come from the *Southern sky*.

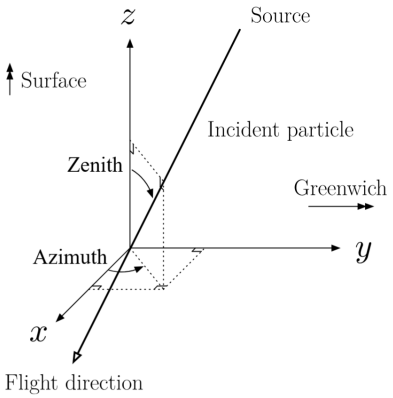


Figure 3.2: Diagram of the IceCube coordinate system, with the origin at the center of the detector, the positive y-axis parallel to the prime meridian, the positive x-direction parallel to the 90^{th} Meridian East and the positive z-direction towards the ice surface. Particle directions are described by spherical coordinates, with a vector pointing towards the particle source. Source: The IceTray documentation

3.1.2 Digital optical modules

The main components of the IceCube detector are its 5160 digital optical modules (DOMs), see Figure 3.3. They are 33 cm diameter glass spheres, enclosing the DOM's photo multiplier tube (PMT) and electronics in 13 mm thick glass. The strings with DOMs are deployed into water columns that have been melted by a hot-water drill, after which the water columns were refrozen. The DOMs must be able to withstand extreme pressure and temperature conditions while no longer being accessible for maintenance after being frozen in. The PMT faces the bottom of the DOM. Here, the single Cherenkov photons are captured by the photocathode and amplified by a factor of 10^7 by ten dynode stages [66]. The PMT used in the IceCube main array DOMs is the Hamamatsu R7081-02 with a diameter of 25 cm. The DeepCore DOMs use a newer version Hamamatsu PMT, the R7081MOD, with an improved efficiency [67]. The photocathode of the PMTs has a spectral response spanning from 300−650 nm [68], with a maximum quantum efficiency of $\sim 25\%$ at 390 nm. The PMT is coupled with the glass by a silicone gel, which cuts off photons of a wavelength below 350 nm [68]. Next to the PMT, the DOM includes a high voltage power supply, electronic calibration systems, light emitting diodes for photonic calibrations and a complete data acquisition (DAQ) system. A diagram of the DAQ system is given in Figure 3.4. An average single photoelectron (pe) generates a pulse of about 10 mV with a width of 5 ns [66]. Recording and digitization of a detected photon is triggered when a charge of at least 0.25 pe is measured. After the trigger, one of two Analog Transient Waveform Digitizers (ATWD) records the waveform for a 420 ns window at a sampling resolution of 3 ns [50]. Simultaneously, a fast analog digital converter (fADC) records a slower and longer signal, covering 6.4 μs at a resolution of 25 ns.

The DOMs have a dark noise rate of ~ 500 Hz, mainly caused by radioactive decays in the PMT and glass shell. Furthermore, occasionally ($< 1\%$) a photon can pass the photocathode and strike the first dynode directly, causing a *pre-pulse*. In $\sim 4\%$ of hits, the first photo electron is reflected on the first dynode, only triggering the dynode with the second impact, causing a *late pulse* at a delay of ~ 60 ns. Lastly, gas molecules inside the PMT can get ionized, drifting to a dynode and causing a second cascade of electrons, causing *afterpulses* at a delay of $300 - 1100$ ns in $\sim 6\%$ of the hits [50]. Attempts are made to remove these noise hits from the event. More about noise cleaning is discussed in Section 3.3.2.

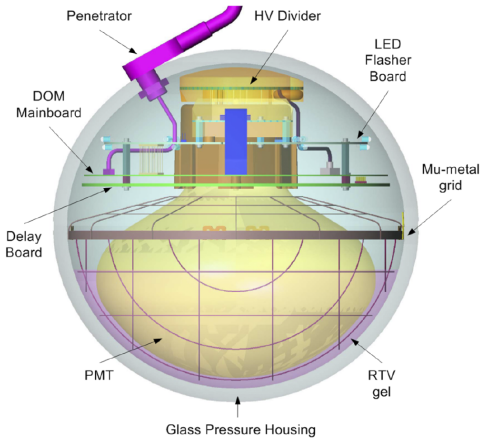


Figure 3.3: Schematic of the IceCube digital optical module (DOM). Source: [50]

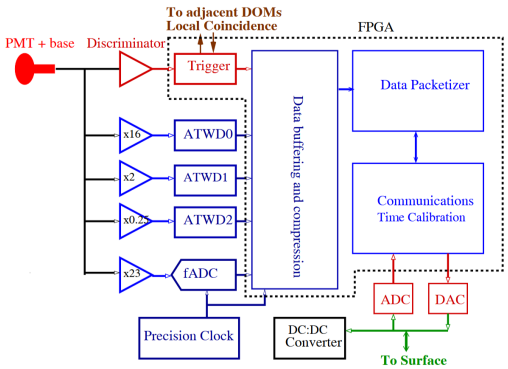


Figure 3.4: Block diagram of the IceCube DOM main board electronics. Source: [66]

3.2 Ice properties

The optical properties of the Antarctic ice are fundamental to IceCube's function as a Cherenkov detector. The glacier in which IceCube is embedded shows a layered structure, where optical properties vary with depth. It consists of compacted snow that can be up to a hundred thousand years old. Its adsorption length varies between $\sim 100 - 300\,\mathrm{m}$, see Figure 3.5. The Antarctic ice is one of the least adsorbent solids to exist, especially at large depths [69]. Since the photon arrival times are an important aspect of the IceCube data, detailed knowledge about the refractive index, scattering and adsorption properties is key. Photons can be absorbed in excitation processes of electrons and molecules, or they can be deflected by trapped volcanic dust, defects in the ice crystals, or air bubbles [70]. Moreover, just below the center of the detector at a depth of $2000 - 2100\,\mathrm{m}$, a dust layer is located [50]. This dust layer strongly affects the optical properties of the detector, with exceptionally high scattering and adsorption in this region. These ice properties are measured by laser dust loggers during drilling and by using LED flashers on the DOMs for calibration [71].

All of these optical properties are summarized in IceCube's ice models. These models have been under constant development since the deployment of IceCube (see [72] for an overview). Through four generations of South Pole Ice (*SPICE*) models, information about scattering of bubbles formed in the refrozen holes of ice [70], tilt of the ice layers [73], ice anisotropies [74], relative DOM efficiencies (RDE) and improvements to the description of Mie scattering [75] have been added. The current recommended ice model is `Spice FTPv3` [76].

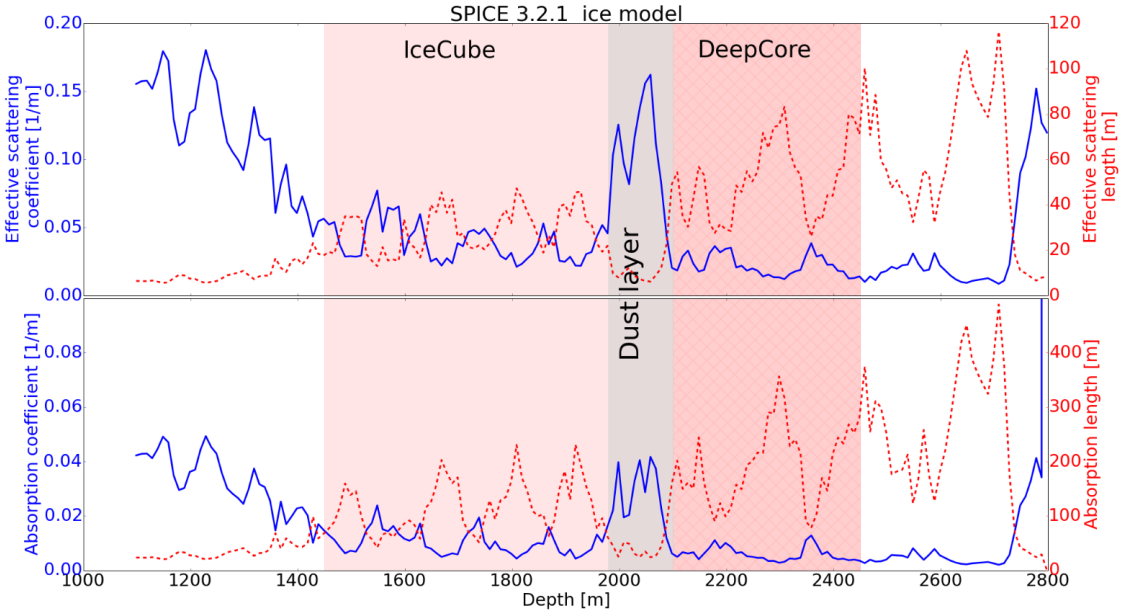


Figure 3.5: Overview of the optical ice properties of the IceCube detector as modeled by the `Spice 3.2.1` ice model. Optical properties vary strongly with depth due to the ice's layered structure. Moreover, a dust layer is located in the middle of the detector, resulting in high scattering and adsorption in that region. Source: [69]

3.3 The data

IceCube detects about one hundred thousand neutrinos with energies above 100 GeV per year [1]. Only a very small part of these are astrophysical neutrinos, at the level of tens of events per year. Moreover, IceCube detects a background of approximately ten billion cosmic ray events per year. Identifying a pure sample of astrophysical neutrinos among all of these events is not an easy task. Since the cosmic ray muons have a short adsorption length in the dense bedrock, they can only enter the IceCube detector from the southern hemisphere. This way, the Earth acts as a cosmic ray filter for upgoing events. As a result, the two major background types from the two hemispheres require separate background rejections [50].

For events from the northern hemisphere, the main task is to remove coincident cosmic ray muons from the southern sky, falsely classified as an upgoing neutrino. Next, the astrophysical neutrinos can be distinguished from the atmospheric neutrinos based on their energy spectrum, as described in Section 2.2.3. Astrophysical neutrinos from the southern sky can be distinguished from the background by the so called *HESE* criterion: A selection of High Energy Starting Events, events with a reconstructed energy of at least 60 TeV and with their interaction vertex within a sub-region of the detector, surrounded by an active veto region [77]. The veto region allows separation of astrophysical neutrinos from the majority of cosmic ray muons, the energy cuts further separate them from the atmospheric neutrinos.

To get a better understanding of the data considered in this thesis, an introduction to the data collection methods, event definition, event cleaning and simulation of events in IceCube is given below.

3.3.1 Data acquisition

The DOMs constantly send digitized, timestamped waveforms of detected photons to the surface, which are collected at the *IceCube Lab*. The hit rate for most DOMs is ~ 500 Hz. The first level of data reduction applied is a check for *local coincidence* (LC). This check is done with in-ice cables, before the waveform is sent to the surface. A DOM hit meets the LC condition if another DOM in a neighborhood of two DOMs in each direction also triggers within 1μs [78]. Hits not meeting the LC condition are still recorded, but their waveforms are reduced to minimal information around the voltage peak. These minimal hit readouts are labeled as soft local coincidence (SLC) hits, while hits meeting the LC condition use more detailed information and are labeled hard local coincidence (HLC). The rate of HLC hits is reduced to $\sim 5 - 25$ Hz per DOM [78].

The waveform from all the DOM hits is combined at the surface. If an *event trigger* is passed, all waveforms are recorded for a certain time window. Various trigger algorithms exist, but the main trigger used is the *Simple Multiplicity Trigger* (SMT) [78]. This trigger looks for N or more HLC hits within a time window of a couple of μs, and extends the window every time this condition is met. The SMT trigger value of N is 8 for the main array and 3 for DeepCore, with respective trigger windows of 5 μs and 2.5 μs. All recorded hits in the trigger window, including those without the HLC flag, are combined into an *event*. If multiple trigger algorithms activate simultaneously, the union of overlapping windows defines the event window [78]. For almost all reconstruction algorithms, working with the waveform data of the event directly is complex and cumbersome [79]. Therefore, the waveforms are unfolded, reconstructing the photon arrival time and charge readout. An example is shown in Figure 3.6. The collection of unfolded hits of the event is called the *pulse series* or *pulse map*. Each hit in the pulse series contains a list of variables, which are described in Table 3.1.

IceCube pulse features

Feature name	Description
`charge`	Reconstructed pulse height in PE (see Figure 3.6)
`dom_time`	Pulse time relative to trigger window start
`dom_x`	x-position of the DOM in IceCube coordinates
`dom_y`	y-position of the DOM in IceCube coordinates
`dom_z`	z-position of the DOM in IceCube coordinates
`pmt_area`	Area of the PMT. Only relevant for upgrade DOMs
`rde`	Relative detection efficiency of the PMTs. Simulated as 1 for IceCube main array DOMs and 1.35 for DeepCore DOMs
`is_bad_dom`	Boolean flag indicating malfunctioning DOMs, information from these DOMs is usually dropped
`is_saturated_dom`	Boolean flag indicating a saturation of charge readout in the DOM, information from these DOMs is usually dropped
`is_bright_dom`	Boolean flag indicating high charge readout in the DOM, information of these DOMs is sometimes dropped for known issues with modeling high charge hits
`event_time`	Global time of the event
`hlc`	Boolean flag for a hard local coincidence hit
`atwd`	Boolean flag indicating if the waveform was recorded with the ATWD. Measure for time resolution
`string`	String number of hit DOM
`pmt_number`	PMT number of hit DOM. Only relevant for upgrade
`dom_number`	Number of hit DOM
`dom_type`	Type of hit DOM. Only relevant for upgrade

Table 3.1: Overview of the pulse features as described in the IceCube pulse maps.

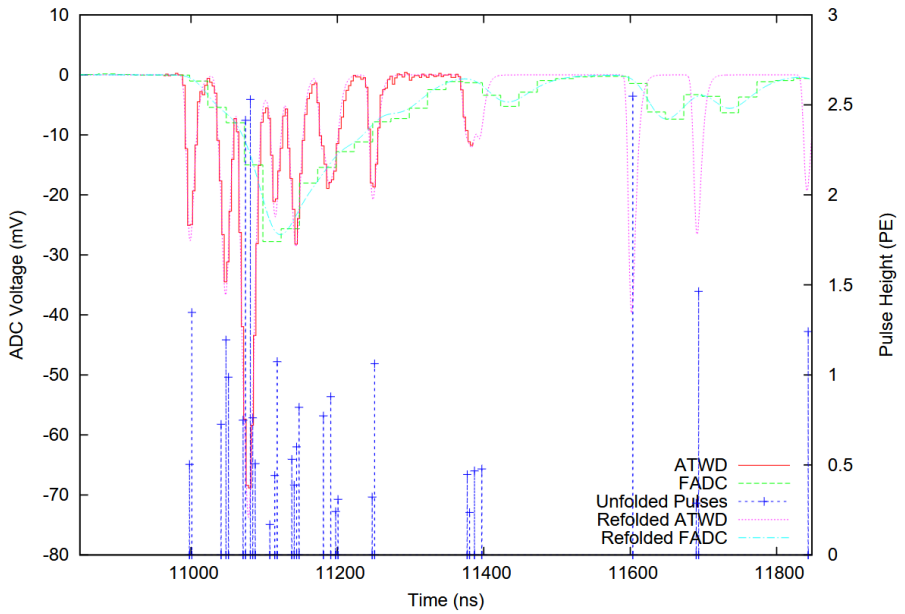


Figure 3.6: Example of waveform unfolding with `Wavedeform`. The blue crosses indicate the reconstructed pulses for this waveform. Pulses are stored in the pulse map for an event. Source: [79]

3.3.2 Hit cleaning and event selection

IceCube's data acquisition (DAQ) produces about 1 TB of data per day, while the satellite bandwidth allocation at the South Pole to send data is 105 GB per day [80]. It is therefore required to reduce the data rate by further processing of hits and cleaning of events. The complete hit cleaning and event selection pipeline is shown in Figure 3.7. The raw events after DAQ are called data at *level* 0 (`L0`). The bulk of the data reduction is done with approximately 25 filters. This is done by the online *Processing and Filtering* (PnF) system. The PnF system is designed to filter triggered events fast, using simple and loose quality cuts [80]. About $\sim 15\%$ of the events pass at least one of the filters [81]. The core filters include:

- `MuonFilter`: A filter searching for high quality track events from all directions. For upgoing events, all energies are accepted, while for downgoing events, only high energy events are selected.
- `CascadeFilter`: A filter searching for events depositing a large amount of energy in or near the detector, used for high energy atmospheric and astrophysical neutrinos.
- `EHEAlertFilter`: A filter targeting the highest energy neutrino events of all types.
- `DeepCoreFilter`: A filter searching for low energy neutrino events contained in DeepCore, mainly used for oscillation research.

Many other filters exist for more specialized searches. In general, the filters in the PnF system have a large overlap for selected events. The 15% of data passing the PnF system is called the `L1` data. The `L1` data is communicated with the *IceCube Messaging System* (I3MS) through satellite to the IceCube servers in Wisconsin [81].

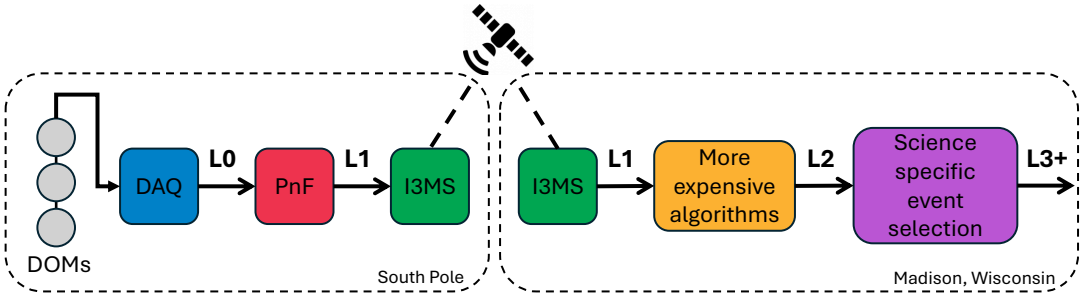


Figure 3.7: Graphical representation of the data acquisition, hit cleaning and event selection pipeline in IceCube. `L2` is the highest level of common data, after which the various groups apply their own science-specific event selection, creating the `L3+` data.

In Wisconsin, more advanced and time-expensive reconstruction algorithms are run, and this information is added to the event information. Moreover, more sophisticated hit cleaning methods can be applied at this stage. The *Seeded Radius Time* (SRT) cleaning is illustrated in Figure 3.8. Its goal is to remove isolated SLC hits from the pulse map. This is done by using the HLC hits of the events as initial seeds, searching for SLC hits within a radius and time window around these seeds, and adding these hits to the seeds. This procedure can be iterated until no new SLC hits are added [82]. At this point, the data is at `L2`, which is the highest level data with common processing. Often, further science-specific event selections are applied to this data, further removing events not contributing to a specific research goal. However, these selections differ per group within IceCube, since the neutrino astronomy group is interested in different events than the oscillation or cosmic ray group, for example. The resulting `L3+` datasets originating from the same `L2` data will therefore contain different events.

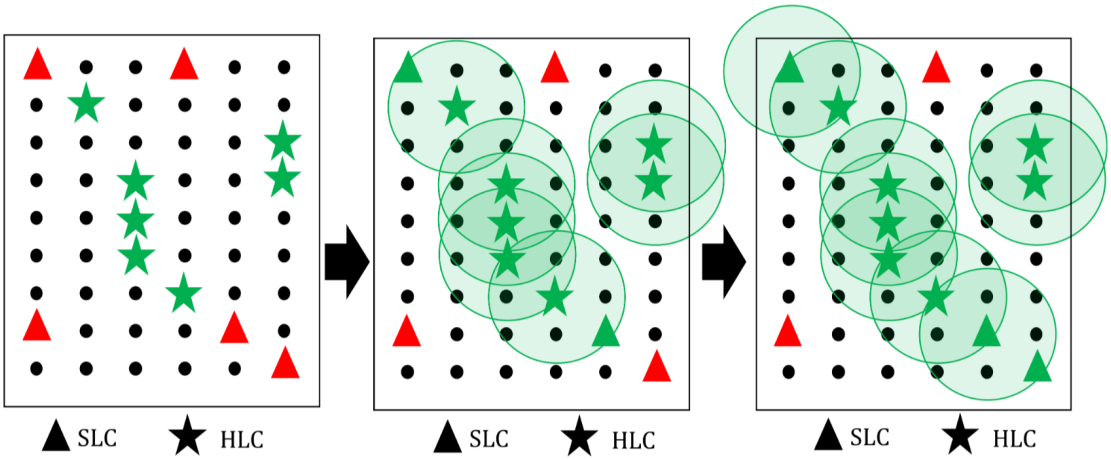


Figure 3.8: Illustration of the SRT hit cleaning method. HLC hits from the original pulse map are used as seeds (S) to search for SLC hits within a certain radius (R) and time window (T). These SLC hits are added to the seeds, and this process can be iterated until no more new hits are added. Source: [82]

3.4 Simulations

The simulation of artificial events is vitally important to the research at IceCube. Comparing simulation data with real data gives valuable information about the physics, hardware and software of the detector. Moreover, simulated data is fundamental to training machine learning models. The simulations allow for the creation of datasets with many millions of events, which are required for training, numbers that would be impossible for real data, especially for rare events like high energy astrophysical neutrinos. This section gives an overview of the simulation chain for generating high energy neutrino Monte Carlo datasets in IceCube. Details on the simulation of other types of events will be left out. Furthermore, it introduces the `SnowStorm` Monte Carlo method, which is used for the simulation of data for this thesis.

3.4.1 The simulation chain

The software framework for simulation, processing and analysis in IceCube is called `IceTray`. The `IceTray` software has a modular structure of independent modules which can be adapted without changing the rest of the chain [50]. The simulation chain attempts to reverse the process of detection and reconstruction as described in Section 3.3: real data start with the waveform output of the DOMs and an attempt is made to reconstruct a particle and its properties (e.g. type, energy, direction etc.), while the simulation starts with the particle and strives to recreate the detector signature it would cause. An overview of the entire simulation chain is given in Figure 3.9.

The chain starts with the creation of a primary particle by the *generator*. Various generators exist: `Corsika` for cosmic ray showers, `MuonGun` for atmospheric muons, `Genie` for low energy neutrinos, and `NuGen` for high energy neutrinos. For `NuGen`, the neutrino properties like energy, flavor and direction are given as input, while the software takes care of the propagation through the Earth, the interaction probabilities (using Monte Carlo), the type of interaction and the generation of the secondary particles after interaction [83]. Next, the propagation of these secondaries is modeled by `PROPOSAL`, which calculates the energy loss of the particles in the ice through, for example, ionization, electron-pair production or bremsstrahlung. This information is then used by the

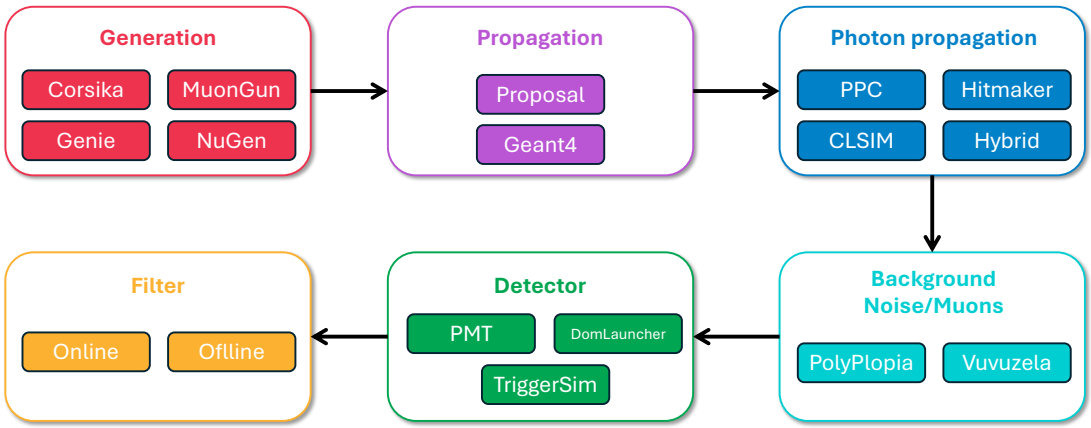


Figure 3.9: Overview of the simulation chain used in IceCube for the production of Monte Carlo datasets. Source: adapted from [83]

photon propagation code (`PPC`), which calculates the propagation of the (Cherenkov) photons inside the detector. Then, coincident atmospheric shower events are merged into the signal event with `PolyPlopia`. Possibly, also thermal and radioactive decay noise is added with `Vuvuzela`. Subsequently, the detector response to these photons is simulated with the `PMTResponseSimulator` for the PMT, `DomLauncher` for the DOM electronics and `TriggerSim` to simulate the triggers. Lastly, the filters are applied.

3.4.2 The SnowStorm method

The details of the physics that are encoded in the modules of the simulation chain described in Section 3.4.1 were purposely left out; each module could be a chapter by itself. A lot of research is invested into investigating the parameters of interaction cross sections of particles [84, 85] (important for generation), the details of energy loss of secondaries [86] (for propagation), the optical properties of the ice [69–71, 76] (for photon propagation) or the calibration of PMT behavior and DOM electronics [87, 88] (to model the detector), just to name a few. However, there still exist many uncertainties on these parameters, something that has to be properly dealt with in simulations.

The `SnowStorm` method is a Monte Carlo technique developed for handling systematic uncertainties on physics parameters during simulations in IceCube [89]. These parameters are referred to as *nuisance parameters*. Specifically, it has been designed to deal with the uncertainties in the dust distributions used in the ice models, an uncertainty notoriously hard to parameterize since the dust concentration varies continuously in depth and has highly non-trivial effects on variables like muon energy or direction. With the `SnowStorm` method, instead of generating multiple simulation sets with specific choices of nuisance parameters, one *ensemble* set is created. Within the ensemble set, each event can have different nuisance parameters, sampled from a parameter phase space. The resulting `SnowStorm` Monte Carlo set has a continuous function of events with respect to the systematic parameters, and by re-weighting the events, each specific choice of nuisance parameters can be recreated from a single set [90]. In principle, this method could be applied to any systematic uncertainty, not just those in the ice models [89]. The `SnowStorm` simulation chain has been slightly adjusted from the regular simulation chain: the neutrinos generated with `NuGen` and the cosmic ray background of `Corsika` are merged with `PolyPlopia` before any particle propagation (including `proposal`) to make sure that signal and background are treated in the exact same way [90].

4

Machine learning

In a matter of about a decade, machine learning has fought its way to the top as a method for achieving striking performance for tasks in a large variety of fields, including natural language processing, computer vision, medicine, and natural sciences. Finding its origin in a 1950s paper on the game of checkers by IBM researcher Arthur Samuel [91], the term *machine learning* has nowadays grown to represent a collection of approaches including supervised, unsupervised, and reinforcement learning. This thesis will confine itself to supervised learning, with an emphasis on *deep learning*, using multi-layered networks to predict a known parameter. For this reason, the terms machine learning and deep learning will be used interchangeably in this chapter.

The crucial role for machine learning in the field of physics was once again emphasized this year with the awarding of the Nobel Prize in Physics to John J. Hopfield and Geoffrey E. Hinton for their "foundational discoveries and inventions that enable machine learning with artificial neural networks". In their motivation, the Nobel committee even highlighted the importance of neural networks in the discovery of the Higgs boson at CERN and the study of astrophysical neutrinos at IceCube [5, 92, 93].

However, it is important to put the capabilities of machine learning into perspective and be aware of its limitations when it comes to giving reliable solutions to complex problems. Creating trustworthy models is exactly the reason why a thorough understanding of the properties of machine learning from a mathematical point of view is a necessity. For that reason, this chapter will start with a rather elaborate mathematical introduction to the basics of supervised learning, before extending this to an introduction to the more complex neural networks central to this thesis, like the transformer.

4.1 A mathematical introduction

The theory of deep neural networks is a very active field of research. Essentially, neural networks are just another family of functions, just like polynomials or trigonometric functions. Combined with gradient-based optimization, they form the main ingredients for the mathematical theory of training neural networks. This section aims to give a mathematical definition of supervised learning with neural networks and answer questions like how to train these models and how to guarantee the generalization of the results.

4.1.1 Supervised learning

In machine learning, the term supervised learning is used for models that consider labeled data. Define X to be the domain of the data, with dimension d. The space X is also known as the *feature space*. Define Y to be the *label space*, with dimension q. The goal of supervised learning is to pair inputs in X with known outputs in Y. The unknown in this task is the mapping between the spaces, which is what we are trying to learn from the available dataset $\mathcal{D}$. This dataset is a subset of $\mathrm{X} \times \mathrm{Y}$. Formally, the task of supervised learning can be defined as:

Definition 1 *Given an unknown function* $f : \mathrm{X} \mapsto \mathrm{Y}$ *between spaces* X *and* Y*, find a good approximation of* f *using the* N *available events in the dataset*

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N} \quad \text{with} \quad y_i = f(x_i) \quad \forall \quad \mathrm{i} \in \{1, 2, ..., \mathrm{N}\}.$$

The elements of Y are called the labels. Note that these labels could be classes, for example $\mathrm{Y} = \{$"neutrino", "noise"$\}$, as well as real numbers. In the first case, the supervised learning task is often referred to as *classification*, while the latter task is often called *regression*.

In general, finding the solution to this task involves three steps:

1. Choose a model $\mathrm{F} : \mathrm{X} \times \mathrm{W} \mapsto \mathrm{Y}$ parameterized by the weight space W, with $w^* \in \mathrm{W}$ such that $\mathrm{F}(x; w^*)$ is as close as possible to $f(x)\ \forall\ x \in \mathrm{X}$.
2. Choose a measure of quality for the model by defining the *loss function* $l : \mathrm{Y} \times \mathrm{Y} \to \mathbb{R}$, where $l(y_1, y_2)$ indicates how different y_1 and y_2 are from each other. This way, $l(F(x; w), f(x))$ will give an indication of how different the model is from the target function.
3. Find w^* by minimizing the loss on the available events in $\mathcal{D}$:

$$w^* = \underset{w \in W}{\arg\min} \sum_{i=1}^{N} l(F(x_i; w), y_i). \tag{4.1}$$

Of course, these three steps do not tell *how* to find w^*. How to choose a model F, how to choose a loss function l, or how to actually minimize Equation 4.1 is still unknown. Unfortunately, the answer to the first two questions in practice often relies on experience and trial-and-error. Still, they are of utmost importance for the quality of the result. There exist many standard models and loss functions that have shown empirical evidence of giving reliable outputs, but it is only an assumption that the same will be true for the mapping between your X and Y of interest. The set of assumptions in machine learning is called the *inductive bias* or *learning bias*. The answer to the last question is a problem of mathematical optimization, where the search for w^* in machine learning is usually called *training* the model. More details on training a neural network will be discussed in Section 4.1.4, but first, the concept of neurons and neural networks will be introduced.

4.1.2 Neurons and activation functions

Machine learning with neural networks considers the class $\mathcal{F}$ of admissible functions $\mathrm{F} : \mathrm{X} \mapsto \mathrm{Y}$ consisting of 'neural network functions'. These functions are, like the name suggests, inspired by biological neurons. Each neuron takes some inputs and produces some outputs. In machine learning, the most common model neuron is a combination of an affine transformation with a non-linear *activation function*. Let $x \in \mathbb{R}^d$ be the input and $y \in \mathbb{R}^q$ be the output of a neuron. Output y is calculated from x by

$$y = \sigma(Ax + b), \tag{4.2}$$

with $A \in \mathbb{R}^{q \times d}$, $b \in \mathbb{R}^q$ and σ the scalar activation function. Here, notation is abused to let the scalar function σ accept matrices and vectors as input, with $\sigma : \mathbb{R} \to \mathbb{R}$ following

$$\sigma\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) = \begin{bmatrix} \sigma(x_1) \\ \vdots \\ \sigma(x_n) \end{bmatrix}$$

Matrix A forms the *linear weights*, while vector b is called the *bias*. These neurons form the elementary units of neural networks.

There exist many choices for the activation function σ. Their non-linearity enables neural networks to deviate from simple linear algebra and solve nontrivial problems, as a single hidden layer network with non-linear activation functions can be proven to be a universal function approximator [94]. This is called the universal approximation theorem of machine learning. The two most common scalar activation functions are the *Rectified Linear Unit* (ReLU) and the *Sigmoid*, defined by

$$\mathrm{ReLU(x)} := \max\{0, \mathrm{x}\} \quad \forall \quad \mathrm{x} \in \mathbb{R}, \tag{4.3}$$

$$\mathrm{sigmoid(x)} := \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}} \quad \forall \quad \mathrm{x} \in \mathbb{R} \tag{4.4}$$

and depicted in Figure 4.1. Both activation functions have their drawbacks, something which will be returned to in Section 4.1.4. There exist adapted versions of the ReLU and sigmoid functions that try to mitigate these effects; however, for the sake of introduction, this chapter will limit itself to these 'traditional functions' (and in practice, their performance often turns out to be sufficient).

Next to the scalar activation functions, another common building block for neural networks is vector-valued nonlinear maps. The three most common are *softmax*, *maxpool* and *layer normalization* defined as

$$\mathrm{softmax} : \mathbb{R}^d \to [0,1]^d, \quad \mathrm{softmax}(\boldsymbol{x})_i := \frac{\mathrm{e}^{\mathrm{x}_i}}{\Sigma_{i=1}^{d} \mathrm{e}^{\mathrm{x}_i}}, \tag{4.5}$$

$$\mathrm{maxpool} : \mathbb{R}^d \to \mathbb{R}^q, \quad \mathrm{maxpool}(\boldsymbol{x}) := \begin{bmatrix} \max_{j \in I_1} x_j \\ \vdots \\ \max_{j \in I_q} x_j \end{bmatrix}, \tag{4.6}$$

$$\mathrm{LayerNorm} : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathrm{LayerNorm}(\boldsymbol{x}) := \frac{\boldsymbol{x} - \mu_{\boldsymbol{x}}}{\sqrt{\mathrm{Var}(\boldsymbol{x})}}, \tag{4.7}$$

where bold now indicates a vector input. In maxpooling, for each $j \in \{1, ..., q\}$ there is an $I_j \subset \{1, ..., d\}$ that specifies the domain of inputs over which the maximum is taken. In the LayerNorm, $\mu_{\boldsymbol{x}}$ indicates the mean of $\boldsymbol{x}$. The maxpooling operation can easily be generalized by replacing the max operation with the min, mean, or sum.

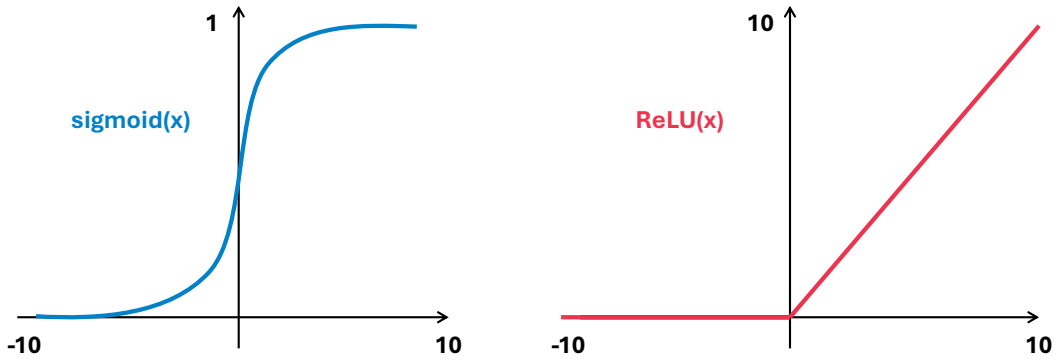


Figure 4.1: Illustration of the ReLu and sigmoid activation functions as described in Equations 4.3 and 4.4. These activation functions introduce nonlinearity into neural networks, allowing them to solve nontrivial problems.

4.1.3 Neural networks

Neurons form the elementary units of a neural network. The standard example of a neural network is the *feed-forward* network, where the outputs of neurons in one layer serve as inputs for the next layer. A graphical representation of a feed-forward network is given in Figure 4.2. Mathematically, a feed-forward neural network can be described in the following way: Let $L \in \mathbb{N}$ and $n_0, n_1, ..., n_{L+1} \in \mathbb{N}$. Let $\sigma_1, ..., \sigma_L$ be activation functions. The layers $T_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ are described by the transformations

$$T_i(x) := A_i x + b_i, \tag{4.8}$$

with $A_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and $b_i \in \mathbb{R}^{n_i}$ for $i \in \{1, ..., L+1\}$. The transformation is parameterized by weight $w = (w_1, w_2, ..., w_L) \in \mathrm{W} = \mathrm{W}_0 \times ... \times \mathrm{W}_\mathrm{L}$ with $w_i = (A_i, b_i) \in W_i = \mathbb{R}^{n_i \times n_{i-1}} \times \mathbb{R}^{n_i}$. The entire network represents the model $\mathrm{F} : \mathrm{X} \times \mathrm{W} \mapsto \mathrm{Y}$ with $\mathrm{X} = \mathbb{R}^{\mathrm{n}_0}$, $\mathrm{Y} = \mathbb{R}^{\mathrm{n}_{\mathrm{L}+1}}$ and

$$\mathrm{F}(\mathrm{x}, \mathrm{w}) := \mathrm{T}_{\mathrm{L}+1} \circ \sigma_\mathrm{L} \circ \mathrm{T}_\mathrm{L} \circ \ldots \circ \sigma_1 \circ \mathrm{T}_1(\mathrm{x}) \quad \forall \quad (\mathrm{x}, \mathrm{w}) \in \mathrm{X} \times \mathrm{W}. \tag{4.9}$$

Here, L is called the *depth* of the network, while n_i is called the *width* of the i^{th} layer. Generally, the complexity of functions the neural network is able to approximate grows faster with depth than with width (see theorem 1.1 and its proof by Telgarsky in [95]). This, however, does not mean that a network with maximal depth will always perform best. In practice, all neural networks benefit from a balance between width and depth.

A feed-forward network is only an example of a deep learning network, and there exist many (more complicated) network architectures. Common deep learning architectures include *recurrent neural networks* (RNNs), *convolutional neural networks* (CNNs), *graph neural networks* (GNNs) and *transformers*. A more detailed description of the transformer architecture will be given in Sections 4.2, but first, an introduction on how to train a neural network for supervised learning tasks is given.

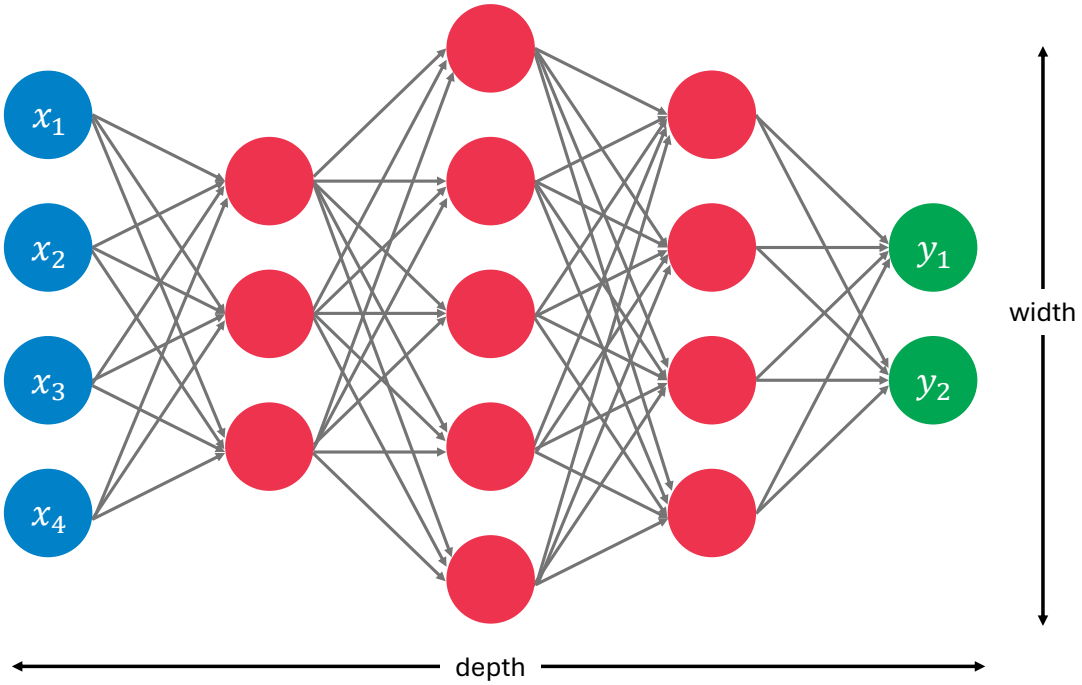


Figure 4.2: A graphical representation of a feed forward neural network with input x, three hidden layers, and output y. The output of each node is given by the activation function applied to an affine transformation of the inputs. The number of layers is called the depth of the network, while the maximum number of nodes in a single layer is called the width.

4.1.4 Training a neural network

With an example of a neural network in our class of functions $\mathcal{F}$ defined, the next step is to try and find the minimum described in Equation 4.1, or in other words, to train the neural network. Assume that the dataset $\mathcal{D}$ was generated by randomly sampling points from a distribution μ on $\mathrm{X} \times \mathrm{Y}$. With this assumption, Definition 1 on the task of supervised learning can be formalized as

Definition 2 *Given a distribution μ on $\mathrm{X} \times \mathrm{Y}$, a loss function $l : \mathrm{Y} \times \mathrm{Y} \to \mathbb{R}$ and a set $\mathcal{F}$ of neural network functions $\mathrm{F} : \mathrm{X} \mapsto \mathrm{Y}$, find the solution of*

$$\min_{\mathrm{F} \in \mathcal{F}} \mathcal{R}(\mathrm{F}), \quad \text{where} \quad \mathcal{R}(\mathrm{F}) := \mathbb{E}_{(\mathrm{x},\mathrm{y})\sim\mu} l(\mathrm{F}(x); y).$$

Here, $\mathcal{R}$ is called the *expected loss*, giving an explicit meaning to 'a good approximation' from Definition 1. Unfortunately, $\mathcal{R}$ is an expectation over μ, which is unknown. The only information on μ is given by the dataset $\mathcal{D}$. Therefore, in practice, the best thing to do is to minimize

$$\mathcal{R}_{\mathcal{S}}(\mathrm{F}) := \frac{1}{|\mathcal{S}|} \sum_{(x,y)\in\mathcal{S}} l(\mathrm{F}(x); y) \tag{4.10}$$

with $\mathcal{S} \subset \mathcal{D}$ a subset of the dataset. One might be tempted to set $\mathcal{S} = \mathcal{D}$; however, for the purpose of generalization of the solution, it turns out to be useful to set aside part of the dataset (more on this in the section Generalization). $\mathcal{R}_{\mathcal{S}}$ is called the *empirical loss* and will be the metric to minimize during training (in a machine learning context, often referred to as the *loss on the training set* or *training loss*).

Stochastic gradient descent

With the neural network model fully parameterized by weights $w \in \mathrm{W}$, Equation 4.10 can be written as

$$\mathcal{R}_{\mathcal{S}}(w) = \frac{1}{|\mathcal{S}|} \sum_{(x,y)\in\mathcal{S}} l(\mathrm{F}(x; w), y). \tag{4.11}$$

Without knowing the exact loss landscape of F, but assuming that everything is differentiable, *gradient descent* can be used to iteratively update w:

$$w_t = w_{t-1} - \eta \nabla_w \mathcal{R}_{\mathcal{S}}(w_{t-1}), \quad \forall \quad t \in \mathbb{N}, \tag{4.12}$$

choosing some starting point $w_0 \in \mathrm{W}$ and *learning rate* $\eta > 0$ (more details about the learning rate will be discussed in the section Convergence). Hopefully, gradient descent converges to the minimum of $\mathcal{R}_{\mathcal{S}}$, a illustrated in Figure 4.3. In practice, using Equation 4.11 is often not feasible, because

- the size of dataset $\mathcal{S}$ can be huge (millions or even billions of events)
- the size of the model can be huge (the dimension of W can also be in the billions).

Taking this into account, calculating $\nabla_w \mathcal{R}_{\mathcal{S}}$ becomes impossible, even with the newest computer hardware. For this reason, the dataset is divided into smaller *batches* of roughly equal size, and the optimization is executed on these batches sequentially. Consider a batch by index set $I \subset \{1, \ldots, N\}$, then the loss on a single batch is defined as

$$\mathcal{R}_I(w) := \frac{1}{|I|} \sum_{i\in I} l(\mathrm{F}(x_i; w), y_i). \tag{4.13}$$

By dividing $\mathcal{S}$ into B batches $I_1, I_2, \ldots, I_B$, the gradient descent of Equation 4.11 can be split up in smaller steps as

$$w_t = w_{t-1} - \eta \nabla_w \mathcal{R}_{I_t}(w_{t-1}). \tag{4.14}$$

At $t = B$, gradient descent has looped over all the batches, which is often referred to as completing an *epoch*. Next, a new set of random batches is sampled from $\mathcal{S}$, and the process is repeated. This process is called *stochastic gradient descent* (SGD), where the word 'stochastic' refers to the random sampling of batches. If the batch I is uniformly drawn from $\mathcal{S}$, then

$$\mathbb{E}[\nabla_w \mathcal{R}_I(w)] = \nabla_w \mathcal{R}_{\mathcal{S}}(w). \tag{4.15}$$

In practice, often slightly different optimization algorithms as an improvement to SGD are used to update the weights. Common methods are to include a momentum-like term to prevent oscillations (the *momentum method*) [96], to use a per-parameter learning rate (*AdaGrad*, adaptive gradient method) [97] or a combination of both (*Adam*, adaptive moment estimation, probably the most popular optimizer at the time of writing) [98, 99]. Although it is important to understand how these optimization algorithms work when using them, a mathematical description of these methods exceeds the introductory purpose of this section. An efficient way of calculating these gradients, combined with its usage in an optimization method to update the model weights, is called *backpropagation* in machine learning.

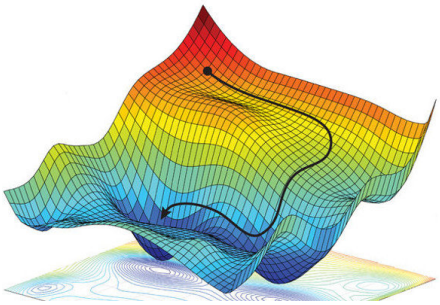
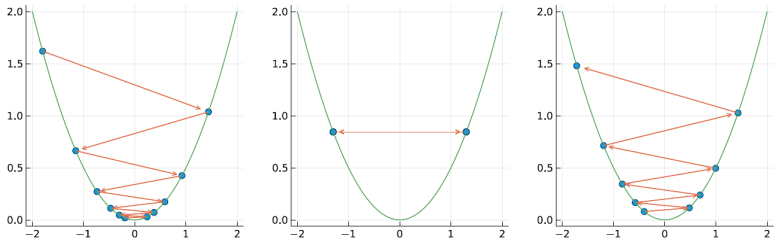


Figure 4.3: An illustrated example of the process of gradient descent on the loss landscape of loss $\mathcal{R}$. The black line indicates the path taken from the starting point w_0 to a (local) minimum by iteratively updating the model weights w as described in Equation 4.11. Source: [100].

Convergence

Stochastic gradient descent and the other optimization algorithms discussed in the previous section are powerful methods to try and minimize $\mathcal{R}_{\mathcal{S}}$; however, they give no guarantee of actually converging to this minimum. Without careful consideration of convergence, neural networks are likely to get stuck in local minima, converge too slowly, or even diverge. Therefore, when training a neural network, there are various things one should take into account.

The most important parameter when it comes to convergence is the learning rate η as introduced in Equation 4.11. For a convex loss $\mathcal{R}$, it can be proven that gradient descent will always converge with a constant η given that η is sufficiently small [101]. However, for a learning rate at or above this limit, the optimization algorithm will get stuck or diverge. An illustrated example of this, for a one-dimensional loss, is given in Figure 4.4. Because of this, one could think it is a good idea to just use a very small learning rate, but this would cause the number of epochs required to reach the minimum to explode. Furthermore, the assumption that $\mathcal{R}$ is convex is false for the vast majority of loss landscapes in machine learning, making convergence even harder. In practice, it turns out to be useful not to use a constant η, but to use a *learning rate scheduler*, reducing the learning rate while training. Popular methods are to reduce η linearly or exponentially with epochs, or to step-wise reduce η upon reaching a loss plateau.



(a) $\eta = 1.8$ **(b)** $\eta = 2$ **(c)** $\eta = 2.2$

Figure 4.4: Illustration of gradient descent on $\mathcal{R}(w) = w^2/2$, with three different learning rates η. The learning rate $\eta = 1.8$ is sufficiently small for the iterations to converge to the minimum. The learning rate $\eta = 2$ is exactly at the threshold and results in iterative bounces back and forth on $\mathcal{R}$. For learning rates above this value, e.g. $\eta = 2.2$, the gradient descent algorithm diverges.

Another important phenomenon to be aware of concerning convergence is *vanishing* or *exploding gradients*. Especially for deep neural networks, a vanishing gradient can cause the model to stop improving during training. To illustrate this, consider a feed-forward network of depth L, disregarding the affine transformations and focusing solely on the activation functions. Let σ be the Sigmoid activation function and define

$$\sigma^L := \underbrace{\sigma \circ \sigma \circ \ldots \circ \sigma}_{L}. \tag{4.16}$$

Applying the chain rule when calculating the derivative of σ^L gives

$$\frac{\partial}{\partial x}\sigma^L(x) = \sigma'\left(\sigma^{L-1}(x)\right)\frac{\partial}{\partial x}\sigma^{L-1}(x). \tag{4.17}$$

Figure 4.5 shows the Sigmoid function together with its derivative. Since $0 < \sigma'(x) < \frac{1}{4}$, the total gradient of the network is

$$\frac{\partial}{\partial x}\sigma^L(x) \leq \left(\frac{1}{4}\right)^L, \tag{4.18}$$

which means that for deep networks, the gradient is very close to zero. This behavior is emphasized by $\sigma'(x)$ going to zero for large absolute values of x. Moreover, considering the floating point precision of computers, a near-zero gradient will actually become an exactly-zero (vanished) gradient. The problem of vanishing gradients is not limited to the Sigmoid activation function. Also, the ReLU activation function has this problem, since its derivative becomes zero for any $x < 0$. If one of the neurons has a gradient of zero, it is called a *dead neuron*, casting a 'gradient shadow' on all preceding layers during backpropagation. One solution to this problem is the use of the *leaky ReLU* [102], a variation on the ReLU with a slightly nonzero derivative for $x < 0$. Of course, many more parameters influence the convergence of the model, like the choice of initial weights w_0, for example, but the learning rate and vanishing gradients are the most common causes of non-convergence.

Generalization

Successfully converging to a minimum of $\mathcal{R}_\mathcal{S}$ has still not solved the supervised learning task as described in Definition 2. The true goal is to create a model F that minimizes $\mathcal{R}$, and not $\mathcal{R}_\mathcal{S}$. It could be that the model has learned features of $\mathcal{S}$ that are not generalizable to μ. In machine learning, this problem is called *overtraining*. In practice, overtraining is prevented by splitting the dataset $\mathcal{D}$ into parts and only training on one of them, while keeping the other parts to test how

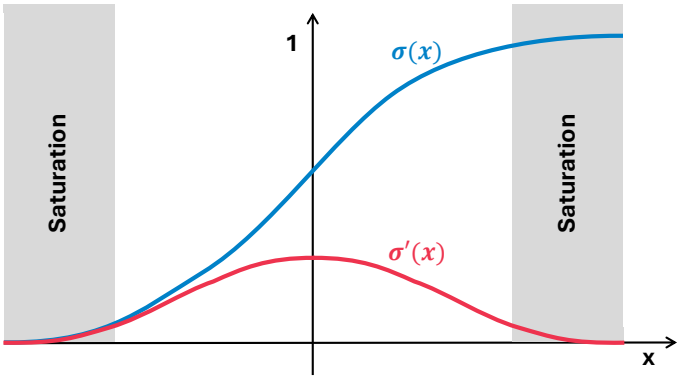


Figure 4.5: Sketch of the sigmoid activation function and its first derivative. For large absolute values of x, the derivative $\sigma(x)$ goes to 0, possibly contributing to the vanishing gradient problem.

well the network performs on unseen data. Often, $\mathcal{D}$ is split in three parts, $\mathcal{D}_{train}$, $\mathcal{D}_{validation}$ and $\mathcal{D}_{test}$. During SGD, the following training loss is minimized:

$$\mathcal{R}_{\mathcal{D}_{train}}(w) := \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} l(\mathrm{F}(x;w),y). \tag{4.19}$$

To determine when to stop training the model, the validation loss is used:

$$\mathcal{R}_{\mathcal{D}_{validation}}(w) := \frac{1}{|\mathcal{D}_{validation}|} \sum_{(x,y)\in\mathcal{D}_{validation}} l(\mathrm{F}(x;w),y). \tag{4.20}$$

The validation loss is often determined after each epoch in training. In general, it is accepted for the training loss to deviate from the validation loss, however, an increasing validation loss is a clear indicator that one should stop training at that epoch. An example is given in Figure 4.6. Practically, one often also likes to experiment with different model *hyperparameters* to improve validation loss when training, for example, changing the learning rate η or the network width or depth. Using the validation set for this purpose could result in overfitting the hyperparameters. Therefore, the final performance of a fully trained and finished model is always judged based on the testing loss of the unseen testing set:

$$\mathcal{R}_{\mathcal{D}_{test}}(w) := \frac{1}{|\mathcal{D}_{test}|} \sum_{(x,y)\in\mathcal{D}_{test}} l(\mathrm{F}(x;w),y). \tag{4.21}$$

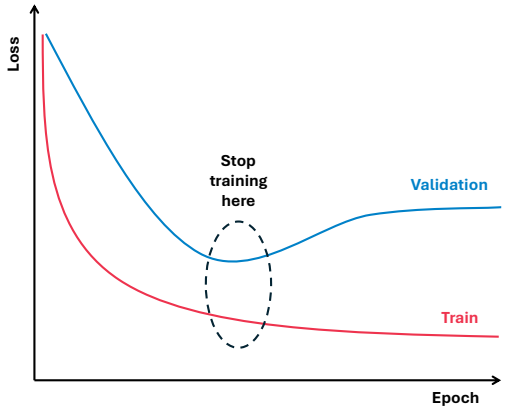


Figure 4.6: Example of the evolution of loss $\mathcal{R}$ as a function of training epoch. In general, it is acceptable for the validation loss to deviate from the training loss. When the validation loss starts increasing, one should stop training.

4.2 Transformers

The machine learning model central to this thesis is the *transformer*. The transformer architecture was first described in the now-famous 2017 paper "Attention is all you need" by Google researchers [103]. Transformers were developed for machine translation, but later found many new applications, most famously the *generative pre-trained transformers* (GPTs) like OpenAI's *ChatGPT*.

Transformers are designed to take a sequence of data as input. Like the title of the paper suggests, transformers are based on the mechanism of *attention* to relate the elements of the sequence to each other. Before the introduction of transformers, most sequence transduction models were based on recurrent or convolutional neural networks (RNNs or CNNs). The most noticeable RNN model was the *long short-term memory* (LSTM) model invented in 1995 [104], which remained the default model for sequence modeling until the introduction of the transformer. Unlike feed-forward neural networks, RNNs process the input elements across multiple time steps. A major drawback of this is their inherently sequential nature of computational steps, which prevents them from being parallelized. This becomes especially pivotal for long input sequences. An attempt to reduce the sequential computations was made by using one-dimensional CNNs [105], however, the downside of these models is that the number of operations to relate faraway sequence elements, also known as the path length, grows with distance. Learning long-range dependencies can be a challenge, but in general, the shorter these paths are, the easier it is to learn these dependencies [106]. The transformer was the first transduction model to entirely rely on the mechanism of attention, relating sequence elements with a constant number of operations independent of their position in the sequence and allowing for significantly more parallelization. These scaling characteristics are summarized in Table 4.1.

Scaling characteristics of various neural network architectures

Layer Type	Complexity per layer	Sequential operations	Maximum path length
Transformer	$\mathcal{O}(n^2 \cdot d)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
RNN	$\mathcal{O}(n \cdot d^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
CNN	$\mathcal{O}(k \cdot n \cdot d^2)$	$\mathcal{O}(1)$	$\mathcal{O}(log_k(n))$

Table 4.1: Per layer complexity, minimum number of sequential operations and the maximum path length between sequence elements for different neural network architectures. Here, n is the length of the input sequence, d the representation dimensionality of its elements and k the kernel size of the convolution. Transformers are parallelizable and have a constant path length between sequence elements. Furthermore, their complexity scales well for sequences of limited length n and high dimensionality d. Source: adapted version of table 1 from [103].

4.2.1 Model architecture

Since transformers were developed for language translation, they are designed to take a (text) sequence as input and output another (text) sequence. For this reason, the original transformer consists of two halves, an *encoder* and a *decoder*. The encoder is designed to extract useful information from the input sequence, while the decoder uses this extracted information and decodes it back into a 'transformed' output sequence. A graphical representation of the original transformer architecture is shown in Figure 4.7. For this thesis, the goal is to use the transformer encoder to extract information from an IceCube event (represented as a sequence), however, the goal is not to decode this information back into another sequence. Rather, the goal is to predict the direction of the neutrino directly from the encoder output. Hence, this section restricts itself to a discussion of the encoder part of the original transformer only.

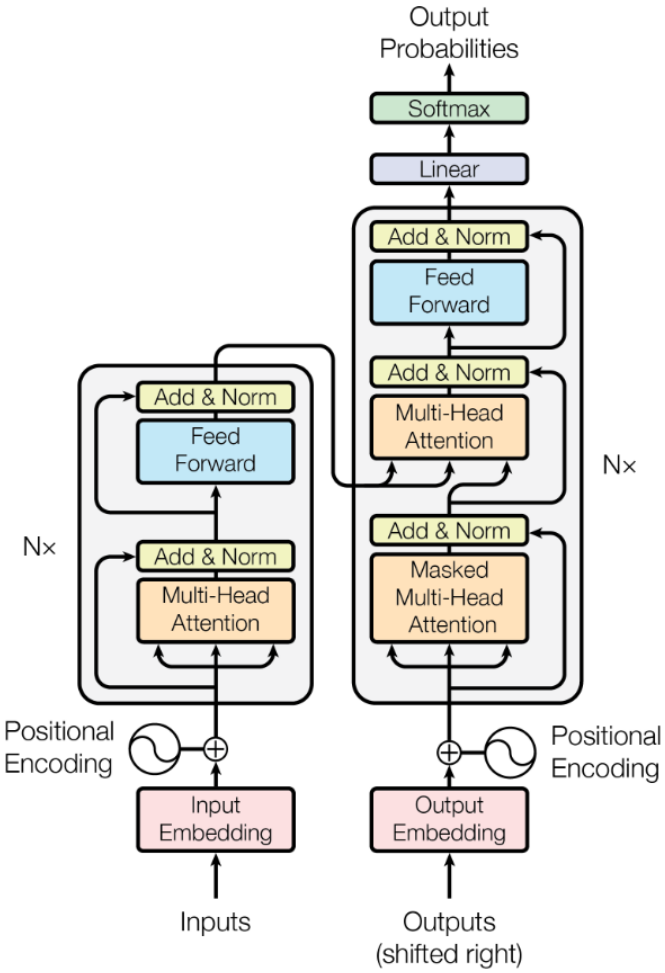


Figure 4.7: Model architecture of the original transformer from the "Attention is all you need" paper. The left half shows the encoder and the right half shows the decoder. Source: [103]

Input embedding

The first step of the transformer is the input embedding. Here, the elements of the input sequence are mapped to a vector of fixed size d_{model}, see Figure 4.8. In the case of language models, these elements are referred to as *tokens*, which often are the words or subwords of a larger sequence of text. In IceCube data, a sequence element is the set of detector features. These input embeddings are a single learnable neural network layer from input dimension to embedding dimension.

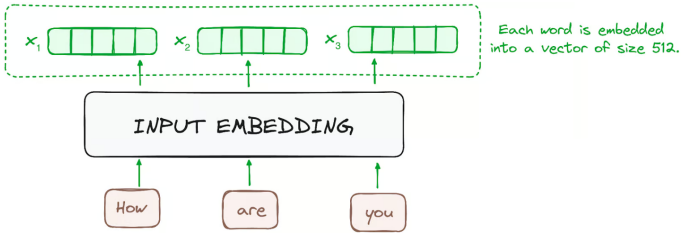


Figure 4.8: Illustration of the transformer input embedding, a learnable mapping for each of the sequence elements to a vector of fixed dimension d_{model}. Source: [107]

Positional encoding

Next to the content of the sequence elements themselves, their position within the sequence is important information for the transformer. This is captured by the positional encoding. The positional encoding maps the absolute or relative positions of the sequence elements to a vector of length d_{model}, the same size as the embedding dimension. There exist many choices for this mapping, both learnable and fixed [103]. Often, a combination of sine and cosine functions of different frequencies is used. The resulting vector of the positional encoding is added to the vector of the input embedding for each of the sequence elements, as shown in Figure 4.9. The sequence of embedded vectors forms the input to the first encoder layer.

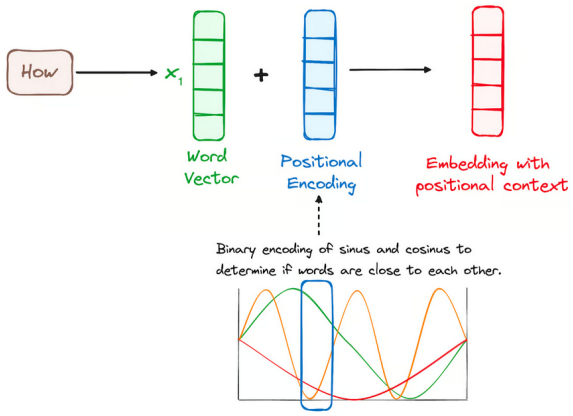


Figure 4.9: Illustration of the transformer positional encoding, a fixed or learnable mapping of the (relative) position of each of the sequence elements to a vector of dimension d_{model}, the same as the input embedding. Positional encoding often uses a combination of sine and cosine functions. Source: [107]

Multi-head attention

The first step of the encoder block is the multi-head attention layer. The multi-head attention consists of multiple parallel single-head attention layers. These single-head attention layers use a mechanism called *scaled dot-product self-attention* (further simply referred to as self-attention). This attention mechanism is the key to how transformers learn, hence the name of the paper "Attention is all you need". The self-attention function maps a query and a set of key-value pairs to an output. First, each of the embedded vectors in the sequence is mapped to a query and key vector of dimension d_k and a value vector of dimension d_v. Next, the dot-product of the query with all other keys is calculated and scaled by dividing by $\sqrt{d_k}$. A softmax is applied to the outcome (see Equation 4.5). The result is called the attention weight, a measure of how good the query matches with one of the keys. The output is determined by a weighted sum of the value vectors using these attention weights.

In practice, these vector dot-products between all queries and all keys are done simultaneously by packing them together in a matrix and using matrix multiplications. With queries packed in a matrix $Q \in \mathbb{R}^{d_{seq} \times d_k}$, keys in matrix $K \in \mathbb{R}^{d_{seq} \times d_k}$ and values in matrix $V \in \mathbb{R}^{d_{seq} \times d_v}$, the output of the self-attention can be described as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{4.22}$$

This way, computation of the self-attention is rather fast. The scaling by $\sqrt{d_k}$ is required for large d_k, since the values of the dot-product can grow large, resulting in vanishing gradients after the softmax [103].

When learning with self-attention, it turns out to be useful to use multiple smaller heads in parallel instead of one single head. In the multi-head attention layer, each element of the input sequence is mapped h times to a query, key, and value with dimensions $d_k = d_v = d_{model}/h$ with an independent learnable layer. Here, h indicates the number of heads. The output of the smaller heads is concatenated and projected back to d_{model}. The goal of multi-head attention is that each head focuses on learning about a different aspect of the input data [103]. Because of the reduced size of each head, the total computational cost is similar to using one single-head-attention.

Single-head and multi-head attention are described in Figure 4.10. After the multi-head attention layer, the original input to the layer is added to the output. This is called a *skip connection*, which helps mitigate the effects of a vanishing gradient. After the skip connection, a layer normalization is applied as described in Equation 4.7.

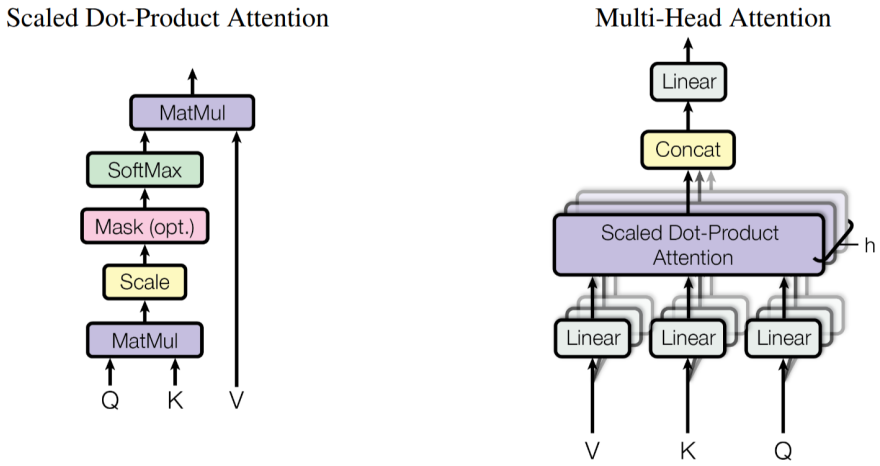


Figure 4.10: Left: Scaled Dot-Product Attention or single-head attention. Right: Multi-head attention, several smaller attention heads in parallel. Source: [103]

Feed forward network

After the layer normalization, a position-wise feed-forward network is applied to the sequence elements. The same feed-forward network is used for all positions. A very simple two-layer neural network is used in the original transformer, mapping the input with dimension d_{model} to a hidden layer with dimension $4d_{model}$ and back to the output layer with dimension d_{model}. After the hidden layer, a ReLU activation function (see Equation 4.3) is used. Similarly to after the multi-head attention layer, a skip connection is used, adding the input before the feed-forward network to its output, and once more, a layer normalization is applied.

Encoder blocks

The multi-head attention layer and feed-forward layer together form a single *encoder block*, indicated by the gray box at the left of Figure 4.7. The full transformer encoder repeats these encoder blocks N times, with the output of one block being passed on as input to the next block. Although the architecture of all encoder blocks is the same, they are independent parts of the model, meaning they all have their own trainable parameters. The original transformer used $N = 6$ encoder blocks and model dimension $d_{model} = 512$. Similar to regular feed-forward neural networks, transformers tend to perform better at increasing model parameter size, once again with a balance in the depth N and width d_{model} [103]. Multi-head attention outperforms single-head attention, but too many heads result in a quality drop-off. Of course, bigger models come with larger computational costs.

4.3 IceCube Kaggle competition 2023

In 2023, the IceCube collaboration hosted the Kaggle competition *IceCube - Neutrinos in Deep Ice*. Kaggle is a competition platform for data science and machine learning challenges, owned by Google. The competition data is provided by Kaggle, and participants submit their code solutions for the challenge to the site, often competing for a prize pool. A condition for receiving the prize money is that the models must be made public, free to use worldwide. The IceCube Kaggle competition had as its goal to create a machine learning model for the angular reconstruction of high energy neutrinos. The competition data consisted of 138 million simulated neutrino events of all flavors with energies between 100 GeV and 100 PeV [13].

A total of 11 206 solutions were submitted to the competition. Strikingly, each of the top 3 solutions (and many other well-performing solutions) was based on the transformer architecture [12]. They showed that IceCube data is very suitable for transformers, outperforming other machine learning architectures in the competition like boosted decision trees, convolutional neural networks or the baseline solution, the graph neural network called *DynEdge* [11] by the open source neutrino telescope deep learning library *GraphNeT* [108]. These top 3 solutions served as a major inspiration for the work done in this thesis. The models were developed to reconstruct both track and cascade events. The distribution of their opening angle (angle between true and reconstructed neutrino direction), including a kernel density estimator (KDE) to estimate the primary mode, is shown for events in the $1 - 10$ PeV range in Figures 4.11 and 4.12 for tracks and cascades respectively.

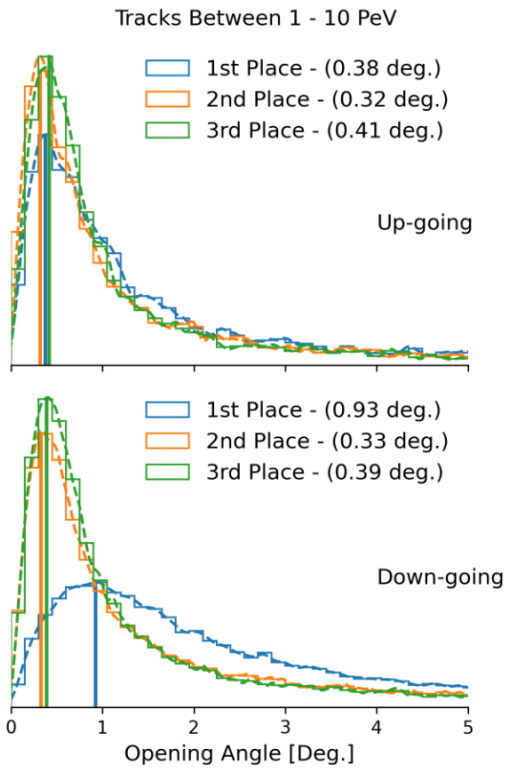


Figure 4.11: Distribution of the opening angle for the top 3 solutions of the IceCube Kaggle competition for $1 - 10$ PeV tracks. Performance is expressed as the primary mode, indicated with vertical lines. Source: [12]

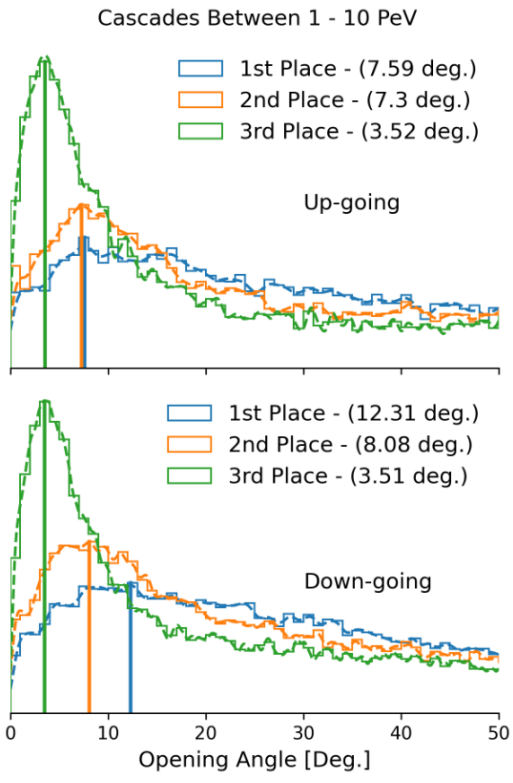


Figure 4.12: Distribution of the opening angle for the top 3 solutions of the IceCube Kaggle competition for $1 - 10$ PeV cascades. Performance is expressed as the primary mode, indicated with vertical lines. Source: [12]

5

Traditional reconstruction methods

Currently, IceCube does not use machine learning as its main method to reconstruct the direction of the detected high energy neutrinos (although recent efforts have been made to introduce the use of CNNs [109], normalizing flows [110] or even hybrid models [111].) Traditionally, the reconstruction of neutrino events in IceCube relies on maximum-likelihood-based methods. This chapter will describe these likelihood methods, which will serve as a baseline for comparison for the machine learning methods developed in this work. The benefits and limitations of the use of each of these likelihood methods are discussed. Since the focus of this work is on the reconstruction of muon tracks from ν_μ charged current interactions, the bulk of this chapter will be on track reconstruction methods. For completeness, a short description of traditional cascade reconstruction algorithms is given at the end of the chapter.

5.1 Muon track reconstruction

The angular reconstruction of muon tracks in IceCube is based on the detection of Cherenkov photons in space and time. The more advanced algorithms incorporate information about the emission and propagation of the photons, while the algorithms applied at the South Pole do not, but rather aim to give a fast first guess. These guesses can later be used as a starting point for the more involved algorithms [50]. A definition of the coordinates used in all reconstruction algorithms is shown in Figure 5.1. The muon track is defined by an interaction vertex at position $\vec{r_0}$ and time t_0. The direction of the particle is given by $\vec{p}$, while DOMs at positions $\vec{r_i}$ and perpendicular distance d to the track can be hit by a Cherenkov cone of light emitted at the Cherenkov angle θ_c. An unscattered photon would arrive at the DOM at the time t_{geo}.

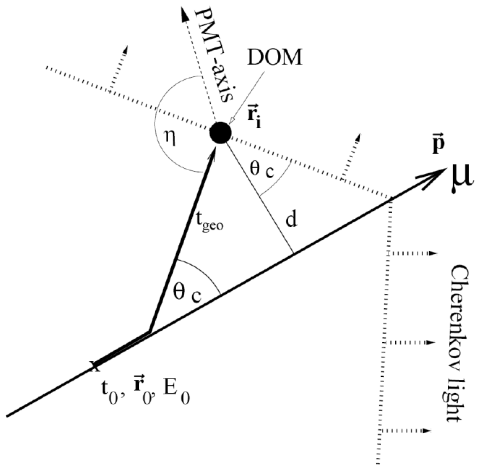


Figure 5.1: Overview of the coordinates used for traditional muon track reconstruction algorithms. Source: adapted from [112]

5.1.1 Line-fit

The most basic algorithm used for angular reconstruction is called `line-fit`. It is a fast algorithm that does not use a seed. Moreover, it does not take into account any optical ice properties or the Cherenkov cone; rather, it approximates the muon light by a plane wave [50]. After the removal of some very obvious outlier hits [10], it fits a line to all remaining hits, using the least-squares method. With the plane wave approximation, all DOM positions are projected onto the 1D muon track. For a muon with track vertex $\vec{r_0}$ and velocity $\vec{v}$, the position $\vec{r_i}$ of a DOM measuring a photon at t_1 can be approximated with

$$\vec{r_i} \approx \vec{r_0} + \vec{v} \cdot t_1. \tag{5.1}$$

The most likely track can then be found by minimizing

$$\chi^2 = \sum_{i=1}^{N_{hit}} (\vec{r_i} - \vec{r_0} - \vec{v} \cdot t_i)^2, \tag{5.2}$$

with N_{hit} the total number of hits after outlier removal. An updated version of `line-fit` made a slight adjustment to Equation 5.2, substituting it for a Huber fit, putting quadratic weights to hits close to the track and linear weights to hits far away [10]. This makes the model more robust to noise. The median opening angle of `line-fit` between the reconstructed angle and true direction of the track is $\sim 5°$. The `line-fit` algorithm is predominantly used to generate an initial track as a *seed* for more sophisticated reconstruction methods.

5.1.2 SplineMPE

More advanced reconstruction algorithms use the physics of neutrino detection by incorporating the parameters of light emission, absorption and scattering. The most likely muon track is determined using the maximum likelihood method. When the muon track is parameterized by a parameter $\vec{\theta}$, this parameter can be estimated from a set of statistically independent measurements $\vec{x}$ by maximizing the likelihood $\mathcal{L}$, given by

$$\mathcal{L}\left(\vec{\theta} \,|\, \vec{x}\right) = \prod_i p(x_i \,|\, \vec{\theta}), \tag{5.3}$$

with $p(x_i \,|\, \vec{\theta})$ the probability to measure x_i given $\vec{\theta}$. For a track in IceCube, $\vec{\theta}$ has five degrees of freedom, namely the interaction vertex (x, y, z) and its zenith and azimuthal direction. Moreover, it is useful to define the photon PDF not based on the absolute timestamp t_{hit}, but on its delay time caused by scattering, defined as

$$t_{res} = t_{hit} - t_{geo}, \tag{5.4}$$

with t_{geo} the unscattered time introduced in Figure 5.1, which is defined as

$$t_{geo} = t_0 + \frac{|\vec{p} \cdot (\vec{r_i} - \vec{r_0})| + d \cdot \tan\theta_c}{c_{vac}}. \tag{5.5}$$

Here, c_{vac} is the speed of light in vacuum [112]. The PDF can now be written as a function of the residual time given a track, $p(t_{res} \,|\, \vec{\theta})$. For all hits in an event, the first arrival photons per DOM can be considered to be the least scattered and, therefore, give the most information about the track. To computationally simplify things, only the first *single-photo-electron* (SPE) per DOM could be considered for the likelihood:

$$\mathcal{L}_{SPE}(\vec{\theta} \,|\, \vec{x}) = \prod_i^{1^{st}\, hits} p(t_{res,i} \,|\, \vec{\theta}). \tag{5.6}$$

However, the first photon in a DOM with just one detected photon is way more likely to still be scattered than the first photon in a DOM with many detected photons. Therefore, in the *multi-photon-electron* likelihood, the DOMs are weighted by their total number of detected photons n_i by

$$\mathcal{L}_{MPE}(\vec{\theta}\,|\,\vec{x}) = \prod_{i}^{1^{st}\,hits} n_i \cdot p(t_{res,i}\,|\,\vec{\theta}) \cdot (1 - P(t_{res,i}\,|\,\vec{\theta}))^{n_i - 1}, \tag{5.7}$$

$$P(t_{res}\,|\,\vec{\theta}) = \int_{-\infty}^{t_{res}} p(t\,|\,\vec{\theta})dt, \tag{5.8}$$

where $P(t_{res}\,|\,\vec{\theta})$ is the cumulative density function (CDF) of the residual time [50].

The key to a good reconstruction with the likelihood method is a good description of the probability density function of t_{res} for the photon arrival time. Ideally $p(t_{res}\,|\,\vec{\theta})$ would be a Dirac delta function. However, in practice, this peak is broadened by various effects, as shown in Figure 5.2. The PMT jitter limits the resolution of t_{res} to some σ_t [112]. PMT dark noise and radioactive decay noise lead to extra hits randomly in time. The stochastic energy losses along the muon track result in little photon showers traveling behind the Cherenkov cone front, resulting in a tail at higher residual times [9]. This is further amplified by the scattering of photons on ice impurities, especially for DOMs that are far away from the track. The effect of scattering is significantly less for water detectors [113].

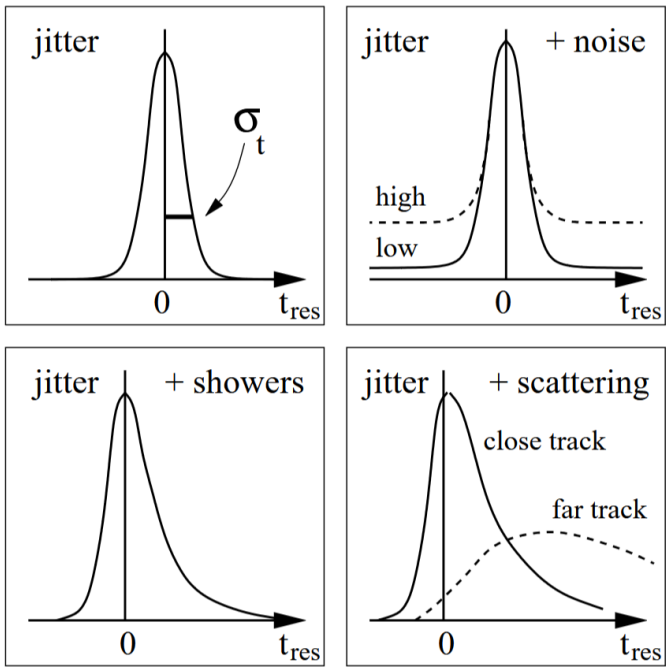


Figure 5.2: Examples of the probability density distribution of photon delay time t_{res} and the effects of PMT jitter, noise, stochastic showers and scattering. Source: [112]

Throughout the years, IceCube has used a variety of descriptions of these probability density functions. First, the analytical Pandel function was used, a normalized parameterization of the Gamma function [112]. This model, called `PandelMPE`, implicitly contains the information about the Cherenkov emission and scattering properties of ice, making it significantly more accurate than `line-fit` [9]. `PandelMPE` was later replaced by IceCube's most used reconstruction model, `SplineMPE`. This model uses basis splines (continuous connected piecewise polynomial functions, also called *B-splines*) to approximate $p(t_{res}\,|\,\vec{\theta})$ [50]. This was done by simulating muon tracks for

many different configurations, storing the detected photon information in high-dimensional, fine-binned histograms. However, with roughly 2 billion parameters, it is computationally infeasible to use this histogram for each event. Therefore, this histogram was fit with multi-dimensional splines and later convolved with a Gaussian function to account for detector timing uncertainties [50]. These splines now represent the PDF and CDF for different track configurations in Equation 5.7.

`SplineMPE` significantly outperforms `PandelMPE`, especially at higher energies [50]. It models the optical properties of the ice much better. However, it still considers the muons as minimally ionizing particles and neglects stochastic energy losses [9]. To mitigate the effects of this simplification, various methods have been implemented to improve the model. This includes hit cleaning of late pulses as a result of stochastic processes with a Kolmogorov-Smirnov test, non-uniform modeling of the PMT noise hits as a function of time, and convolving the entire likelihood of Equation 5.7 with a Gaussian in addition to the individual PDFs [50]. These adjustments further improve the reconstruction with $\sim 5-10\%$, however, its performance strongly depends on a good seed. As a result, it is often necessary to perform the reconstruction iteratively, with the optimal track of each iteration forming the starting seed for the next [50]. This adjusted version of `SplineMPE` is often referred to as *max settings* or `SplineMPEmax`. At PeV energies, the median opening angle between the true track direction and reconstructed direction for `SplineMPEmax` is approximately $0.2°$ for through-going tracks and $0.4°$ for starting tracks [9].

Above $\sim 1\,\text{TeV}$, muons primarily lose energy through stochastic processes like bremsstrahlung, pair production and nuclear interactions [9]. Rather than correcting for the wrong hypothesis of a minimally ionizing track, the latest likelihood reconstruction method `SegmentedSplineReco` splits the hypothesized track into segments and includes information on these stochastic energy losses into the spline-based PDFs. This method yields a better performance, especially for high energy starting tracks, but comes at the cost of a significantly longer runtime [9].

5.2 Cascade reconstruction

Reconstructing the neutrino direction for cascade events in IceCube is significantly harder than for track events. The typical propagation lengths of the hadronic and electromagnetic showers in these events are well below the spacing of the DOMs, which makes these events appear as almost point-like light sources [114]. The standard cascade reconstruction method is to do a likelihood fit with a *cascade template*, which contains the simulated light yield at each DOM given an interaction vertex, interaction time and neutrino direction [115]. Similarly to `SplineMPE`, these templates are stored by a B-spline fit of tabulated data. Because these templates model the cascades as point sources, they are a lot more dependent on the ice model than track reconstructions [116]. The improved description of the optical ice properties in the recently developed fourth-generation ice models has shown a remarkable advancement in the directional reconstruction of high energy cascades, bringing down the median angular resolution to $3.5-4.5°$ [116].

Part II

Methods and results

6

Methods

This chapter will introduce the methods used for the analysis presented in Chapter 7. It includes a description of the datasets used, the selection of training and validation events, and the data's pre-processing to make it suitable for machine learning. Furthermore, the exact model architectures and training hyperparameters are described. For reproducibility of this research, all the methods have hyperlinks included to the relevant scripts on my Github page. Moreover, this chapter aims to give an insight into the development of the methods throughout the thesis, not just those used for the final analysis. It describes the thought process of changes, including unsuccessful ones, to hopefully prevent others from making the same mistakes in future work. For some of the methods, their difference in performance for angular reconstruction has been analyzed in more detail and included as separate results in Chapter 7.

6.1 The dataset

The backbone of this thesis is three datasets with Monte Carlo simulated high energy neutrino events. The files have been converted from `i3` to `SQLite`-databases by former Niels Bohr Institute (NBI) student Aske Rosted. The simulation sets were produced using the `SnowStorm` method [89]. The simulations were produced early 2022, using `icetray v1.3.3` and the `Spice3.2.1` ice model (Note: this is not the newest ice model, `Spice FTPv3` [76] but these were the latest completed simulations at the time of `SQL` conversion. The `Spice3.2.1` ice model does not include the newest birefringence-based anisotropy model or updated tilt model, but is still recent enough to be used for this work). The data considered for this master thesis consists of ν_μ and $\bar{\nu}_\mu$ events in the energy range $100\,\text{GeV} - 100\,\text{PeV}$, split over three datasets. An overview of these datasets is given in Table 6.1. Full details, including ensemble parameters, are described on the IceCube Wiki. The databases include the `SRTInIcePulses` pulse map at `level 2`.

SnowStorm Event Ensemble

Dataset ID	Flavor	Energy range (GeV)	Energy spectrum	Event count	File location
22010	ν_μ	$10^2 - 10^4$	$E^{-1.5}$	47 856 945	{1}
22011	ν_μ	$10^4 - 10^6$	$E^{-1.5}$	3 695 159	{2}
22012	ν_μ	$10^6 - 10^8$	$E^{-1.0}$	789 324	{3}

Table 6.1: Description of the `SnowStorm` simulation sets used in this thesis. Full details on the datasets can be found on the IceCube Wiki.

6.1.1 Data cleaning

In addition to the neutrino signal generation, a cosmic ray (CR) background is generated during the simulation. This background is merged with the signal. After particle propagation and detector simulation, this results in a pulse map containing photons from both neutrino and CR origins. After the `level 2` processing and `SRT` cleaning, a part of these CR photons are removed. However, some of these CR photons still persist in the datasets. Figure 6.1 illustrates this, where it can be seen that there is even a significant fraction of events that have (nearly) no signal photons.

For the goal of training a model to learn to reconstruct neutrinos, it is beneficial to use a dataset containing pure signal photons. It was, therefore, decided to remove events with CR photons from the datasets. The script for this can be found at `CR_cleaning.py`. The reduced event counts and survival rates for each of the data sets are summarized in Table 6.2.

Cosmic Ray cleaning

Dataset ID	New event count	Survival rate
22010	31 529 954	0.659
22011	2 797 458	0.757
22012	588 324	0.745

Table 6.2: Overview of the new event counts and the respective survival rates for the datasets of interest after applying the cosmic ray cleaning procedure.

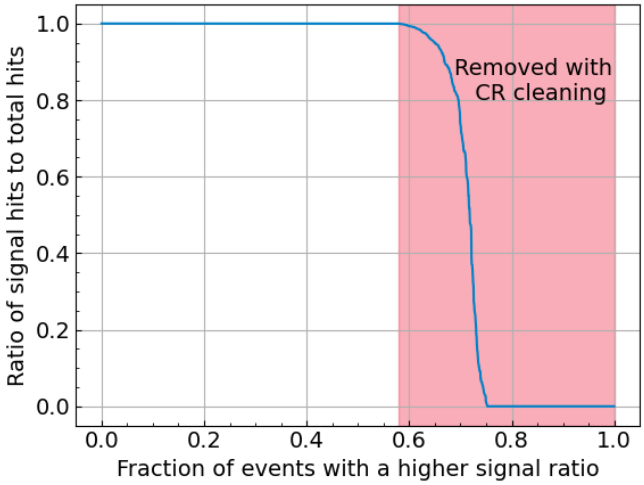


Figure 6.1: Ratio of the number of signal (neutrino) hits to the total number of hits per event. The total number of hits includes background photons from cosmic rays. Data is shown for $\sim 20\%$ of the `22011` dataset for illustration. A significant part of the events has (almost) no signal photons.

6.1.2 Event selection

Having clean data alone is not sufficient for a good training dataset. In addition to the cleaning, various selection cuts are made on the signal events themselves to select those with a clear detector characteristic. The idea is that by selecting well-reconstructible events for the training set, the machine learning model can more easily learn the characteristics of these events compared to when the training set is contaminated with events that were hard or impossible to reconstruct in the first place, like cascade events or events that miss or only clip an edge of the detector volume.

There exist copious ways of defining high energy neutrino events as well-reconstructible. One could select events based on their physics, for example, by only selecting charge current (CC) muon neutrino interactions, resulting in a muon track. Another cut could be using the detector geometry, e.g. only selecting events with their interaction vertex in or close to the detector, traveling towards the detector. Furthermore, IceCube standard filters could be used, such as the `MuonFilter_13`, to only select events whose resulting detector signal looks like a muon neutrino. Alternatively, an existing reconstruction algorithm like `SplineMPE` could be used to train the model on events that these algorithms can reconstruct well. The downside of using such a selection is that it can bias the model towards the existing reconstruction algorithm. There does not exist

a clear best answer to the question of what cuts to use. However, one should keep in mind the goal of a generalizable machine learning model. Too strict training cuts tend not to translate well to performance on the validation or test set. Furthermore, it is important not to become too dependent on Monte Carlo truth variables for cuts that are impossible to know for real data. Lastly, one should be aware that introducing many selection cuts can significantly reduce the number of available training events, which could also harm training. A more detailed analysis of the influence of the number of training events on the model performance is described in Section 7.2.1.

In this research, the influence of various event selections for training has been investigated. The first selection is a set of cuts inspired by the work of Thorsten Glüsenkamp (and therefore in the future referred to as the *Thorsten Cut (TC)*) [*personal communication, January 20, 2025*]. On top of the SRT and CR cleaning, it only considers CC events passing IceCube's MuonFilter. The second selection considered is based on `SplineMPE` and simply selects the events that `SplineMPE` is able to reconstruct with an opening angle less than 2°. Naturally, these selections have a large overlap. This is illustrated in the diagram in Figure 6.2 for the `22011` dataset specifically. A more elaborate study on event selection is performed for three subsets of the `22011` set:

1. Applying the Thorsten Cut: $\sim 59\%$ of all events survive.

2. Applying the `SplineMPE` cut: $\sim 40\%$ of all events survive.

3. Applying both (1) and (2): $\sim 33\%$ of all events survive.

The full results of this study are set out in Section 7.1. For all other models in this thesis, the Thorsten Cut is applied. Note that the fraction of events passing the TC is slightly different for the `22010` and `22012` sets. The total number of events after CR cleaning and applying the selection cut is summarized in Table 6.3.

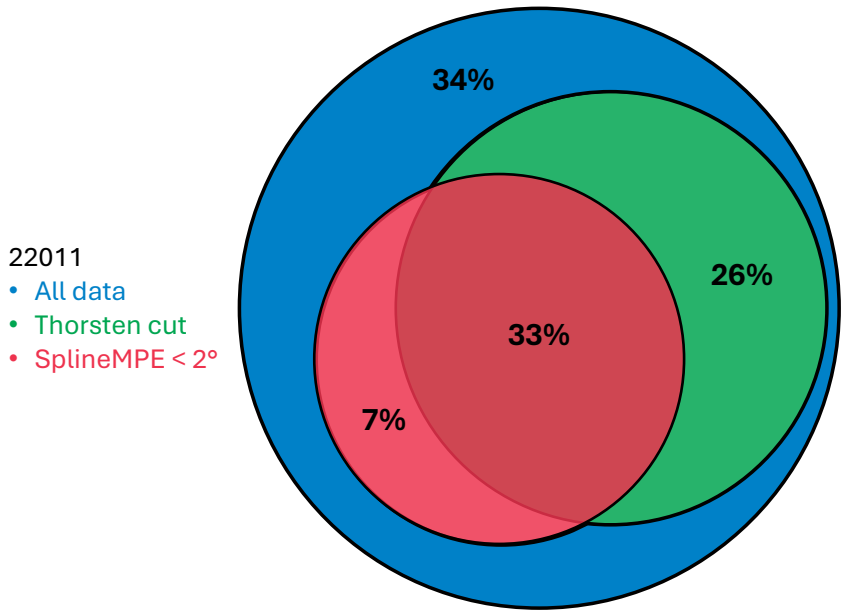


Figure 6.2: Graphical representation of the considered event selection cuts, demonstrating the overlap between the Thorsten Cut and the SplineMPE cut. Areas are not to scale.

Event counts after CR cleaning and Thorsten Cut

Dataset ID	New event count	Survival rate
22010	20 460 097	0.428
22011	2 128 033	0.576
22012	489 008	0.620

Table 6.3: Overview of the new event counts and the respective survival rates for the datasets of interest after applying cosmic ray cleaning and the Thorsten cut event selection.

6.1.3 PMT-fication

The next step in the data pre-processing is a process that will be referred to as *PMT-fication*. A downside of transformers is that their computational complexity scales with the square of the context length; see Section 4.2. For IceCube data, one would like to use the entire pulse series as the input sequence for the transformer. However, since the number of pulses for a high energy event can be over a hundred thousand, using all pulses as input sequence is computationally infeasible, even on state-of-the-art hardware. Previous transformer implementations for IceCube tried solving this problem by truncating to a maximum number of pulses and implementing tricks like *sequence bucketing* [12]. Typical truncation lengths are in the order of a couple of hundred pulses. Inevitably, by truncating the pulse series, a lot of information is thrown away. Preferably, one would instead apply some other method to reduce the sequence length to limit the loss of information.

The method explored in this work is the use of summarization features per PMT, inspired by the idea of Thorsten Glüsenkamp [*personal communication, October 8, 2024*]. This effectively reduces the maximum sequence length for an event to 5160, the current number of DOMs in IceCube. The process of summarizing all pulses of an event on a PMT basis will be referred to as *PMT-fication* for the rest of this thesis. The pipelines for PMT-fication have been co-developed in this work, with the majority of work done by *Cyan Jo*, and the code is released as a separate toolkit on the GitHub page `IcePACK`. The PMT-fication process reduces the information of all pulses hitting a single PMT in an event to 32 features. These features can be categorized into information about the position of the DOM, first pulses, quantiles of all the pulses and general characteristics of the PMT, as described in Table 6.4. The charge-weighted center of the event is calculated by

$$\vec{r}_{cwc} = \frac{\sum_i^{N_{doms}} \sum_j^{N_{pulses,i}} q_j \vec{r}_j}{Q_{event}}. \tag{6.1}$$

If features are not present for the event, they are filled with padding values `-1`. The choice for these features is motivated by physics, but there is no claim that these features are the best possible set. It could be possible that future research finds new, valuable features or deems some existing features redundant. Since the nodes of the data are now the PMTs instead of pulses, the relative DOM position to the charge-weighted center is added to give some sort of weight of importance to the nodes, assuming that PMTs close to the action contain more valuable information. The details of the first few pulses are added without summarization, as previous transformer-based reconstruction algorithms for IceCube data found this information to be important for their performance [12].

The effect of PMT-fication on the event sequence length for the three datasets is shown in Figure 6.3. While the mean sequence length of the pulse maps for the `22010`, `22011` and `22012` sets used to be 56, 600 and 8455, respectively, this is reduced by PMT-fication to 30, 102 and 356. As expected, the effect is the strongest for the `22012` dataset, which spans the highest energy range. An elaborate analysis of the effect of PMT-fication on the transformer performance is presented in Section 7.3.

PMT-fication features

Feature name	Description
	DOM-position
`dom_x`	x-position of the DOM in IceCube coordinates
`dom_y`	y-position of the DOM in IceCube coordinates
`dom_z`	z-position of the DOM in IceCube coordinates
`dom_x_rel`	relative x-position of the DOM to the charge weighted center of the event
`dom_y_rel`	relative y-position of the DOM to the charge weighted center of the event
`dom_z_rel`	relative z-position of the DOM to the charge weighted center of the event
	First pulses ($i \in [1, 5]$)
`t_i`	time of hit i on the PMT relative to the trigger window start
`q_i`	charge of hit i on the PMT
`hlc_i`	boolean value indicating if hit i was HLC
	Quantiles
`Q25`	cumulative charge of the PMT after 25 ns
`Q75`	cumulative charge of the PMT after 75 ns
`Qtotal`	cumulative charge of the PMT for all hits
`T10`	time of detection of 10% charge of Q_{total} relative to trigger window start
`T50`	time of detection of 50% charge of Q_{total} relative to trigger window start
`sigmaT`	standard deviation of detection time for all hits
	PMT characteristics
`pmt_area`	Area of the PMT
`rde`	Relative detection efficiency of the PMTs
`bad_dom_status`	Boolean flag indicating malfunctioning DOMs
`saturation_status`	Boolean flag indicating a saturation of charge readout in the PMT
`bright_dom_status`	Boolean flag indicating high charge readout in the PMT

Table 6.4: Overview of the 32 features used for the PMT-fication process, summarizing all hits on a single PMT in an event.

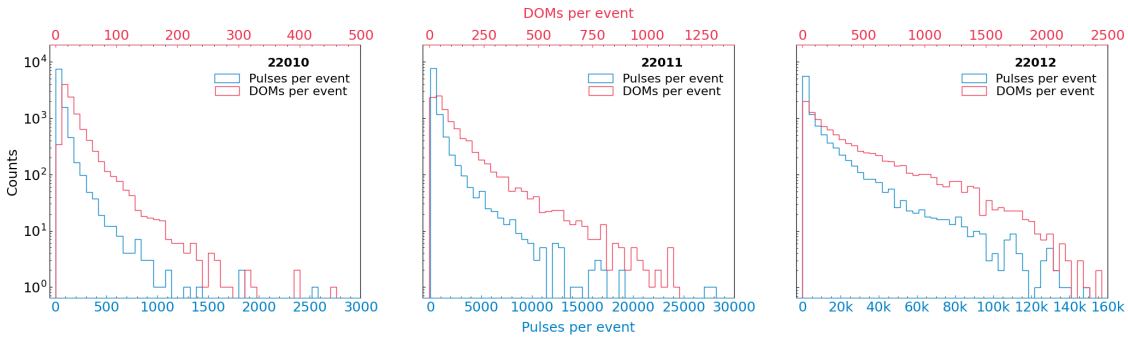


Figure 6.3: Reduction of the event sequence length by PMT-fication. The bottom axis shows the distribution of pulses per event for the three datasets, while the top axis shows the distribution of DOMs per event. This reduction allows the use of the sequence as input to transformers with minimal truncation.

6.2 Dataloading

A fast dataloader is an often underrated but absolutely critical part of a machine learning pipeline. Storing, loading and processing data in an efficient manner is pivotal to training machine learning models with large numbers of training events. Without proper data handling, one is not able to effectively utilize computational resources, especially to perform the many parallel computations on the GPU. Developing machine learning algorithms requires iterative experimentation with model architectures and hyperparameters. Optimized dataloading significantly speeds up the training time, allowing for quicker model iterations, likely resulting in better performance of the final model.

A lot of effort in this thesis was invested into developing such an efficient dataloading pipeline, especially for the PMT-fied data. This section will present that work in detail, including motivations for certain decisions, discussion of unsuccessful attempts and highlighting of key elements in the final pipeline. It introduces a novel way of event storage in `parquet` files, the development of a `PyTorch` dataset and dataloader class to load these events and explains the applied feature normalizations before passing the data to the model.

6.2.1 Data storage

As introduced in Section 6.1, the `SnowStorm` datasets used in this thesis are stored in `SQLite` databases. `SQLite` databases are lightweight embedded databases, designed for ease of use for relatively small datasets. However, `SQLite` databases are not optimized for handling large datasets, which makes querying datasets for large machine learning projects slow, especially when using advanced event filtering or sampling during dataloading. Therefore, it was decided to convert and store the datasets in a different format, `Apache Parquet`, an open-source column-oriented data storage format. `Parquet` is optimized for bulk and parallel reads, allows for efficient batch loading with `PyArrow` and selective reads make it easy to use in combination with the `PyTorch DataLoader` class.

While converting file format, it was decided to apply the PMT-fication data reduction algorithm at this stage, storing the PMT-fied events straight into the `Parquet` files. In the future, it could be nice to incorporate the PMT-fication pipeline into the dataloading pre-processor, remove this as an extra step, and allow others to use the transformer model directly on their pulse-map data. However, this would, of course, come with a reduction in dataloading speed. For the development of the models in this work, expecting the need for many training iterations, it was decided to separate the PMT-fication step and store the reduced event information directly.

To further improve training efficiency, the feature tables of the events were split up into many small files. This process is illustrated in Figure 6.4. Doing this allows entire data files to be stored in and loaded from cache (more on this in Section 6.2.2). Since the average event length scales with energy, it was decided to store $\sim$ 200k events per file for the `22010` dataset, $\sim$ 20k events per file for `22011` and $\sim$ 2 000 events per file for the `22012` dataset. Each of these files is indexed by a `shard_no`. A single `truth.parquet` file for each of the original `SQLite` parts was created, containing physics truth variables like the true direction, but also containing information on where to find the features of the particular event, like `shard_no` and start and end row within the PMT-fied feature file.

It is common practice to use the truth variable `event_no` as a unique classifier for the event during training. However, each of the `SnowStorm` datasets resets the index of the `event_no`, starting at 0 and counting up. To be able to combine files from different datasets without duplicate event numbers, a new unique `event_no` was added to the truth information of each event in the `truth.parquet` file. The new event numbers are a concatenation of `dataset_ID`, part number

and original event number, for example:

$$\text{new\_event\_no} = \underbrace{1}_{\text{SnowStorm}} \quad \underbrace{12}_{\text{ID}} \quad \underbrace{0004}_{\text{part\_no}} \quad \underbrace{01234567}_{\text{event\_no}}.$$

Here, the first number indicates if the event is a `SnowStorm` (1) or `Corsika` (2) event. The next two digits indicate the `dataset_id` (here, 12 refers to set 22012). The next four digits indicate the `part_no`, and the last eight represent the original event number.

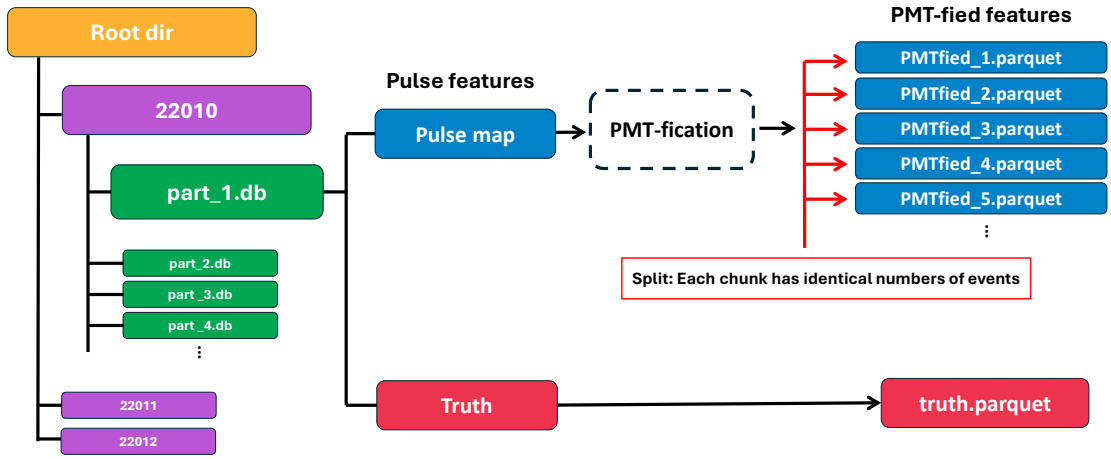


Figure 6.4: Illustration of the file storage structure. Individual `SnowStorm` datasets are divided into several `SQLite` database parts. Each of these parts is converted to the `Parquet` storage format. In this process, the PMT-fication algorithm is applied to the pulse maps, and the features are stored in many small files of a fixed number of events. Source: Adapted version of figure by *Cyan Jo*.

6.2.2 PyTorch dataloader

A lot of time in this thesis was invested into writing an efficient `PyTorch` dataloader, which can handle PMT-fied IceCube data for transformer training. Many iterations of improved dataloaders resulted in a step-wise increase in the number of loaded events per second from a couple of events to a couple of thousand events. The relevant classes and functions can be found in `dataset.py` and `dataloader.py`. The dataset class is the most important for efficient dataloading. When iteratively calling the dataset class, it returns a `PyTorch Geometric` data object, which contains both a feature tensor and a label tensor from a single event. By wrapping the dataset class in a dataloader class, these data objects are bundled into batches.

Key to the performance of the dataset class is the usage of data from *cache*. This is a space for temporary storage of memory, built into the CPU hardware. By using cached data, the feature and truth tables have to be read from the disk only once, which significantly speeds up loading the events. Unfortunately, this comes with the drawback that events must be read sequentially, and shuffling events among batches every epoch is no longer possible. The truth table contains information about the start and end rows of the event in the feature table, and by slicing the feature table, the corresponding rows are selected. At first, the data was read into `Pandas` arrays, however, this turned out to be very slow. Although `Pandas` is easy to use, it has a large memory overhead and is inefficient with handling large sets of data. Instead, the data is read into `PyArrow` tables, the Python interface to `Apache Arrow`, using an in-memory data format optimized for columnar data [117]. `PyArrow` allows for zero-copy read and write operations, efficiently accessing data without duplicating it in memory. This allows for optimized slicing and efficient pre-processing operations on the columns (more on pre-processing in Section 6.2.3).

The dataset class is wrapped with the default `PyTorch` dataloader class. The class uses a custom `collate function`. This function handles the batching of events, but, more importantly, also pads or truncates the events to a fixed sequence length. This is needed since storing in batches with `PyTorch` requires each event to have the same dimensions. A fixed sequence length of 256 PMTs is used. If the event is shorter, the torch tensor is filled with padding values. If the event is longer, the PMTs are sorted in decreasing order on cumulative charge and truncated after the first 256 sequence elements. The original sequence length of the event is also passed on to the model since this information is required for the *attention mask* (see Section 6.3 for more details). For good performance, it is essential to use `shuffle = False` and `persistent_workers = True` when initializing the dataloader class since both are required to allow usage of feature data from cache memory. Note that in `PyTorch`, the default for `persistent_workers` is `False`.

The dataloader allows the use of multiple workers, or subprocesses, to load data in parallel. Each of the workers puts batches of data into a queue, and the main training loop uses the batches from this queue. This can significantly speed up dataloading, since it reduces the time the GPU spends idle, waiting for data to be loaded. However, using too many worker processes can cause CPU contention or RAM constraints [118]. The optimal number of workers depends on the type of storage, the size of the dataset and the memory constraints of the system. To find the right number of workers to use, the *strong* and *weak scaling* behavior of the dataloader was investigated. For strong scaling, the workload was kept constant at loading 1000 batches at batch size 64. For weak scaling, the workload was scaled with 100 batches of batch size 64 per worker. The speedup for strong scaling was calculated by dividing the single-worker runtime by the total runtime for i workers:

$$S_{i,\,strong} = \frac{t_{(1\,worker)}}{t_{(i\;workers)}}. \tag{6.2}$$

For weak scaling, the speedup was calculated by multiplying the ratio of workload size by total runtime for i workers with the inverse for a single worker:

$$S_{i,\,weak} = \frac{W_{(i\,workers)}}{t_{(i\,workers)}} \cdot \frac{t_{(1\,worker)}}{W_{(1\,worker)}}. \tag{6.3}$$

The results are shown in Figure 6.5. Overall, the dataloader shows great scaling behavior, with both the strong and weak speedups close to the ideal speedup. Still, as expected, a clear drop-off can be seen at a high number of workers. This indicates that using more workers will probably be redundant (moreover, not using all CPU cores will probably also be appreciated by the other users of the cluster). For most of the thesis, using 6 workers was found to be sufficient to maximize GPU utilization during training.

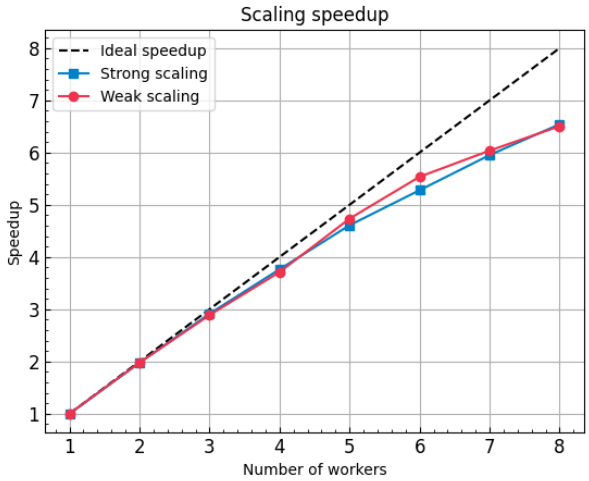


Figure 6.5: Strong and weak scaling behavior for the `PyTorch` dataloader developed for this thesis, showing speedup as a function of the number of worker processes.

6.2.3 Data pre-processing

The normalization of input data is crucial for neural network performance. It enhances numerical stability, accelerates convergence by allowing optimization algorithms to work more efficiently during backpropagation, and overall improves generalization of the results [119]. Therefore, after PMT-fication, the 32 input features are pre-processed before being used as input to the transformer. Although it would have been more efficient to store the data in a pre-processed fashion, it was decided to implement this step during data loading instead, such that the values in the dataset files keep their physical meaning. The performance harm because of this is minimal. The code for the feature pre-processing can be found in `dataset.py`. The following transformations are applied to the input:

- `DOM_pos = DOM_pos / 500`
- `t_i = (t_i - 1e4) / 3e4`
- `q_i = log10(q_i)`
- `Q_i = log10(Q_i)`
- `T_i = T_i / 1e4`
- `sigmaT = sigmaT / 1e4`
- `pmt_area = pmt_area / 0.05`
- `rde = (rde - 1.25) / 0.25`

The transformations are similar to the ones applied by `GraphNeT`. They do not perfectly normalize the data to have a mean of zero and a standard deviation of one, but they sufficiently shift the feature distributions to be in the range of order one.

6.3 Model architecture

For this work, a new machine learning model has been developed from scratch. The model uses a transformer encoder inspired by the original transformer model as described in the paper 'Attention is all you need' [103] (introduced in Section 4.2). The transformer encoder is followed by an aggregation layer and a single linear layer for regression. The model class can be found in `model.py`. A graphical representation of the model is given in Figure 6.6. This section will give a detailed description of each of the model layers. The model is designed to take PMT-fied IceCube events as input and output three numbers, the (x, y, z)-coordinates of a vector representing the neutrino direction. However, in principle, the model accepts a numerical input of arbitrary length and can output a prediction vector of arbitrary length, allowing the architecture to also be used for other tasks.

6.3.1 Input embedding and positional encoding

The input of the model consists of a `torch tensor` of dimensions $d_{batch} \times d_{seq} \times d_{features}$. By default, $d_{seq} = 256$ and $d_{features} = 32$, as described in Sections 6.2.2 and 6.1.3 respectively. In the input embedding, each sequence element of size $d_{features}$ is mapped to a vector of size d_{emb} with a single `torch nn.Linear` (trainable) layer. The same transformations are applied to each of the sequence elements. The positional encoding maps the index of the sequence element to a vector of d_{emb} using `torch nn.Embedding`. This is a trainable lookup table. The positional encoding is the

same for all events in a single batch but is modified through backpropagation between batches. Finally, the input to the transformer encoder block is formed by summing the input embedding and positional encoding, resulting in a tensor of dimension $d_{batch} \times d_{seq} \times d_{emb}$.

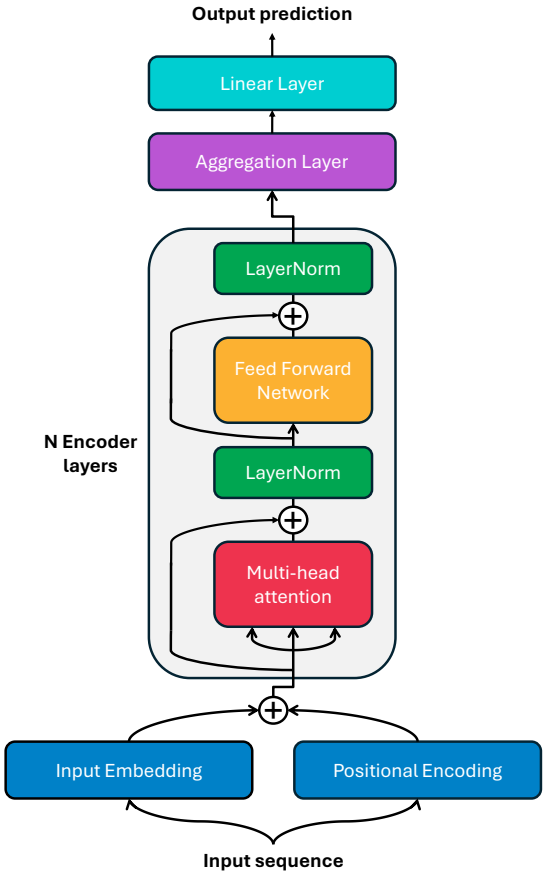


Figure 6.6: Illustration of the machine learning model architecture developed in this thesis. The model uses a transformer encoder followed by an aggregation layer and a linear prediction layer.

6.3.2 Encoder block

Each encoder block starts with a multi-head attention layer. It starts with three parallel `nn.Linear` layers projecting the input of d_{emb} to a query, key and value vector, each also at dimension d_{emb}. These query, key and value vectors are split into h vectors of size d_{emb}/h, with h the number of heads in the multi-head attention. The attention weights in each head are calculated by taking the inner product between the query and key tensor and normalizing, similar to as described in Equation 4.22. The resulting weight tensor in each head has dimension $d_{batch} \times d_{seq} \times d_{seq}$. The next step is to apply an attention mask based on the original sequence length of each event in the batch. An illustration of this mask is shown in Figure 6.7. This is very important to the functionality of the transformer model since it ensures that padded sequence elements are removed from the attention weight tensor. It ensures that every weight outside the upper left block of the original event length is set to negative infinity. Next, the `torch softmax` function is applied, which transforms all weights of negative infinity to 0 while rescaling the other weights between $[0, 1]$. The attention mask of this model should not be confused with the *causal attention mask* often used in transformer decoders.

The output of a single head is calculated by a matrix multiplication between the attention weights and the value tensor, resulting in an output tensor of dimension $d_{batch} \times d_{seq} \times d_{emb}/h$. The outputs of all parallel heads are concatenated to a tensor size $d_{batch} \times d_{seq} \times d_{emb}$, retrieving the shape of the input before the multi-head-attention layer. The concatenated tensor is fed to a `nn.Linear` projection layer from d_{emb} to d_{emb}, allowing learned information to mix across heads. A `torch nn.Dropout` is applied to the output, randomly zeroing a fraction of the attention weights to prevent overfitting.

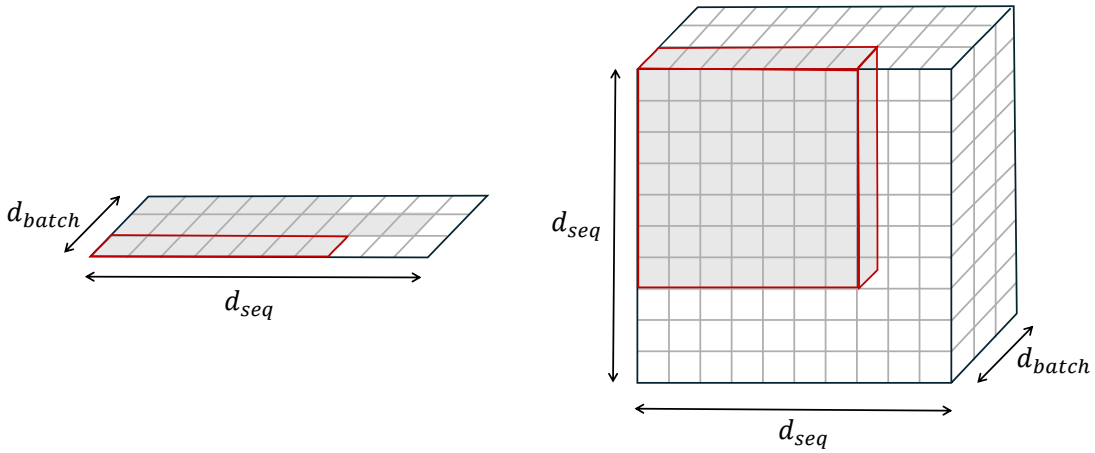


Figure 6.7: Illustration of the attention mask used in the model developed in this work. The left illustrates the variable sequence length of events within a batch, padded to a fixed length. The right shows how the length of the event is translated to the mask shape in the attention weight tensor. Only weights within the attention mask are taken into account.

Through a residual connection, the input and output tensors to the multi-head attention are summed. A `torch nn.LayerNorm` is applied on the embedding dimension. Next, the tensor is passed to a two-layer feed-forward neural network. The first layer maps from d_{emb} to $4\,d_{emb}$, while the second layer maps back to d_{emb}. In between the two layers, a `ReLU` activation function is used. After the second layer, once more, a dropout is applied. With another residual connection, the input to the feed-forward network is summed to the output, after which one more `nn.LayerNorm` is applied. The output of the encoder block has dimension $d_{batch} \times d_{seq} \times d_{emb}$.

The output of one encoder block serves as the input for the next encoder block. A total of N encoder blocks are used. If not explicitly mentioned otherwise, a total of $N = 6$ blocks are used. The terms 'encoder block' and 'encoder layer' are used interchangeably in the rest of this work.

6.3.3 Aggregation and linear layer

After the last encoder block, the output is passed to an aggregation layer. Here, the tensor is aggregated over the sequence dimension, reducing it to a $d_{batch} \times d_{emb}$ dimension. Experiments comparing a mean and max pooling layer as the aggregation method were conducted without noticeable performance differences. For simplicity, mean pooling has been used for all results in this thesis. Also for aggregation, it is important to take into account the original sequence lengths of the events. Once again, a mask for the sequence length was used to only aggregate non-padded sequence elements. After aggregation, the tensor is passed to a single `nn.Linear` layer, acting as the regressor of the model. This linear layer maps from d_{emb} to the output dimension d_{output}. The final output prediction has dimension $d_{batch} \times d_{output}$.

6.4 Training procedure

This section aims to give an overview and motivation of the training procedure for the transformer models in this thesis. It will describe the hardware used, motivate the choice of model hyperparameters, introduce the optimizer and learning rate scheduler, describe the various loss functions used and explain how the data was separated into a training, validation and testing set for sound scientific practices. Of course, the training procedure has not been constant for all results in this work since many hyperparameter adjustments were made along the way. Therefore, this section will set out the baseline training procedure, while the supplementary information about the individual models is listed in Appendix A. The training procedure can be adjusted by adapting the config.yaml file.

6.4.1 Hardware

The models were trained on a high-performance compute cluster with a 64-thread *AMD Ryzen Threadripper 3970X 32-Core* processor and two *NVIDIA GeForce RTX 3090* GPUs. The processor contains 32 physical cores, each supporting two threads, and operates in the frequency range from 2.2 GHz to 3.7 GHz. The cluster is shared with other users, and fair-use agreements limit usage to a maximum of 16 threads per user. The data was stored on a *Lustre* network file system, allowing for parallel I/O across multiple nodes.

The models were trained using a single RTX 3090 GPU. If both GPUs were available, two independent models could be trained simultaneously to save time. Although the RTX 3090 is a consumer-grade GPU mainly designed for gaming, its 24 GB of VRAM makes it suitable for training relatively large machine learning models such as transformers. The GPUs use NVIDIA's *Ampere* architecture, which allows mixed-precision (FP16) training. This results in faster training with reduced memory, without sacrificing much accuracy. The GPUs maximum performance is ~ 35 TFLOPS ($35 \cdot 10^{12}$ floating point operations per second). With the optimized dataloaders, as introduced in Section 6.2, GPU utilization approached 100% during training.

While training transformers on the RTX 3090 was doable for large models, both the speed and memory of the consumer-grade GPU became limiting factors. In comparison, data center GPUs like the *NVIDIA A100* offer up to 80 GB of VRAM and 312 TFLOPS, making them better suited for large-scale training.

6.4.2 Model hyperparameters

Hyperparameters can critically influence the performance of a machine learning model. Moreover, by systematically changing hyperparameters and comparing performances, the model's robustness can be analyzed. At the start of the thesis, to develop the model and test its capabilities to learn, small models were trained on very small training sets. Generalization of performance was not yet of interest; rather, the goal was to show the transformer to be capable of overtraining on a small sample of IceCube data. Once this was achieved, the number of model parameters and training set size were increased, aiming to generalize learning to the validation set.

For most analyses in this work, the following model hyperparameters were used: a maximum sequence length of 256, a batch size of 64, an embedding dimension of 128, 4 parallel heads in the multi-head attention and a total of 6 encoder layers. For regularization, a dropout of 0.1 was used. These values will be referred to as the baseline hyperparameters, and without explicitly stating the use of different values, it can be assumed that these hyperparameters were used. Little exploration of the maximum sequence length was done, but the other hyperparameters were determined by gradually increasing values. The baseline hyperparameters form a medium-sized

model ($\sim$ 1.2M parameters) that generalizes well while still being relatively fast to train ($\sim 5-20$ hours depending on training set size) and light on the memory usage but with high GPU utilization ($> 80\%$). Models were trained to the convergence of performance on the validation set, which for most models took ~ 100 epochs.

The influence of the hyperparameters defining model size has been studied in more detail, presented as a separate result in Section 7.2.2. For the final results, training set sizes were increased to the limit of available Monte Carlo events, and the model parameter size was scaled to the maximum allowed by the hardware. Learning rates were scaled down accordingly.

6.4.3 Loss function

Various loss functions were used to calculate the optimization metric during training, with varying degrees of success. All of them are described in loss.py. At first, attempts were made to use the opening angle

$$\Delta\theta = \cos^{-1}\left(\frac{\vec{y}_{pred}\cdot\vec{y}_{true}}{||\vec{y}_{pred}||\,||\vec{y}_{true}||}\right) \tag{6.4}$$

between the reconstructed and target vector directly as the loss metric, since the goal is to minimize this quantity. However, some problems exist with this loss function. Mainly, the arccos introduces some nasty derivatives during back propagation, with very large gradients at -1 and 1. Because of this, using the opening angle directly leads to instabilities during training. An alternative would be to use the *cosine similarity* loss instead:

$$\mathcal{L}_{cos} = 1 - \cos(\Delta\theta) = 1 - \frac{\vec{y}_{pred}\cdot\vec{y}_{true}}{||\vec{y}_{pred}||\,||\vec{y}_{true}||}. \tag{6.5}$$

An even more advanced version of the cosine similarity loss uses the *von Mises-Fisher* (vMF) distribution in 3D [120]. This distribution is like a Gaussian distribution, but on the surface of a sphere; ideal for the prediction of unit vectors. As a loss function, the negative log-likelihood is minimized:

$$\mathcal{L}_{vMF} = -\kappa\cdot\cos(\Delta\theta) + ln(C_3(\kappa)). \tag{6.6}$$

Here, κ is a measure of the model's uncertainty on the prediction, and $C_3(\kappa)$ is the normalization constant of the vMF distribution. This loss was also used for the baseline solution in the IceCube Kaggle competition [12]. If a fixed κ is used, the normalization constant can be dropped during optimization, and the vMF loss becomes the ordinary cosine similarity.

Both loss functions were implemented, and attempts were made to train the transformer model with them, however, this was without success. The stability at the start of the training was fine for both, but after a while, the models would diverge and get stuck in some local minimum without escaping for the rest of the training. Even at tiny learning rates, this behavior persisted. Instability with the vMF loss function is not uncommon, often caused by the Bessel functions (part of C_3), leading to vanishing or exploding gradients. A numerical approximation to C_3 was used for large κ to enhance stability, but improvements were minor. Unfortunately, a thorough mathematical explanation of these instabilities in training complex machine learning models does often not exist.

The loss function that turned out to give the best results in this thesis was a simple Euclidean distance loss between the predicted and true vector heads:

$$\mathcal{L}_{euclidian} = ||\vec{y}_{pred} - \vec{y}_{true}||. \tag{6.7}$$

The predicted vectors were not normalized, so the model had to learn to predict unit vectors by itself. This slows down the learning of direction at the beginning of the training but has the added benefit of acting as a self-regularization against the prediction of vectors with a large norm. Even

though this loss function ignores the spherical geometry of the task, its simplicity resulted in stable training, and the performance generalized surprisingly well to minimization of the opening angle. All presented results used this loss function for training.

6.4.4 Optimizer

Optimization algorithms are responsible for updating the model parameters based on the gradients after backpropagation, playing a crucial role in the convergence of deep learning models. For the transformer models trained in this work, the `AdamW` optimizer was used. This is an improved version of the `Adam` optimizer, where weight decay is decoupled from the gradient update process [121]. Weight decay is a method to penalize model weights from growing large, improving training stability. Both `Adam` and `AdamW` use the gradient momentum and per-parameter adaptive learning rates to update the model weights. However, while `Adam` adds a *L2 regularization term* to the loss (therefore, this term is part of the gradient), the `AdamW` optimizer applies weight decay directly when updating the model parameters.

The `AdamW` optimizer updates the first and second moments of the gradient by:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{6.8}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{6.9}$$

where g_t is the gradient of the weights on the batch. Running averages of the moments are calculated to adapt the learning rate for each parameter:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{6.10}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{6.11}$$

The `AMSGrad` variant of the optimizer was used [122], which prevents an increase in effective learning rate due to a decreasing $\hat{v}_t$ by keeping track of the maximum $\hat{v}_t$:

$$\hat{v}_t^{max} = \max(\hat{\mathrm{v}}_{\mathrm{t-1}}^{\mathrm{max}}, \hat{\mathrm{v}}_{\mathrm{t}}). \tag{6.12}$$

The weights are then updated by combining the first and second moments with weight decay as:

$$w_{t+1} = w_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t^{max}} + \epsilon} - \eta \lambda w_t. \tag{6.13}$$

Here, η is the learning rate, and λ is the weight decay coefficient. The ϵ term is added for numerical stability.

In this thesis, a small weight decay coefficient of $\lambda = 1 \cdot 10^{-4}$ was used since this was enough to make the model's validation loss converge stably, even when training for many epochs (200+). For the other parameters, the `PyTorch` default values were used with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \cdot 10^{-8}$. The learning rate η was determined by the learning rate scheduler.

6.4.5 Learning rate scheduler

The learning rate is the most important hyperparameter when training a machine learning model. A too-high learning rate causes the model to jump around the loss landscape and never converge, while a too-small learning rate makes the training process slow or can cause the model to get stuck in local minima. However, the ideal learning rate likely changes depending on the stage of training. A learning rate scheduler adapts the learning rate over time. For the models trained in this work,

a `OneCycle` learning rate scheduler was used. This scheduler starts with a short warm-up phase where the learning rate is quickly increased to allow exploration of the loss landscape in the early stage of training. In the next phase, a cosine annealing strategy is used, reducing the learning rate to some minimum (orders of magnitude smaller than the initial learning rate). This phase allows for fine-tuning and improves the chance of converging to the minimum.

The maximum learning rate of the `OneCycle` was scaled to the model size. For the baseline model hyperparameters, a maximum learning rate of $\eta = 5 \cdot 10^{-4}$ worked well. Learning rates were updated after each batch. The first 10% of steps were used for increasing the learning rate, the other 90% for decreasing. The other parameters were kept at the `PyTorch` default values. One drawback of using the `OneCycle` learning rate scheduler is that it requires a fixed number of training epochs, not allowing the usage of *early stopping*. To still be able to backtrack to the optimal model weights after training, the weights of both the best and last model were logged during training based on the loss on the validation set.

6.4.6 Train / validate / test

To ensure the generalization of a machine learning model to unseen data, it is important only to use a part of the available data for training while keeping the rest of the events aside in a validation and test set. These events can be used to get an idea of the model's performance on unseen data. If training performance is significantly better than performance on these events, this is a clear sign of overtraining. To optimize the training hyperparameters, for example, to prevent overtraining, both the training and validation sets are used. However, in theory, this could lead to the overfitting of hyperparameters to the performance on the validation set. Therefore, it is good scientific practice to present the results of the final model on completely unseen events, the test set.

Throughout all of this work, parts 4 of the `22010`, `22011` and `22012` datasets have been kept aside as validation sets. For testing, parts $\{10\}$, $\{9\}$ and $\{10, 11\}$ were used for each of the datasets respectively. The choice for these parts was arbitrary but consistent. Most results in Chapter 7 are presented on the validation set, as they were used to improve the model. The results of the final transformer model, presented in Section 7.4, are shown for the test set. The test set was kept blinded until the final transformer model finished training.

7

Results and discussion

This chapter presents the main results of this work, providing a detailed analysis of the performance of the trained transformer models for angular reconstruction of high energy ν_μ and $\bar{\nu}_\mu$ events in IceCube. It examines both the absolute reconstruction ability by presenting the opening angle between the true and reconstructed directions, as well as the computational time and resources required for the training. The results build upon one another, ultimately motivating the design choices for the model used for the final results presented in Section 7.4.

The chapter begins with an analysis of the event selection used for training. Next, the transformer scaling laws are studied by investigating the influence of the training set size and model size on model performance and training times. This is followed by a comparison of using PMT-fied events versus pulse maps as input data for training the transformer. Finally, all findings are integrated into a single large-scale model for reconstructing the direction of muon neutrino events within the $100\,\text{GeV} - 100\,\text{PeV}$ energy range in IceCube. The angular resolution of this final model is compared to the traditional reconstruction algorithm SplineMPE.

7.1 Event selection

Attempts to train the transformer model on all types of events in the dataset were unsuccessful and led to models getting stuck at predicting the mean. As introduced in Section 6.1.2, selecting the right subset of the dataset for training can be very important for the model's performance. Training on clear and well-reconstructible events can help the model converge in learning. However, doing this also increases the risk of overtraining, resulting in poor generalization to the complete dataset. This section will present the results of a performance analysis on the three selection cuts discussed in Section 6.1.2:

1. The *Thorsten Cut*: Selection of CC events passing the MuonFilter.
2. The *SplineMPE Cut*: Selection of events SplineMPE can reconstruct within 2° of the true neutrino direction.
3. The intersection of (1) and (2).

To keep the training times of the models manageable, only a subset of the total available training data was used for this event selection study. Cosmic ray cleaning (see Section 6.1.1) was applied to all training samples. Each model was trained to convergence on the same dataset parts, in the $10\,\text{TeV} - 1\,\text{PeV}$ energy range. The training set size was approximately 1 million events, with the most training events for the Thorsten Cut, since this is the loosest selection criterion. To mitigate the effects of different ratios of starting tracks to through-going tracks among the samples, the models were trained and the performance was evaluated on the reconstruction of the lepton track direction. Other than the selection cuts, all other training parameters were kept constant. A complete description of the model parameters is given in Appendix A.

7.1.1 Performance on raw data

Figure 7.1 shows the opening angle distribution in degrees on the validation set, including a zoom-in for small opening angles. The median of the entire distribution is indicated by a dashed line. The events in the validation set are the same for each of the models. The set consists of *raw* events in the same energy range, without any further cosmic ray cleaning or selection applied. The distribution of the opening angle for all three models shows a sharp peak at small angles but also has a tail with misreconstructed events. This tail is not surprising since the validation set contains many low-quality events or detector signatures that have not been seen in training. The mean, 25%, 50% and 75% percentiles of the opening angle for each of the models are summarized in Table 7.1. Interestingly, the model that has been trained with the event selection using the intersection of the other two selections is performing the worst on all four measures in the raw validation data, even though one would expect this training set to contain the highest quality training events. Clearly, the selection cut was too strict, and its performance does not generalize well to the entire dataset. The model using the SplineMPE selection cut has a slightly sharper peak at low opening angles than the model using the Thorsten Cut, likely because this model has specifically been trained on events that can be reconstructed with high accuracy. However, this comes with a trade-off on the other events, resulting in a thicker tail to the opening angle distribution than the model using the Thorsten Cut. The significantly higher mean and 75% percentile are clear indicators of this.

Performance event selections on raw data

Selection cut	#Events in validation	Mean	25%	50%	75%
Thorsten Cut	404 014	17.5°	0.88°	2.20°	13.15°
SplineMPE $< 2°$	404 014	22.2°	0.83°	2.04°	21.24°
TC + SplineMPE	404 014	23.2°	0.85°	2.26°	26.51°

Table 7.1: Summarization of the opening angle between true and reconstructed lepton direction for models trained with different event selection cuts. All models have been trained to convergence on the same dataset parts and with the same hyperparameters. The models are validated on a validation set with raw events, without further cleaning or selection.

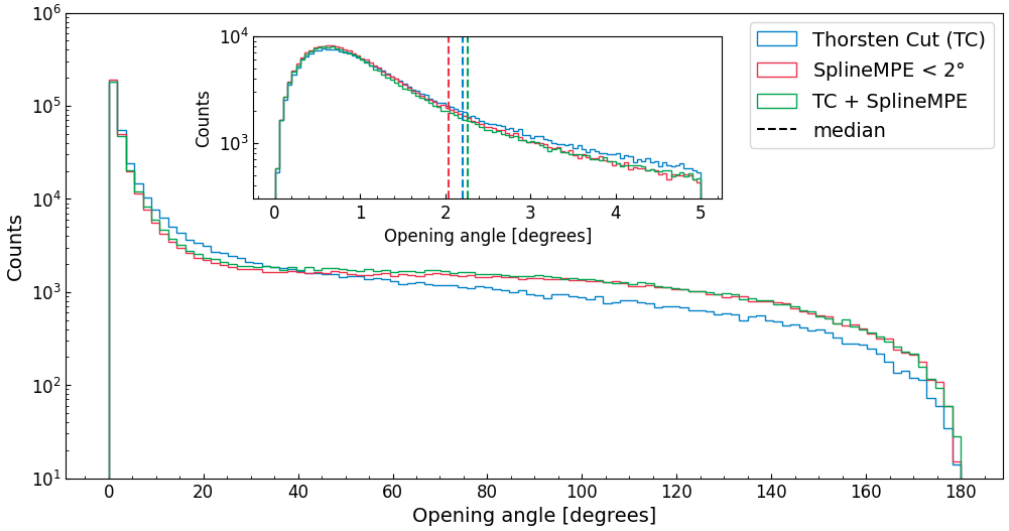


Figure 7.1: Distribution of the opening angle between the true lepton direction and the reconstructed lepton direction for models trained with different selection cuts on the training set. The models are validated on the same uncut and uncleaned validation set. The median of the models is indicated with a dashed line in the zoomed-in plot.

Figure 7.2 shows a 2D histogram of the distribution of the error in both the zenith and azimuth angle within the $[-5, 5]$ degree range for each of the three models. The error is calculated by $\psi_{true} - \psi_{pred}$. The mode of the distributions is indicated by a red cross. These align well with $(0, 0)$, showing that the models are not biased in over-predicting in a certain direction of the truth. The distributions are slightly wider for the error in azimuthal direction than zenith direction, which is similar to what is seen for low energy reconstruction models [123]. This is likely an effect of the IceCube geometry; the density of DOMs is significantly higher in the z-direction than in the (x, y)-plane, allowing for more precise vertical reconstruction.

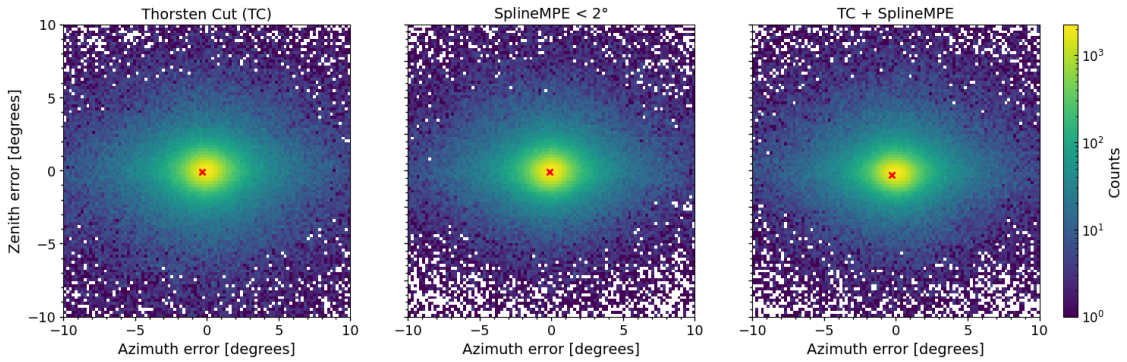


Figure 7.2: 2D histograms showing the zenith and azimuthal angle error distributions for various event selection cuts. The angle error is defined as $\psi_{true} - \psi_{pred}$. The red cross indicates the mode of the distribution.

7.1.2 Binned performance

Next, the influence of the different selection cuts on the performance on raw data is investigated as a function of neutrino energy, true azimuthal and true zenith direction. Possibly, one of the selection criteria could outperform the others for events in a certain energy range or from a specific direction. To do so, the total distribution of Figure 7.1 has been masked for certain bins, and the median of the masked distribution was calculated. The results for energy, azimuth and zenith direction can be seen in Figures 7.3, 7.4 and 7.5, respectively. The shaded area indicates the 68% confidence interval on the median. This uncertainty was calculated by combining the total number of events per bin with a binomial confidence interval (since the position of the true median follows a binomial distribution with success probability $p = 0.5$). This method has been used for all binned confidence intervals in this thesis and is described in detail in Appendix B.

The median opening angle evolves similarly for all three selection cuts as a function of energy and azimuth. As expected, more energetic events are easier to reconstruct. However, since the energy spectrum follows a power law, the number of training events also decreases with energy. This is likely the reason for the slight increase in the median opening angle in the last energy bin; the model has simply seen very few of these events during training. The results of Figure 7.5 are remarkable, however, showing a clear difference in the opening angle as a function of true zenith angle for the Thorsten Cut model compared to the others. All models are doing worse for the downgoing events than the upgoing events, but the model trained on Thorsten Cut data is doing approximately a factor of two better than the SplineMPE models. This is probably an effect of the low fraction of downgoing events being reconstructed within 2° by SplineMPE, once again giving the models few training events in this range. This confirms the idea that too strict selection cuts can notably harm the generalization of performance, even within specific ranges of the data.

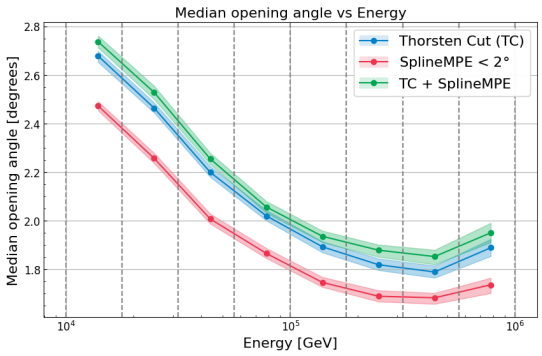


Figure 7.3: Median opening angle as a function of neutrino energy for models with different training selection cuts. Dots indicate the sample median, and shaded bands indicate the 1σ confidence interval on this median.

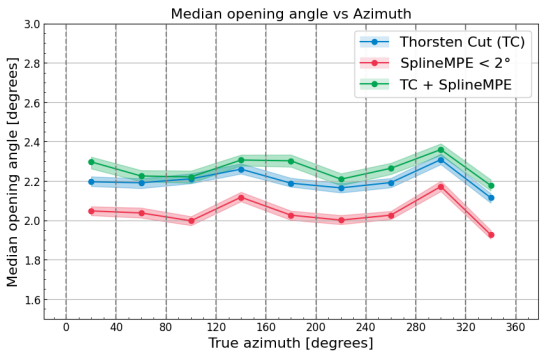


Figure 7.4: Median opening angle as a function of true azimuthal direction for models with different training selection cuts. Dots indicate the sample median, and shaded bands indicate the 1σ confidence interval on this median.

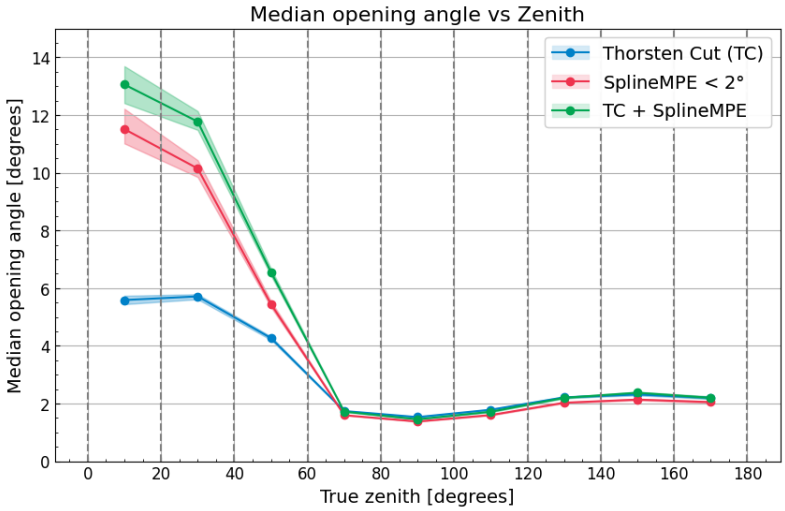


Figure 7.5: Median opening angle as a function of true zenith direction for models with different training selection cuts. Dots indicate the sample median, and shaded bands indicate the 1σ confidence interval on this median.

7.1.3 Performance on selected set

To further explore the generalization of the models, it is useful to also consider their performance on the type of events they have been trained on and compare this to performance on the raw data. Figure 7.6 shows the distribution of the opening angle of the three models, this time validated on the subset of the validation set which meets the respective selection criteria. The mean and percentiles are also summarized in Table 7.2. Of course, the opening angles on the selected validation sets are much smaller than those on the raw validation set. The long tails for the two models using the SplineMPE selection criterion are almost completely gone. This confirms the idea that these tails in Figure 7.1 were indeed caused by the events SplineMPE struggles to reconstruct. As expected, the stricter the event selection cut the better the performance on those events.

Performance event selections on selected data

Selection cut	#Events in validation	Mean	25%	50%	75%
Thorsten Cut	237 392	5.38°	0.68°	1.29°	3.21°
SplineMPE < 2°	164 579	1.11°	0.52°	0.85°	1.35°
TC + SplineMPE	134 060	1.06°	0.51°	0.83°	1.30°

Table 7.2: Summarization of the opening angle between true and reconstructed lepton direction for models trained with different event selection cuts. The models are validated on the subset of the total validation set that meets the same selection criteria as the training events.

To summarize, it can be concluded that the event selection used for the training data has a substantial influence on the model performance. There is no one correct answer to what selection is best to use since this depends both on the goal of the model and the available training events. In general, models trained on a specific subset of events are good at reconstructing those types of events, but the smaller the training subset, the poorer the performance generalizes to other events. A possible solution could be to train a model on bulk data with a loose selection cut and then follow this with a fine-tuning training on the specific events of interest.

For all other models in this thesis, it was decided to use the Thorsten Cut for the event selection. The main motivation for this decision is twofold: the Thorsten cut does not bias the models to another reconstruction algorithm while also having the highest survival rate, allowing for bigger training sets. The importance of training set size will be discussed in detail in the next section.

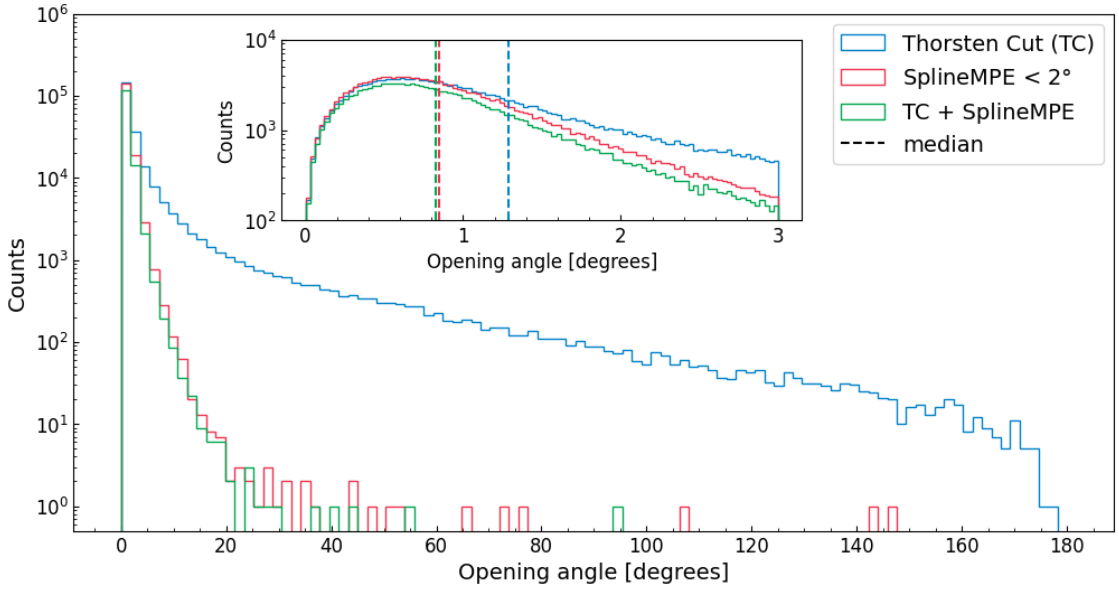


Figure 7.6: Distribution of the opening angle between the true lepton direction and the reconstructed lepton direction for models trained with different selection cuts on the training set. The models are validated on the subset of the total validation set that meets the same selection criteria as the training events. The distribution has significantly smaller tails than when validated on the raw validation set.

7.2 Transformer scaling laws

To understand the potential of the transformer model, it is important to investigate the influence of model scale and training set size. Transformers were designed to efficiently handle large input sizes in parallel [103], and research by OpenAI has shown that transformer performance improves with model size, training set size, and compute for language processing [124]. Modern large language models can have hundreds of billions of parameters and are trained on billions of web pages, books, articles or other text sources. Of course, training larger models comes with the drawback of higher computational costs. Moreover, there are no guarantees that the empirical transformer scaling laws for language processing also hold for IceCube event reconstruction.

This section studies these transformer scaling laws for the angular reconstruction of high energy IceCube events. The influence of both the training set size and model size on the opening angle is investigated. Furthermore, the effects of these parameters on the training time are considered. The goal of this study is to get an idea about the optimal model and training set size while keeping training times feasible for a single GPU.

7.2.1 Size of the training set

Creating Monte Carlo datasets with millions of training events is a time- and resource-intensive process. It is, therefore, valuable to investigate the scaling behavior of the transformer model with the size of the training set. To do so, a transformer model of fixed parameter size N has been trained on various sizes of training sets D. The batch size was kept constant. A minimum training size of one hundred thousand events was chosen to assure generalization of performance to the validation set. Events in the energy range from $10\,\text{TeV} - 1\,\text{PeV}$ were used. The Thorsten Cut was applied to the data. All models were trained to convergence and validated on the same type of events as trained on. A complete description of the used datasets and model hyperparameters is given in Appendix A. The performance was evaluated on the opening angle between the predicted direction and the true neutrino direction. The results can be seen in Figure 7.7. Clearly, using more training events results in a smaller opening angle. The scaling of the loss can be fit with a power law, similar to the scaling found in [124]. The data has been fitted by

$$\mathcal{L}(D) = a \cdot D^{-k}. \tag{7.1}$$

The best fit parameters were determined to be $a \approx 63.66$ and $k \approx 0.275$. It is a great result to see this transformer scaling for angular reconstruction with high energy IceCube data, motivating the production of large Monte Carlo sets.

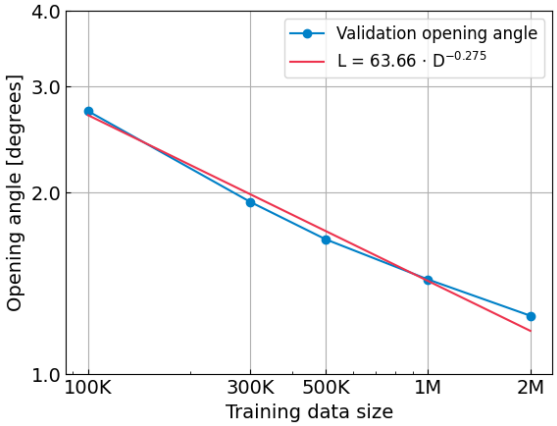


Figure 7.7: Performance of angular reconstruction for high energy IceCube events improves with increasing training set size. The performance follows a power law scaling, shown with a fit in red.

Still, there appears to be some slight flattening of the scaling at larger training sizes. This could be the result of the capacity limitation of the model at fixed size N [124]. In the infinite data regime where $D \to \infty$, one would expect the loss to converge to some model-size-limited loss $\mathcal{L}(N)$. It would be interesting to investigate this flattening further by training at bigger set sizes. Unfortunately, this wasn't possible since more training data for events with similar energy was unavailable for this analysis. Instead, the existing data points were fit on an ln-ln scale with an exponential decay plus a constant. Although the extrapolation of this fit to $D \to \infty$ should be taken with a grain of salt, it could give a rough estimate of the capacity limitation. The data points were fit by

$$\mathcal{L}'(D') = a \cdot e^{-b\,D'} + c, \tag{7.2}$$

with $\mathcal{L}' = \ln \mathcal{L}$ and $D' = \ln D - 11.5$, shifted for the fit. The results are shown in Figure 7.8, with best fit parameters $a = 1.58$, $b = 0.229$ and $c = -0.57$. Based on the fit value of c, a rough estimate of the opening angle of the transformer at this model size N in the infinite data limit is

$$\mathcal{L}(N) \approx e^{c} = e^{-0.57} = 0.57°.$$

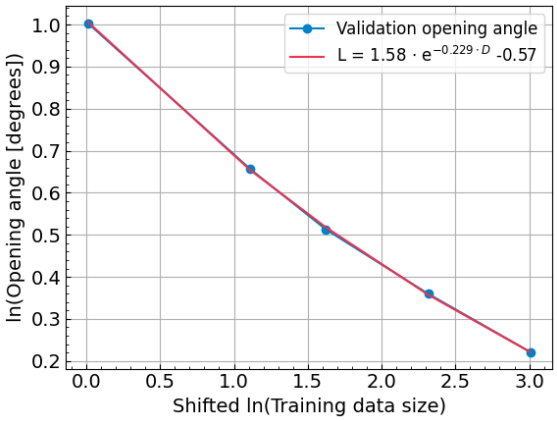


Figure 7.8: Performance of angular reconstruction for high energy IceCube events as a function of dataset size, scaled with a natural logarithm and shifted on the x-axis. The data points are fit with an exponential decay plus a constant to get a rough estimate of the model capacity limitation $\mathcal{L}(N)$.

7.2.2 Model size

Next to the training set size, it is worthwhile to investigate how the transformer model scales with the number of trainable parameters. Transformers in language processing have shown improved performance with increasing model size [124]; however, bigger models also have drawbacks. First of all, the training costs for large models can quickly grow, both in electricity costs and hardware required for the training, like CPUs, GPUs/TPUs and memory. Moreover, the training time increases, allowing for less iterative experimentation of hyperparameters during model development, slowing down innovation. Lastly, also the time for event inference will increase for bigger models, something that is very important for IceCube's high event rate. Although inference with large-scale machine learning models will likely still be faster than the traditional likelihood methods, it is important to keep in mind.

The model size is parameterized by the dimension of the input sequence d_{seq}, the feature dimension $d_{features}$, the number of encoder layers n_{layer}, the embedding dimension d_{emb}, the dimension of the intermediate feed forward layer d_{ff} and the output dimension d_{output}. The approximate number of trainable parameters and FLOPS per event per transformer operation are summarized in Table

7.3. The factor 2 in FLOPS for many of the operations is an approximation of the multiply-accumulate operation for matrix multiplication. The vast majority of the model parameters and FLOPS are part of the encoder block since these terms scale to second order (parameters) or third order (FLOPS) with d_{emb} or d_{seq}. This is emphasized by repeating the encoder for multiple layers. Ignoring the non-leading terms (assuming $d_{features} \ll d_{emb}$ and using $d_{ff} = 4\,d_{emb}$), an approximation for the total number of model parameters N and total compute of a forward pass C per event can be calculated for the encoder. The number of FLOPS during backpropagation is roughly equal to twice the FLOPS of a forward pass. Therefore, the total compute per training event is $\sim 3C$. Note that the number of model parameters and computations is independent of the number of heads h since resources are split over multiple small heads rather than duplicated.

Transformer model size

Operation	Parameters	FLOPS per event (forward)
Input Embedding	$d_{features}\,d_{emb}$	$2\,d_{seq}\,d_{features}\,d_{emb}$
Position Encoding	$d_{seq}\,d_{emb}$	$d_{seq}\,d_{emb}$
Attention: QKV	$n_{layer}\,d_{emb}\,3\,d_{emb}$	$2\,d_{seq}\,n_{layer}\,d_{emb}\,3\,d_{emb}$
Attention: Weights	–	$2\,d_{seq}\,n_{layer}\,d_{seq}\,d_{emb}$
Attention: Output	–	$2\,d_{seq}\,n_{layer}\,d_{seq}\,d_{emb}$
Attention: Projection	$n_{layer}\,d_{emb}^2$	$2\,d_{seq}\,n_{layer}\,d_{emb}^2$
Feed Forward	$n_{layer}\,2\,d_{emb}\,d_{ff}$	$2\,d_{seq}\,n_{layer}\,2\,d_{emb}\,d_{ff}$
LayerNorms	$4\,n_{layer}\,d_{emb}$	$2\,d_{seq}\,n_{layer}\,d_{emb}$
Aggregation	–	$d_{seq}\,d_{emb}$
Linear Regression	$d_{emb}\,d_{output}$	$2\,d_{emb}\,d_{output}$
Encoder total	$\boldsymbol{N \approx 12\,n_{layer}\,d_{emb}^2}$	$\boldsymbol{C \approx 2\,d_{seq}(N + 2\,n_{layer}\,d_{seq}\,d_{emb})}$

Table 7.3: Approximation of the number of model parameters and compute for different operations in the transformer model. Ignoring non-leading terms, an estimate of the total number of trainable parameters N and FLOPS per event on a forward pass C is given for the encoder.

To study the influence of the model size, nine transformer models were trained with a variable number of encoder layers $n_{layers} = \{4, 6, 8\}$ and embedding dimensions $d_{emb} = \{128, 256, 512\}$. A fixed training set size $D = 500$k and number of heads $h = 4$ was used for all models. The models were trained for a constant number of steps with equal hyperparameters (except the largest model was trained at a 20% reduced learning rate, required to prevent the model from diverging). Both for training and validation, events in the energy range $10\,\text{TeV} - 1\,\text{PeV}$ passing the Thorsten Cut were used. A complete overview of training details is given in Appendix A.

The model performance was evaluated on the median opening angle between the predicted and true neutrino direction. The results are shown in Figure 7.9. The model performance improves both with an increase in the number of layers and with an increase in the embedding dimension. The effect seems stronger with increasing embedding dimension; however, the number of trainable model parameters also scales squared in this direction. This means that at a fixed model parameter size, the transformer actually scales better with depth (an example: the model with $n_{layers} = 8$ and $d_{emb} = 128$ has a total of $1.6M$ trainable parameters and still performs better than the model with $n_{layers} = 4$ and $d_{emb} = 256$ at $3.2M$ parameters). This result contrasts the results on language processing by *Kaplan et al.*, who found a power law scaling of performance with model size, which was almost independent on model shape (ratio d_{emb}/n_{layer}) [124].

However, training deeper models is more time-consuming than training wider models. This is shown in Figure 7.10, showing each model's training time, normalized to the smallest model. It clearly shows the parallelizable nature of computation across the embedding dimension at the GPU: even though the number of model parameters grows a factor ~ 16 from $d_{emb} = 128$ to $d_{emb} = 512$, the training time is only a factor ~ 3 longer. The opposite is true for the number of layers: their sequential nature shows near-linear scaling of training time with model parameters.

Therefore, it can be concluded that to train well-performing models with a high number of trainable parameters effectively on the GPU, a balanced scaling of both depth and width is required: depth for memory-efficient performance per trainable parameter; width for time-efficiency to scale the model size without linear scaling in training time.

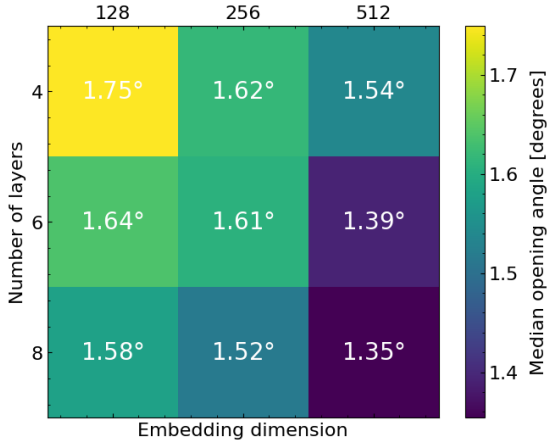


Figure 7.9: Scaling of the median opening angle between the predicted direction and true neutrino direction as a function of transformer model size.

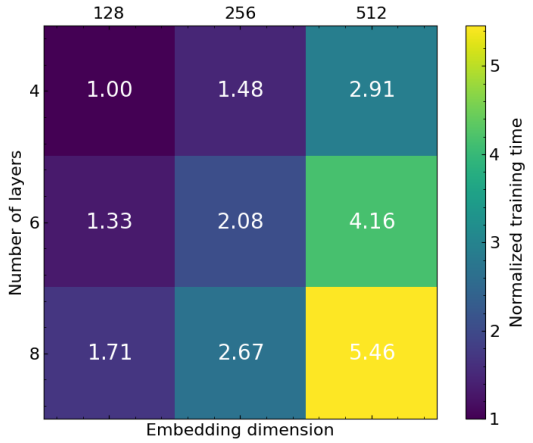


Figure 7.10: Scaling of the training time as a function of transformer model size. Training times are normalized to the time of the smallest model.

7.2.3 Number of heads

Lastly, a small study was performed to investigate the effect of the number of heads h in the multi-head attention. The developers of the original transformer found it beneficial to use multi-head attention for language processing tasks compared to single-head attention, claiming it allows the model to attend to information from different representation subspaces simultaneously [103]. However, they also concluded that quality drops again when using too many heads. *Kaplan et al.* found the influence of the number of heads to be less than 2% but also found an average number of heads to be optimal [124].

A comparison of three models was made with the number of heads $h = \{1, 4, 8\}$. The training set size was fixed at $D = 500\,000$, the model size was fixed with $d_{emb} = 512$ and $n_{layer} = 8$ and the batch size was fixed at $d_{batch} = 64$. All other hyperparameters were also kept constant and are detailed in Appendix A. The models were trained on events in the 10 TeV − 1 PeV energy range using the Thorsten Cut. The results for both the median opening angle on the validation set and training speed are presented in Table 7.4. Even though the comparison is limited, it seems like a medium number of heads also works best for IceCube data. Interestingly, even though the model size and number of computations are constant for all models, increasing the number of heads slows down the training process. Likely, this is due to the inefficiency of smaller matrix multiplications and more overhead due to increased reshaping and concatenation operations.

Scaling with transformer heads

Heads	Median opening angle	Speed (batches/s)
1	1.42°	5.7
4	1.35°	4.8
8	1.40°	3.8

Table 7.4: Influence of the number of transformer heads on the median opening angle between true and predicted neutrino direction and the training speed. A medium number of heads seems optimal for performance. Using more heads decreases training speed.

7.3 Pulse maps vs. PMT-fication

Transformers scale quadratically with the sequence length, both in terms of memory and computation, since every element attends to every other element. Therefore, the maximum sequence length feasible for training is limited. Using PMT-fication, the sequence length of the high energy IceCube events was significantly reduced by aggregating pulse information on a PMT level, as described in Section 6.1.3. This section studies the effect of this method on the transformer's capabilities of directional reconstruction of the neutrino.

Models were trained on the same set of events, once represented by a sequence of pulses and once by a sequence of PMTs. Both sequences were truncated at a maximum sequence length of 256, keeping the computational load fixed. Before truncation, the PMTs were sorted in descending order on their cumulative charge for the event. Pulses were sorted in descending order on their individual charge. All other model hyperparameters, like model size, batch size and number of training steps, were kept the same for both models.

7.3.1 Neutrino energy of 10 TeV - 1 PeV

For the first test, the transformers were trained on 1 million events in the intermediate available energy range $10\,\mathrm{TeV} - 1\,\mathrm{PeV}$. The Thorsten Cut was applied in both training and validation. Full details on the model hyperparameters are set out in Appendix A. Figure 7.11 shows the distribution of the opening angle between the predicted and true neutrino direction in the range $[0, 5]$ degrees. A dashed line indicates the median of the entire distribution. The reconstruction accuracy of the model trained on the PMT-fied events is significantly better than the one trained on the pulse maps, with medians at 1.41° and 2.61°, respectively.

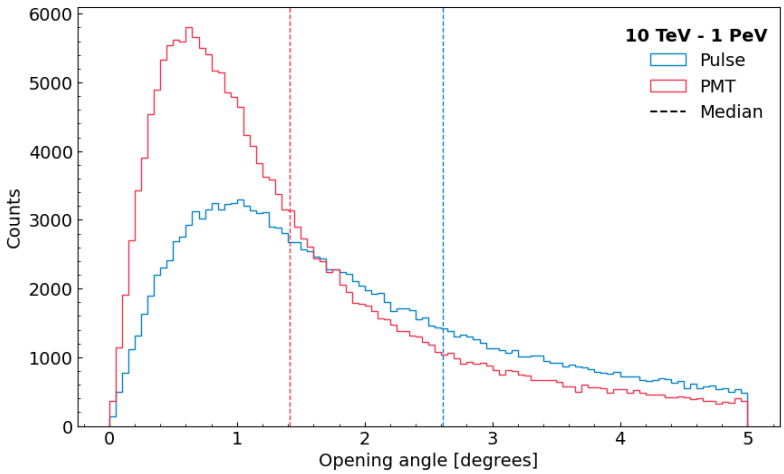


Figure 7.11: Distribution of the opening angle between the predicted and true neutrino direction for a transformer model trained on events represented as pulse maps versus PMTs. The distribution is truncated at 5°. The dashed lines indicate the median of the full distributions.

To investigate this difference in performance in more detail, the events in the validation set were binned by the number of pulses before truncation. For events with fewer than 256 pulses, no information should have been thrown away, and one would expect the pulse model to perform equal to or better than the PMT-fied model. The results are shown in Figure 7.12. The dashed lines indicate the boundaries of the bins, with the black line indicating the boundary at 256. Once again, statistical uncertainties on the median are determined by binomial intervals. At the bottom, the ratio of the PMT median to the pulse median is presented. The figure shows that

for events with fewer than 256 pulses, the models perform about equally, as expected. However, it is evident that the transformer trained on pulse maps is not performing well on events with a high number of pulses. Clearly, using only the 256 pulses of the highest charge does not represent these events well, and learning to properly reconstruct these tracks becomes impossible. On the contrary, the performance of the PMT-fied model on high-pulse events is doing very well. This is a strong indication that PMT-fication is an effective method to represent high energy IceCube events with a large number of pulses with the purpose of transformer training, at least in the 10 TeV − 1 PeV energy range.

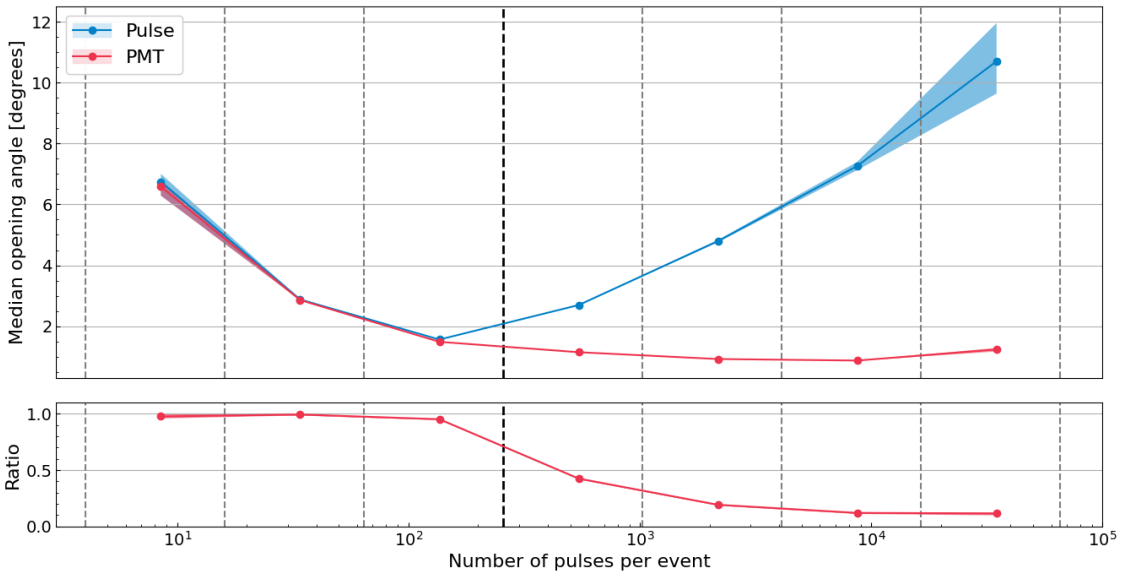


Figure 7.12: Median opening angle as a function of pulses per event for a transformer trained on pulse maps or PMT-fied data. The training data is in the energy range 10 TeV − 1 PeV. The models were truncated to a maximum sequence length of 256, indicated by the black dashed line. The bottom plot shows the ratio of the median of the PMT model to the pulse model.

To study if using a maximum of 256 PMTs still gives a proper representation of the brightest IceCube events, a similar figure is made with the median opening angle as a function of the number of PMTs before padding or truncation. The results are shown in Figure 7.13. The truncation boundary of 256 PMTs is indicated by the black dashed line. Unlike the pulse map model in Figure 7.12, the median opening angle of the PMT-fied model shows a smooth curve as a function of the number of PMTs, even when crossing the truncation boundary. This indicates that also the brightest IceCube events, triggering more than 256 PMTs, can be reconstructed well using just a selection of PMTs with the highest cumulative charge. Moreover, it can be concluded from Figure 7.13 that the PMT-fied transformer actually performs best on these bright events, reconstructing them with a median opening angle between true and predicted neutrino direction well below 1°.

A comparison of the PMT-fied and pulse map transformers as a function of true neutrino energy in the range 10 TeV − 1 PeV is presented in Figure 7.14. It reaffirms the conclusion of the other results: the model trained on PMT-fied events outperforms the model trained on pulse maps across the entire energy spectrum. While the median opening angle of the pulse model worsens for brighter/more energetic events, the PMT-fied transformer improves in performance with energy. Based on these results, PMT-fication seems to be an effective method to represent high energy IceCube events for transformer training.

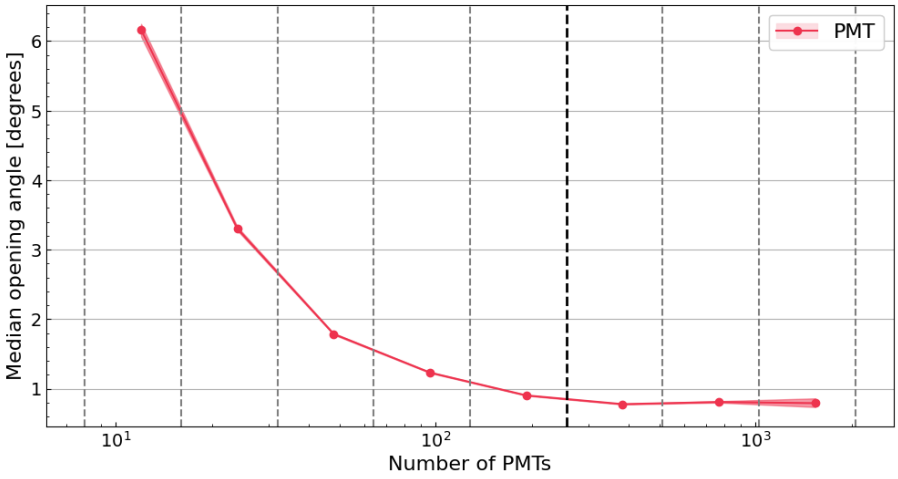


Figure 7.13: Median opening angle of the PMT-fied transformer model as a function of PMTs per event before padding or truncation. The training data is in the energy range 10 TeV − 1 PeV. The model was truncated to a maximum sequence length of 256 PMTs during training, indicated by the black dashed line.

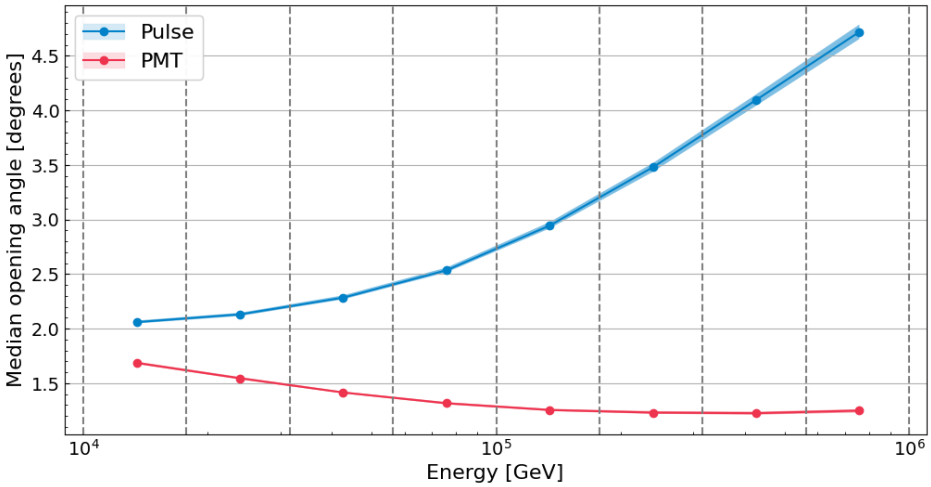


Figure 7.14: Median opening angle of the PMT-fied and pulse map transformer models as a function of true neutrino energy. The training and validation data are in the energy range 10 TeV − 1 PeV.

7.3.2 Neutrino energy of 100 GeV - 10 TeV

A similar analysis was performed for training and validation on the 22010 dataset in the energy range of 100 GeV − 10 TeV. A second time, transformer models with equal hyperparameters were trained on 1 million training events, once represented as pulse maps and once as PMT-fied events. Full training hyperparameters are given in Appendix A. The distribution of the opening angle in the range $[0, 15]$ degrees is shown in Figure 7.15. Interestingly, although the mode of the PMT-fied distribution seems slightly smaller than for the pulse transformer, the median opening angle of the model trained on pulse maps is smaller. This is not completely surprising, since in the 100 GeV − 10 TeV energy range, about 97.5% of the events have less than 256 pulses (see Figure 6.3). Therefore, the vast majority of the pulse maps in training were not truncated, and the PMT-fication might have been an unnecessary reduction, harming overall performance.

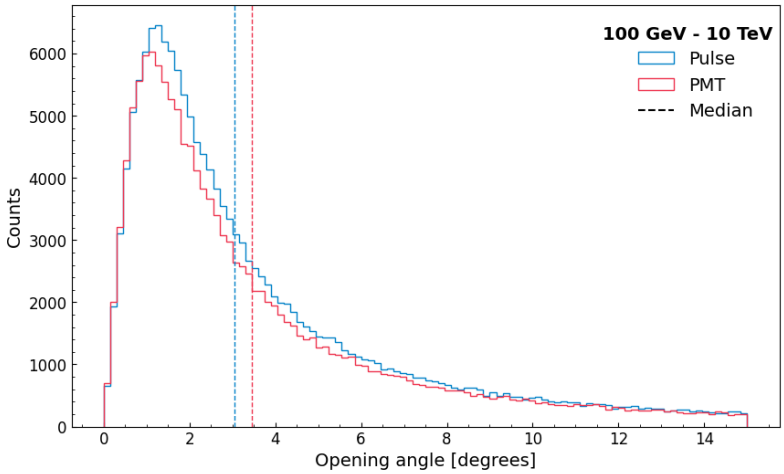


Figure 7.15: Distribution of the opening angle between the predicted and true neutrino direction for a transformer model trained on events represented as pulse maps versus PMTs. The distribution is truncated at 15°. The dashed lines indicate the median of the full distributions.

Figure 7.16 presents the median opening angle as a function of the number of pulses before padding or truncating for the energy range 100 GeV − 10 TeV. A similar increase in the median opening angle for the transformer trained on pulse maps for events with more than 256 pulses as in Figure 7.12 can be seen. Also for this dataset, the transformer was not able to reconstruct truncated pulse maps well. The PMT-fied model performs better on these events. However, for events with less than 256 pulses, the pulse model outperforms the PMT-fied model by approximately 17%. This contrasts with the results seen in the 10 TeV − 1 PeV energy range, where both models performed similarly on these low-pulse events. One possible explanation is that the `22011` dataset included a higher proportion of bright events, allowing the PMT-fied model to learn more effectively than the pulse model. This advantage may have compensated for the weaker performance of the PMT-fied model on low-pulse events compared to the pulse model, where events are unnecessarily reduced in information. Since the fraction of events with more than 256 pulses is so small for the `22010` dataset, this compensating effect is lost. As a result, the transformer trained on pulse maps outperforms the PMT-fied model for low-pulse events for this dataset.

The median opening angle as a function of true neutrino energy in the 100 GeV − 10 TeV range is shown in Figure 7.17. Both models show similar development of the median opening angle with energy. The transformer trained on pulse maps outperforms the model trained on PMT-fied events across the entire energy spectrum. The ratio plot shows that at lower energies, this difference is approximately 14%, while at higher energies, the models approach each other within 10%. Based on these results, it is hard to conclude whether PMT-fication is an effective method in this energy range. If one is only interested in events between 100 GeV − 10 TeV, it is probably better to train the transformer model on the pulse maps. The distribution of sequence lengths for these events is not problematic for the transformer, and aggregating pulses will likely result in a slight loss of performance. However, if one would like to train a combined model spanning the 100 GeV − 100 PeV range, the transformer trained on pulse maps is expected to quickly lose accuracy. Here, especially for the reconstruction of the higher energy events, it will be beneficial to use PMT-fication. Possibly, including many of these high-pulse events can even compensate for the unnecessary data reduction at low-pulse events, equalizing the performance of the PMT-fied model to the pulse model.

Preferably, the comparison between pulse maps and PMT-fied data would have been extended to the `22012` set in the 1 PeV − 100 PeV energy range. Unfortunately, these events are so large that it would have required the development of a new pulse map dataloader before training at a reasonable speed would have been possible. Since this is a very time-intensive task, it was decided

not to do this. It is expected that the results of the 22011 set extend to the ultra high energy events, and PMT-fication is very likely a more effective reduction method in this energy range than truncating the pulse map. Therefore, for the final analysis, a transformer model is trained on PMT-fied events spanning the entire available energy range, from 100 GeV − 100 PeV.

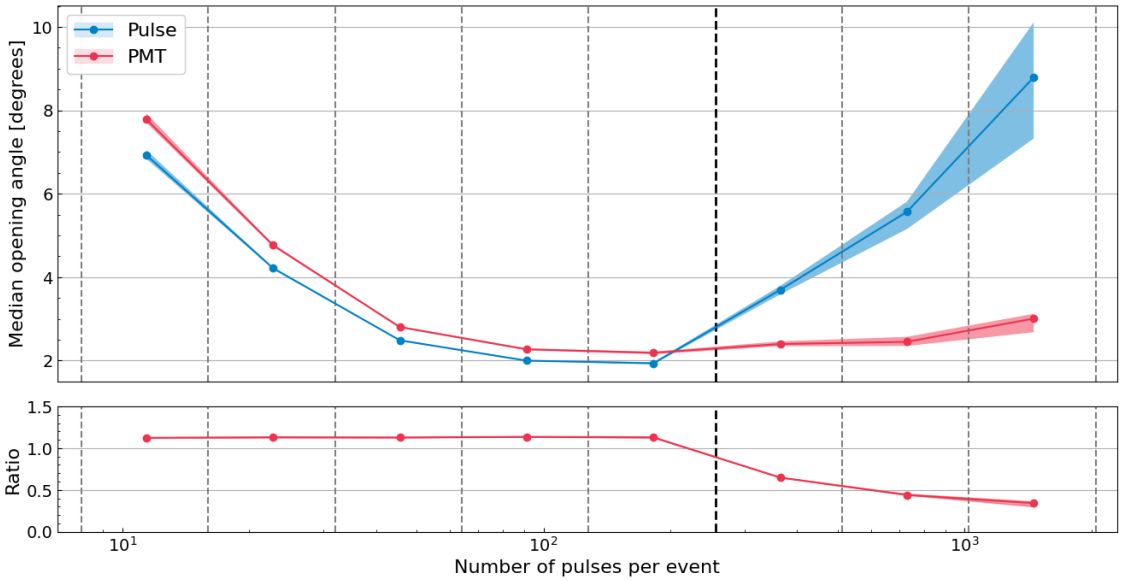


Figure 7.16: Median opening angle as a function of pulses per event for a transformer trained on pulse maps or PMT-fied data. The training data is in the energy range 100 GeV − 10 TeV. The models were truncated to a maximum sequence length of 256 during training, indicated by the black dashed line. The bottom plot shows the ratio of the median of the PMT model to the pulse model.

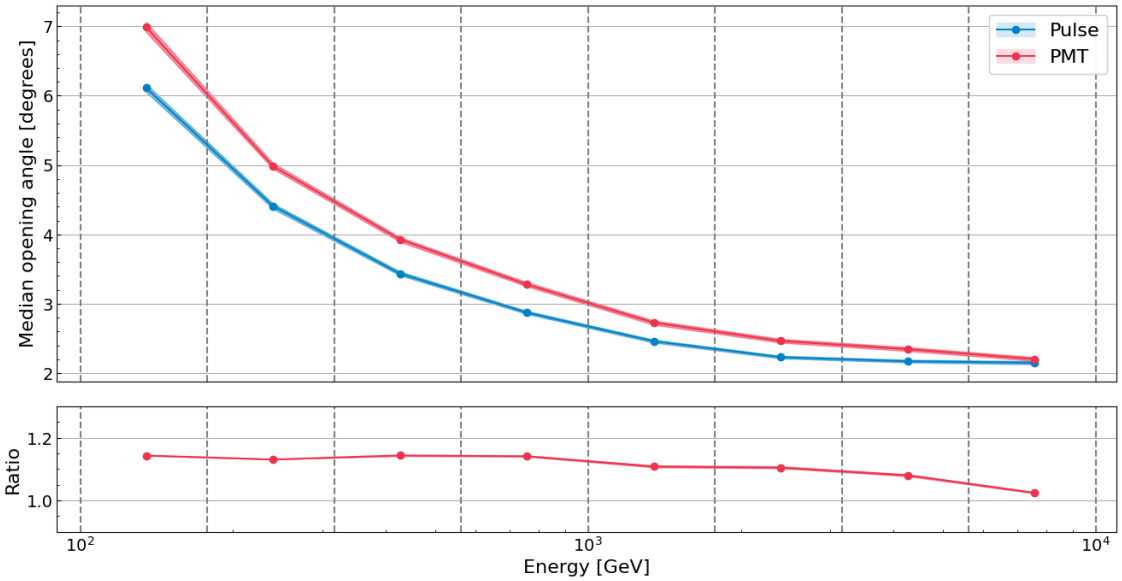


Figure 7.17: Median opening angle of the PMT-fied and pulse map transformer models as a function of true neutrino energy. The training and validation data are in the energy range 100 GeV − 10 TeV. The bottom plot shows the ratio of the median of the PMT model to the pulse model.

7.4 Directional reconstruction

All findings in the previous sections served as the motivation for training a final, large transformer model, tasked with the angular reconstruction of high energy neutrino tracks. This model combines the 22010, 22011 and 22012 datasets, spanning the entire available energy range from 100 GeV − 100 PeV. To maximize performance on the TeV and PeV events, the model was trained on almost all available events with energies > 10 TeV (still keeping aside sufficient events for validation and testing), combined with an approximately equal amount of events from the 100 GeV − 10 TeV range. After cleaning and applying the Thorsten cut event selection, this totals to roughly 3.9M muon (anti)neutrino track events used for training of this final model, as described in Table 7.5. All events were PMT-fied and truncated at a maximum sequence length of 256 PMTs. An embedding dimension of $d_{emb} = 512$ was used, with 4 parallel multi-heads and 6 encoder layers, resulting in a total of 27.0M trainable parameters. The transformer was trained for 100 epochs at a batch size of 64 events. Details on all training hyperparameters can be found in Appendix A. Training this model took approximately three weeks.

Training events final model

Dataset	Number of events
22010	1 812 866
22011	1 700 713
22012	430 654
Total	**3 944 233**

Table 7.5: Number of muon (anti)neutrino training events used for each of the datasets for the final transformer model.

On the same hardware as used for training, the trained transformer model can predict the direction of approximately 1000 high energy neutrinos per second. This section will present the angular resolution of these reconstructions and compare them to the performance of the SplineMPE likelihood reconstruction algorithm applied to the same events. For this, reconstructed values of SplineMPE at level 3 data are used. All results in this section will show the performance on the testing set, which was kept completely blinded until the final model was finished training. The opening angle between the true neutrino direction and the predicted neutrino direction for both the transformer model and SplineMPE is analyzed for different subsets of the testing set, including starting and through-going tracks. Moreover, the influence of neutrino energy, detector-contained track length, and true zenith angle on the quality of the reconstructions is studied.

7.4.1 Level 3 muon (anti)neutrinos

Figures 7.18, 7.19 and 7.20 show the full distribution of the opening angle for all level 3 events passing the Thorsten cut in the datasets 22010, 22011 and 22012 respectively. The smaller zoomed-in plots show the same distributions, truncated at smaller opening angles. The dashed lines indicate the medians of the distributions. Moreover, the mean and the 25%, 50%, and 75% percentiles for both the transformer and SplineMPE are described in Table 7.6. For each of the datasets, the distribution in opening angle shows a sharper peak at small opening angles for SplineMPE than for the transformer. Unfortunately, this indicates that the likelihood reconstruction method is still able to reconstruct the 'best' events more accurately. However, it is an interesting result to see the tails of the opening angle distributions being orders of magnitude smaller for the transformer model. It suggests that the transformer might be able to predict the direction of 'hard-to-reconstruct' neutrinos better than SplineMPE, especially for the highest end of the energy spectrum. This idea is reinforced by the mean opening angle and 75% quantile of the 22011 and 22012 datasets, which are much smaller for the transformer.

Opening angle on level 3 data

Model	Dataset ID	Mean	25%	50%	75%
Transformer	22010	16.5°	1.17°	2.67°	9.66°
Transformer	22011	3.61°	0.58°	1.01°	2.20°
Transformer	22012	3.74°	0.56°	1.00°	2.27°
SplineMPE lvl3	22010	11.6°	0.88°	2.21°	6.73°
SplineMPE lvl3	22011	13.2°	0.28°	0.75°	3.52°
SplineMPE lvl3	22012	15.1°	0.20°	0.67°	5.50°

Table 7.6: Summarization of the opening angle between true and reconstructed neutrino direction for the transformer and SplineMPE. Results are shown for the test set events that pass both the Thorsten cut and level 3 selection criteria.

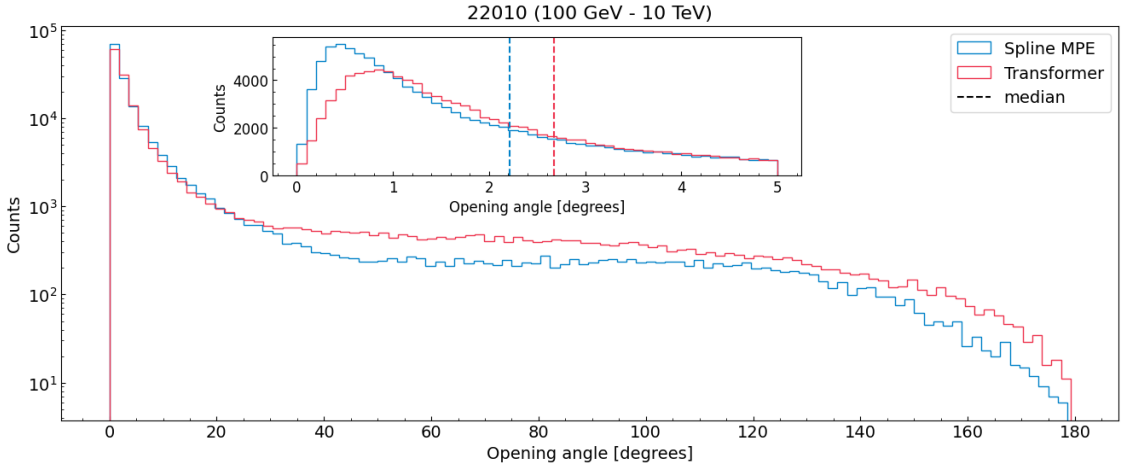


Figure 7.18: Distributions of the opening angle between true and predicted neutrino direction for the transformer and SplineMPE, shown on the test set of the 22010 dataset.

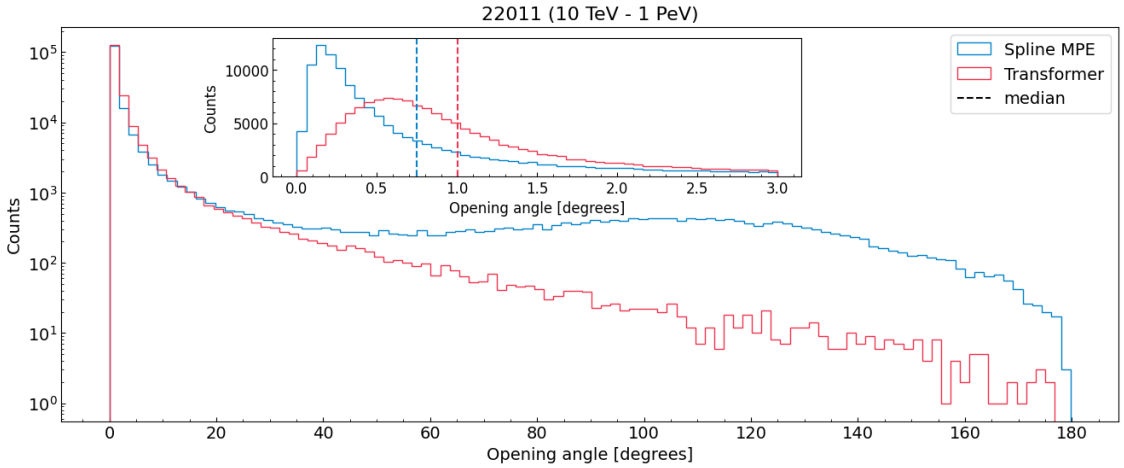


Figure 7.19: Distributions of the opening angle between true and predicted neutrino direction for the transformer and SplineMPE, shown on the test set of the 22011 dataset.

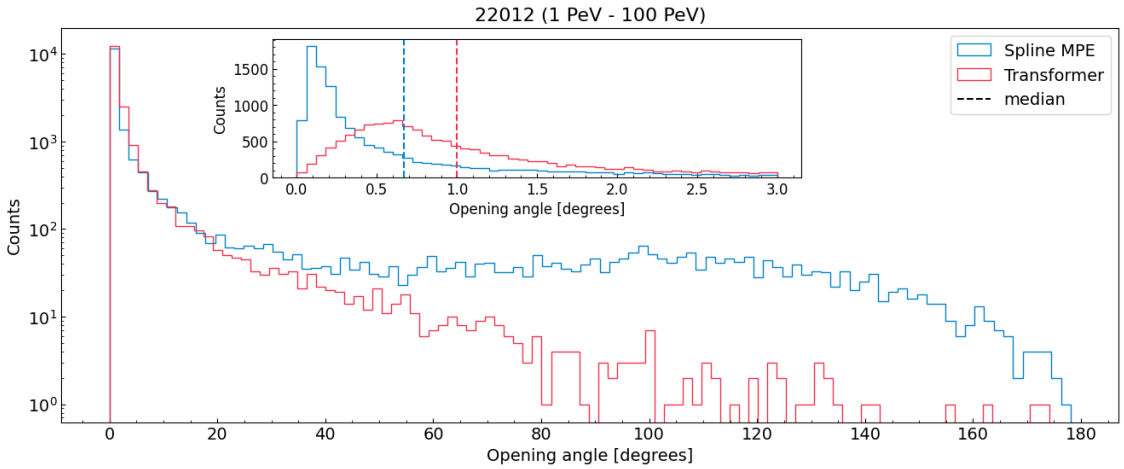


Figure 7.20: Distributions of the opening angle between true and predicted neutrino direction for the transformer and SplineMPE, shown on the test set of the 22012 dataset.

All three datasets are combined in Figure 7.21, showing the opening angle as a function of energy across the entire energy range. Except for the 75% percentile, SplineMPE outperforms the transformer across the entire energy range. The plot at the bottom of the figure shows the ratio of the median opening angle of the transformer to the median opening angle of SplineMPE, with the baseline ratio of 1 in blue. The quality of reconstructions of the transformer is closest to that of SplineMPE at the lower energies, but overall, both algorithms show a similar trend as a function of energy. In the next sections, the performance of both algorithms will be studied in more detail for specific types of events.

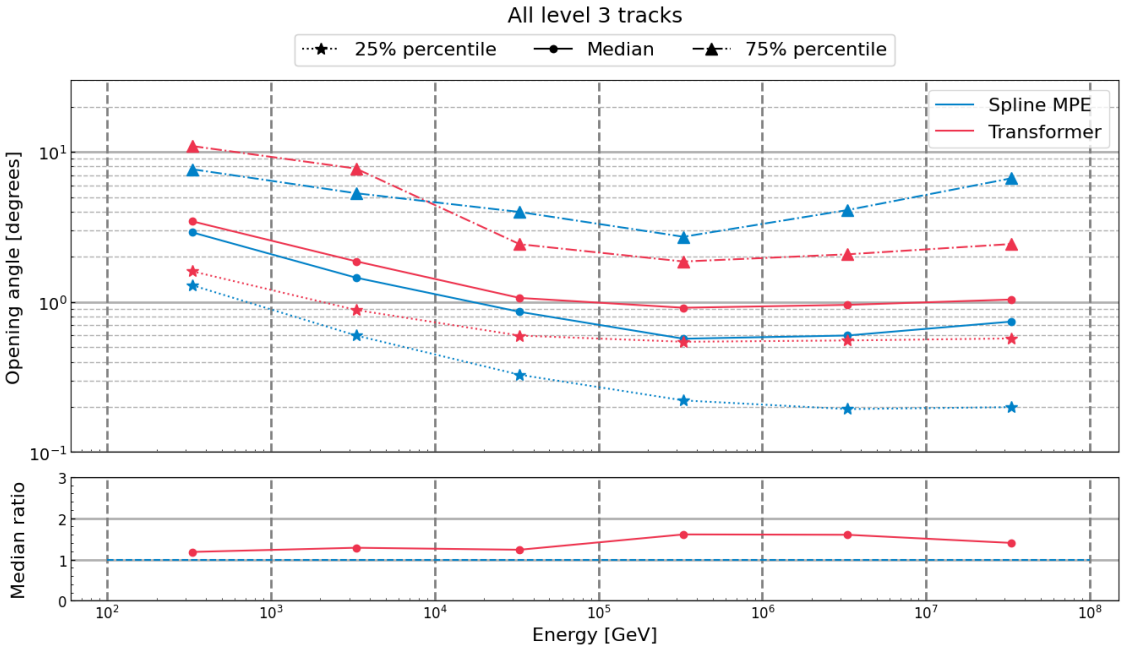


Figure 7.21: Opening angle between the true and reconstructed neutrino direction for SplineMPE and the transformer, as a function of true neutrino energy for all level 3 tracks passing the Thorsten cut. The top plot shows the 25%, 50%, and 75% percentiles in opening angle per bin. The bottom plot shows the ratio of the median opening angle of the transformer to SplineMPE.

7.4.2 Starting and through-going tracks

As discussed in Section 3.3, the analysis of astrophysical neutrinos in IceCube mainly considers two datasets, using different methods to reject the atmospheric background: high energy starting events (HESE) and tracks of neutrinos that have traversed the Earth (northern tracks). Therefore, when considering the angular reconstruction of high energy neutrinos, it makes sense to divide the analysis into two categories: starting tracks, muon (anti)neutrinos with their interaction vertex inside the detector, and through-going tracks, muon (anti)neutrinos that interact outside the detector, but where the muon travels through the detector. This way, one can get an idea of the performance on each of these IceCube analysis sets separately. In general, through-going tracks can be reconstructed with a better angular resolution than starting tracks. For the analysis in this section, the Monte Carlo truth information of the interaction vertex and muon direction is used to split the test set into starting tracks, through-going tracks, and tracks that miss the detector. Since these truth values were used, no veto region was required for the starting tracks.

Through-going tracks

The median opening angle of the through-going tracks as a function of energy for both the transformer and SplineMPE is shown in Figure 7.22. The ratio plot once again shows the ratio of the transformer median to the median of SplineMPE. An additional quality cut was used to select the testing events, requiring the through-going tracks to have a minimum track length contained inside the IceCube detector volume of 400 m. This removes 'through-going' events that only clip an edge of the detector, which are not the clear muon tracks aimed for with this analysis. A more elaborate discussion on the effects of detector-contained track length will be given in Section 7.4.3. Figure 7.22 unfortunately shows that SplineMPE is outperforming the transformer for the angular reconstruction of through-going tracks across the entire energy spectrum; with the median opening angle of the transformer being $\sim 0.3 - 0.5°$ away from SplineMPE for all energies. This is not completely surprising, since especially these events are known to be reconstructed very well by SplineMPE. Still, the transformer is able to reconstruct the highest energy through-going tracks with a median opening angle of only 0.61°.

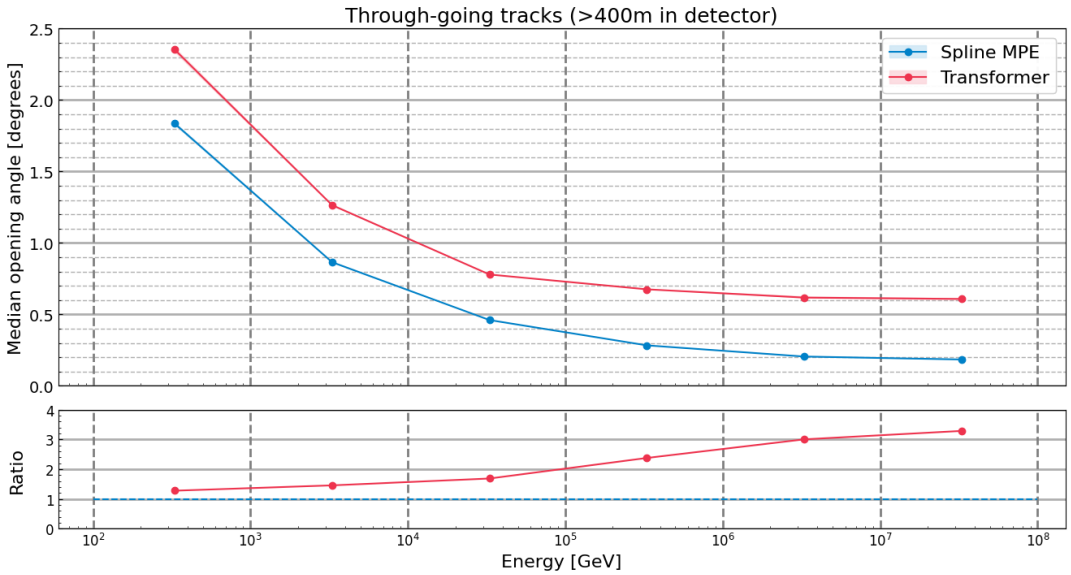


Figure 7.22: Median opening angle between the true and predicted neutrino direction for the transformer and SplineMPE for through-going tracks with a minimum detector-contained track length of 400 m. The shaded area indicates the 1σ confidence on the median.

Starting tracks

The median opening angle as a function of energy for starting tracks is shown in Figure 7.23. The 400 m detector-contained track length cut is applied again; removing edge-clipping events, but also starting events that interact on their way out of the detector. Up to neutrino energies of ~ 100 TeV, the transformer achieves a similar angular resolution as SplineMPE. This is a great result, as these events traditionally are more difficult to reconstruct, but the transformer seems to generalize well. At the higher energies, SplineMPE is still able to reconstruct the starting events better than the transformer, and the median opening angle of the transformer is approximately a factor of 2 larger than that of SplineMPE. The median opening angle of the transformer for the highest energy bin reaches 0.66°, which is not much off from the resolution of the through-going tracks. Overall, it can be concluded that the performance gap between the angular reconstruction of through-going tracks and starting tracks is smaller for the transformer than for SplineMPE. This is a hopeful sign, as it could indicate that the transformer is able to generalize better to a wider quality range of events.

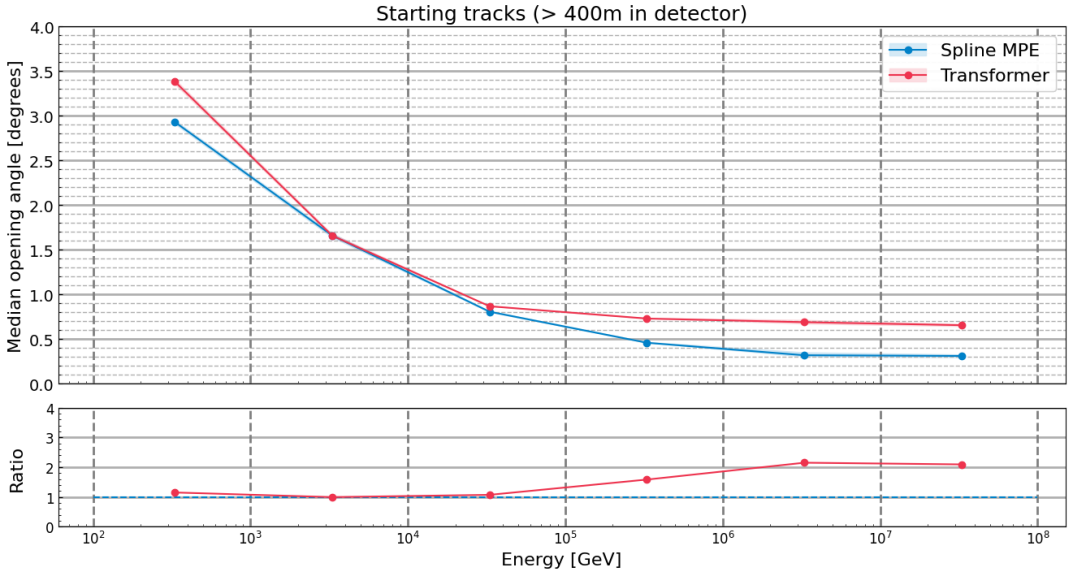


Figure 7.23: Median opening angle between the true and predicted neutrino direction for the transformer and SplineMPE for starting tracks. All test events shown have a minimum detector-contained track length of 400 m. The shaded area indicates the 1σ confidence on the median.

7.4.3 Track length

A useful quality parameter for high energy neutrino tracks in IceCube is the length of the track contained within the detector volume. Especially in the PeV energy range, events can be so bright that even when they miss the detector volume or only clip an edge, they can still produce a lot of detector signal. However, the detector signature of these tracks is completely different from when the muon travels a large distance in the detector, and in general, it is harder to reconstruct their event geometry. The effect of the detector-contained track length on the angular reconstruction of both starting and through-going tracks has been studied. The median opening angle for the transformer and SplineMPE is shown as a function of track length in Figure 7.24. Tracks that do not travel through the detector volume are excluded from this plot. As expected, the figure shows a decrease in the median opening angle with detector contained track length for both starting and through-going tracks. Interestingly, the transformer is outperforming SplineMPE for events with a small track length. This finding is most pronounced for the starting tracks, where the transformer

model is able to improve the neutrino reconstruction compared to SplineMPE up to a detector contained track length of $\sim 700\,\mathrm{m}$. This result is of significant importance, as most starting tracks have a short track length (71% of starting tracks have a detector contained track length of less than 700 m). An improved angular resolution of starting events is highly impactful for the all-sky neutrino source search with the HESE sample [77].

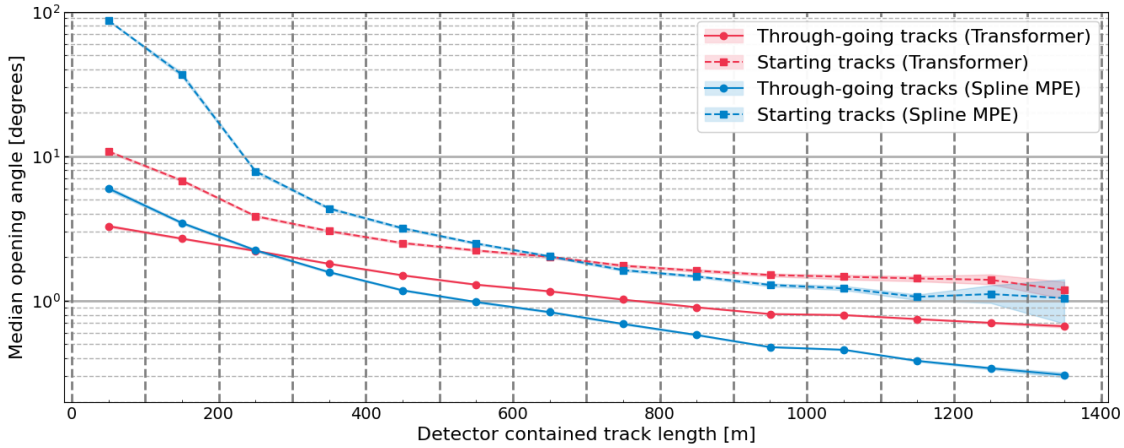


Figure 7.24: Median opening angle between the predicted and true neutrino direction of the transformer and SplineMPE as a function of detector contained track length. Testing events are split into through-going and starting tracks. The events for which the muon did not travel through the detector volume have been excluded. The shaded area indicates the 1σ confidence interval on the value of the median.

To further analyze the domain of muon (anti)neutrino events where the transformer model outperforms the conventional likelihood approaches in reconstruction, the ratio of the median opening angle of the transformer to SplineMPE is plotted as a function of both energy and track length in a 2D heatmap. The results for through-going tracks are shown in Figure 7.25, and the results for the starting tracks are shown in Figure 7.26. The small number in parentheses below the ratio represents the number of testing events in this bin, which gives an indication of the statistical significance of the ratio in that bin. Normalized colors highlight regions where the transformer outperforms SplineMPE in red, while the opposite is indicated in blue. For the through-going tracks, the transformer only outperforms SplineMPE for the smallest track lengths, especially in the TeV energy range. The precise reason why the transformer does well in this energy range is unclear. A possible explanation could be the over-representation of events of this energy for edge-clipping tracks in the training set. For the rest of the through-going tracks, the opening angle ratio grows with both the detector-contained track length and the energy. The longest and most energetic through-going tracks are reconstructed with a median opening angle of $0.52°$ by the transformer, while SplineMPE reconstructs these events with a median opening angle of $0.10°$.

For the starting tracks, the transformer has been able to improve the angular reconstruction compared to SplineMPE for a larger domain of events. For starting tracks with a detector-contained track length of less than 200 m, the relative improvement of the transformer grows with increasing energy. While the reconstruction of SplineMPE for events of this type at neutrino energies of more than 100 TeV basically becomes random guessing (with median opening angles of $\sim 90°$), the transformer is still able to reconstruct their direction with a median opening angle of just $2.3 - 3.2°$. For starting tracks with a track length between $200 - 600\,\mathrm{m}$, the highest relative improvement of the transformer once again shifts to the TeV energies. As the starting tracks travel longer distances within the detector, their signatures increasingly resemble those of through-going tracks, at which point the transformer no longer shows improved performance relative to SplineMPE.

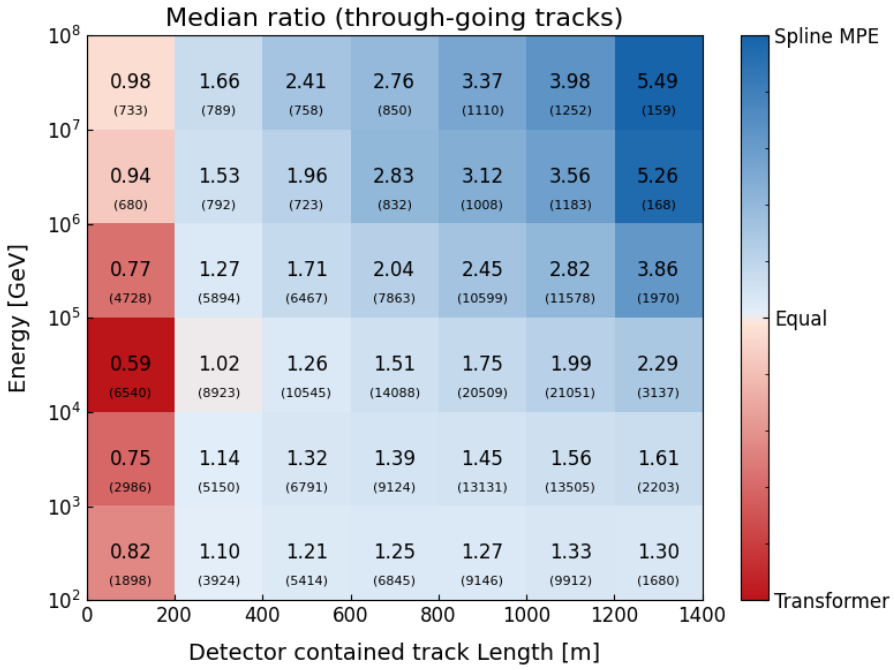


Figure 7.25: Ratio of the median opening angle between predicted and true neutrino direction for the transformer to SplineMPE for through-going tracks, as a function of both energy and detector-contained track length. The small number in parentheses below the ratio shows the number of testing events in each bin. Normalized color scales highlight regions where the transformer outperforms SplineMPE in red, while the opposite is indicated in blue.

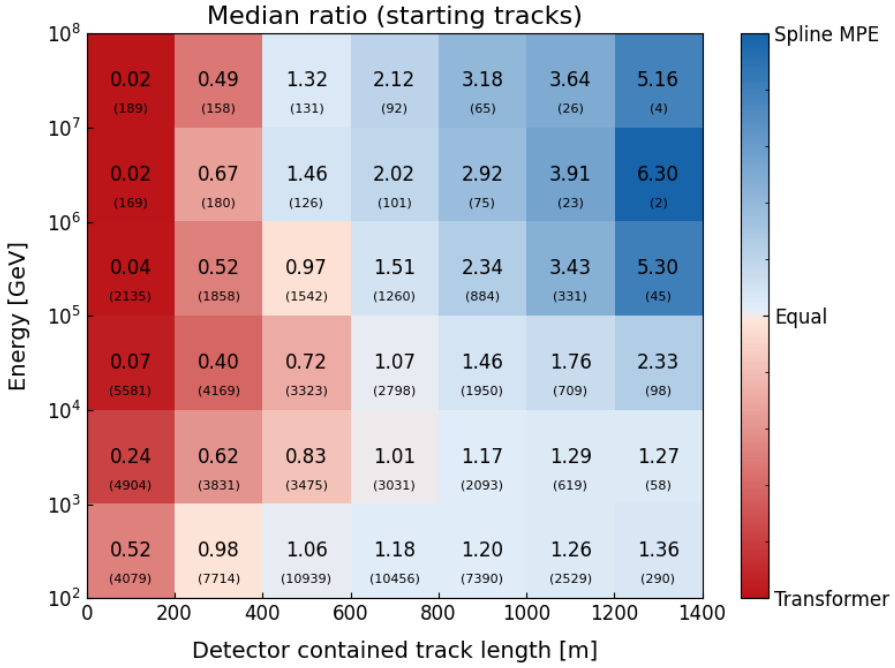


Figure 7.26: Ratio of the median opening angle between predicted and true neutrino direction for the transformer to SplineMPE for starting tracks, as a function of both energy and detector-contained track length. The small number in parentheses below the ratio shows the number of testing events in each bin. Normalized color scales highlight regions where the transformer outperforms SplineMPE in red, while the opposite is indicated in blue.

7.4.4 Up- and downgoing events

Since IceCube detects so many background events from above due to cosmic ray air showers, separating events into up- and downgoing in the detector can be useful. The angular resolutions of traditional muon track reconstruction algorithms are known to depend on the true zenith direction. This is an effect of the background, but also of the downward-facing PMTs. To see if the transformer shows similar dependencies, the opening angle as a function of zenith was studied for each of the three datasets. The results are shown in Figures 7.27, 7.28 and 7.29 for the 22010, 22011 and 22012 datasets, respectively. The gray histogram in the background shows the shape of the underlying truth distribution in zenith, since this strongly depends on energy (very few downgoing neutrinos make it past the MuonFilter and level 3 selection cut at lower energies, while at the highest energies less neutrinos make it through the Earth, resulting in fewer upgoing events).

A strong dependency of the angular resolution as a function of true zenith can be seen in Figure 7.27. For neutrinos with $0° < \theta < 60°$ in the 100 GeV − 10 TeV energy range, the median opening angle of SplineMPE is more than 80°. The reconstruction of these downgoing events also presents difficulties for the transformer; however, it can still reconstruct them with a median opening angle of less than 20°. By sampling more downgoing training events in this energy range, this resolution could probably be improved further in future trainings. In Figures 7.28 and 7.29, it can be seen that the median opening angle of SplineMPE is larger for the downgoing events at higher energies as well. For all datasets, the angular resolution of the transformer is better than SplineMPE's for events coming from the southern sky with $\theta < 40°$ (although for the 22012 dataset the difference is marginal). Overall, for higher values of true zenith, the median opening angle of the transformer and SplineMPE scale similarly. Moreover, it can be concluded that both reconstruction algorithms perform best for horizontal events.

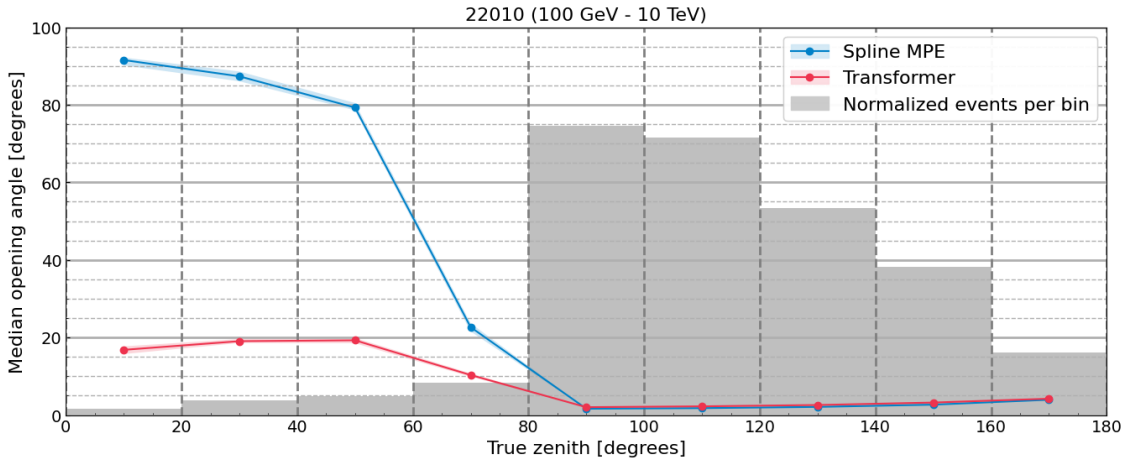


Figure 7.27: Median opening angle between true and predicted neutrino angle as a function of true zenith angle for the 22010 dataset. The results are shown for SplineMPE (blue) and the transformer (red), with shaded areas indicating the 1σ confidence bounds on the median. The gray histogram shows the shape of a normalized distribution of the true zenith angle, scaled to the y-axis for visualization.

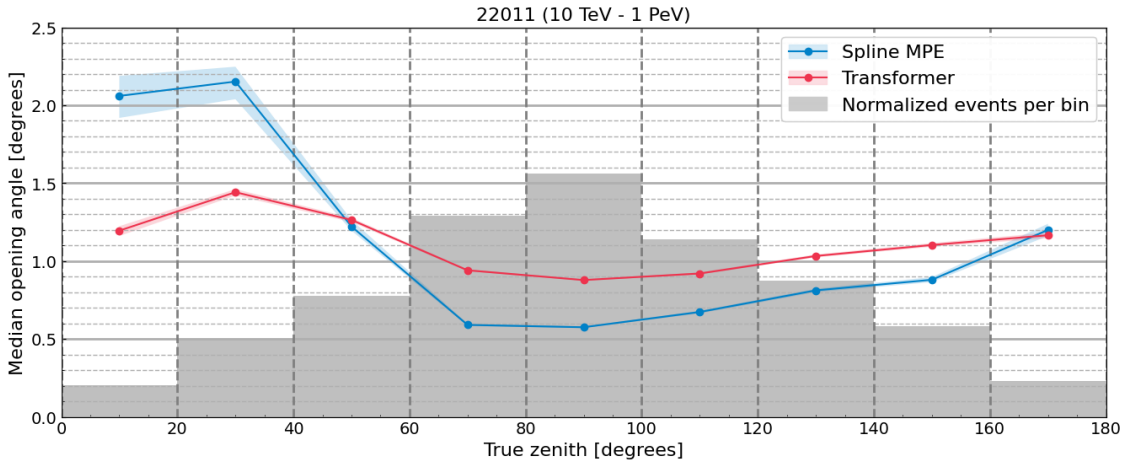


Figure 7.28: Median opening angle between true and predicted neutrino angle as a function of true zenith angle for the 22011 dataset. The results are shown for SplineMPE (blue) and the transformer (red), with shaded areas indicating the 1σ confidence bounds on the median. The gray histogram shows the shape of a normalized distribution of the true zenith angle, scaled to the y-axis for visualization.

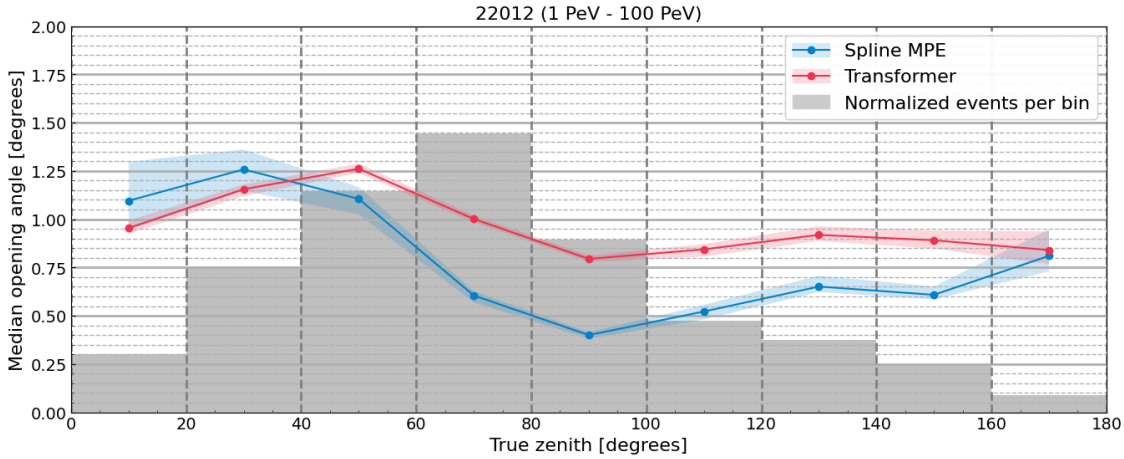


Figure 7.29: Median opening angle between true and predicted neutrino angle as a function of true zenith angle for the 22012 dataset. The results are shown for SplineMPE (blue) and the transformer (red), with shaded areas indicating the 1σ confidence bounds on the median. The gray histogram shows the shape of a normalized distribution of the true zenith angle, scaled to the y-axis for visualization.

7.4.5 Systematic checks

Placing these final results in context requires the consideration of potential biases in the angular reconstruction of the transformer model. While the presence of such a bias does not inherently pose a problem, its identification is important to accurately interpret the results. To study this, various systematic checks on the reconstructions have been performed, which will be presented in this section. First, the distributions of true and reconstructed zenith and azimuthal angles were compared. The results for all test events passing the Thorsten cut are shown in Figures 7.30 and 7.31 respectively. In general, the shape of the truth and transformer distributions align well. Interestingly, the distribution of azimuthal angle of the transformer in Figure 7.31 shows six

periodic peaks, which are not present in the underlying truth distribution. Further exploration of these peaks reveals that these angles match the DOM alignment in the IceCube detector, as shown in Figure 7.32. Apparently, the transformer model exhibits a slight detector geometry bias, predicting the azimuth direction of the neutrinos along the direction of the DOMs. This is not surprising, as the hexagonal grid structure results in denser hit patterns along these six principal directions, which likely resulted in the transformer more consistently learning the patterns along these axes. This bias is not detrimental, as it can easily be removed from the model by applying random rotations in azimuth to events during training.

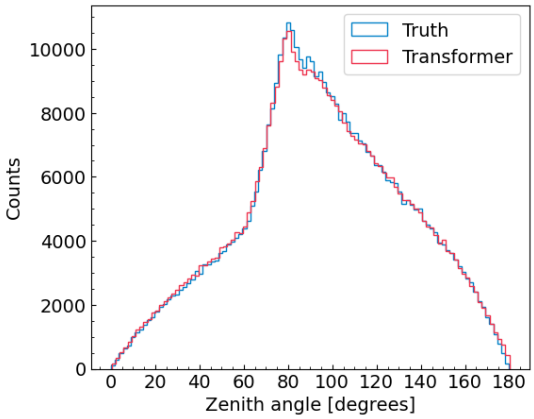


Figure 7.30: Distributions of the true and predicted zenith angle for the test set (all datasets combined).

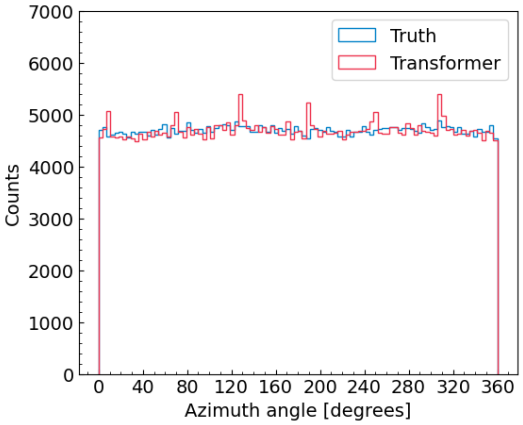


Figure 7.31: Distributions of the true and predicted azimuth angle for the test set (all datasets combined).

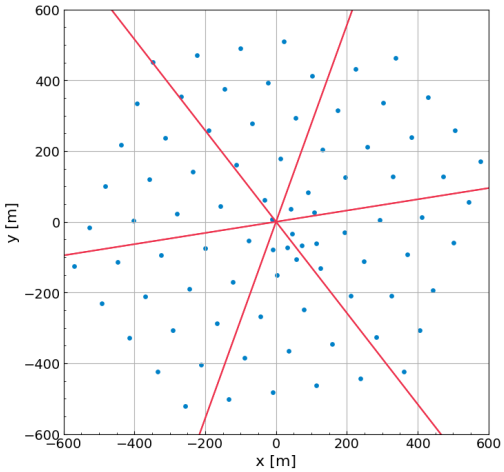


Figure 7.32: Graphical representation of the direction of the six peaks in azimuthal direction of Figure 7.31, with respect to the IceCube string locations. Blue dots indicate the location of the strings, while the red lines (drawn through the origin) show the azimuthal directions at the bin centers of the first six modes.

Next, the errors in the zenith and azimuth angles were studied separately to check for systematic under- or over-prediction in these different directions by the transformer. The error is defined as

$$\psi_{error} = \psi_{pred} - \psi_{true}. \tag{7.3}$$

Figures 7.33 and 7.34 show the distribution of the error for the zenith and azimuth angle respectively, both truncated between $[-3°, 3°]$. For an unbiased model, one would expect the distributions

to be centered around 0 (the black dashed line). This is the case for the azimuthal error, however, the error in zenith shows a clear shift, indicating a bias in the reconstructions of the transformer towards smaller values of θ. Depending on the energy of the dataset, this shift is approximately $-0.15°$ to $-0.4°$, and seems strongest at the higher energies. Moreover, this bias is not present in the reconstructions of SplineMPE (see Figures D.1 and D.2 in the appendix).

One might expect the bias to be related to the underlying zenith distribution, where an over-representation of downgoing events could cause upgoing events to be slightly misreconstructed. In an effort to explain the transformer's bias for zenith reconstruction, the median zenith error as a function of true zenith for the 22011 and 22012 datasets is shown in Figure 7.35 (as Figure 7.27 showed, the opening angle of the 22010 dataset strongly depends on true zenith. Because of this, the 22010 dataset has been excluded from Figure 7.35 for clarity. A version including this dataset can be found in the appendix, Figure D.3). Figure 7.35 shows however that the over-representation of downgoing events is not the cause of the bias. Rather, it seems like the bias is strongest for downgoing events with $20° < \theta_{true} < 80°$. Unfortunately, a definitive explanation for this bias remains unclear.

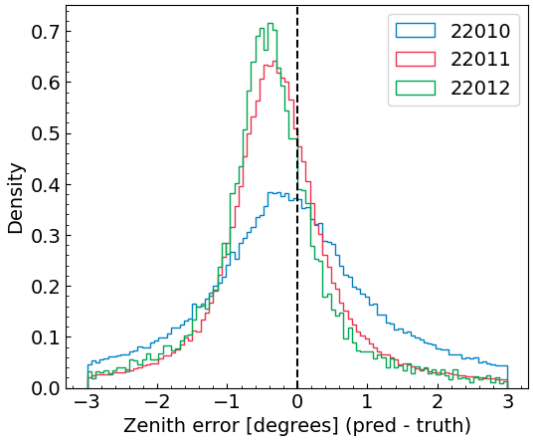


Figure 7.33: Distribution of the error between true and predicted zenith angle by the transformer for the 22010, 22011 and 22012 datasets. The dashed line indicates an error of 0°.

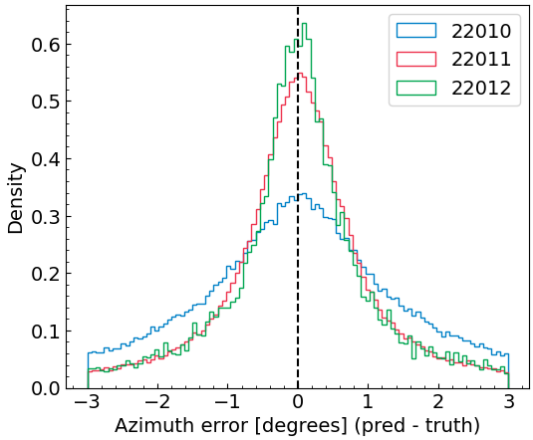


Figure 7.34: Distribution of the error between true and predicted azimuth angle by the transformer for the 22010, 22011 and 22012 datasets. The dashed line indicates an error of 0°.

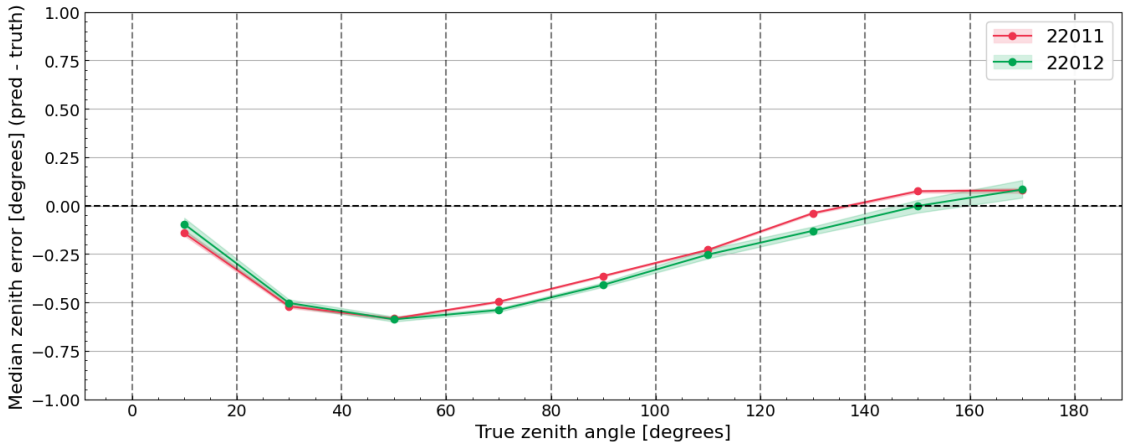


Figure 7.35: Median error in zenith prediction for the 22011 and 22012 datasets as a function of true zenith direction. The shaded area indicates the 1σ confidence bounds on the sample median.

8

Conclusion

A machine learning model was developed and trained to reconstruct the direction of high energy muon anti(neutrino) track events in the IceCube detector in the $100\,\mathrm{GeV} - 100\,\mathrm{PeV}$ energy range. The model architecture was inspired by the transformer encoder. The transformer computationally scales quadratically with the input sequence length due to the self-attention mechanism. However, the pulse map of a single high energy event in IceCube can have tens of thousands of pulses, which is far beyond the maximum sequence length a transformer can handle on regular hardware. Previous transformer implementations in IceCube solved this problem by truncating the pulse maps to a maximum number of pulses. This work took a different approach, aggregating pulses on a PMT level using summary statistics; a method termed PMT-fication. Using the PMTs as elements of the input sequence instead of the individual pulses, the maximum sequence length of the high energy events was significantly reduced, making the data suitable for transformer training.

Various small-scale transformer models were trained to study the effects of event selection, model and training set size, model shape, and the effectiveness of PMT-fication as a reduction method. These findings were later used to train a single, large-scale model. This chapter will present a summary of the main results of this work. Moreover, an outlook with recommendations for future work is given.

8.1 Summary

It was found that the choice of event selection for the training set can strongly impact the types of events the model can accurately reconstruct; using a strict selection criterion led to the best angular resolution of high-quality events, while generalization to the entire dataset was poor. Using a looser selection cut when training the model demonstrated improved average angular reconstruction across the entire dataset, but came at the expense of a minor decline in performance on the best events. It was decided to use the more relaxed cut for all other models in this work, as it allowed for bigger training set sizes, while also having the benefit of being purely physics-informed, without being biased towards an already existing reconstruction algorithm.

At a fixed model size, an approximate power law scaling of the opening angle with the number of training events was found. This result aligns with prior research on transformer scaling on text-based inputs. A slight flattening in the scaling curve was observed, suggesting the existence of a capacity limitation of the model, but this could not be further explored as no more training events were available. The influence of model size and shape was studied by varying the embedding dimension and number of encoder layers at a fixed training set size. Overall, increasing the model size improved the angular resolution, however, this was accompanied by increased training times. Interestingly, deep models performed slightly better than wide models with the same number of total parameters. This result contrasts with the findings of a previous study of transformers on language processing, which saw no dependency on model shape. On the other hand, the sequential nature of scaling model parameters in depth resulted in a near-linear scaling of the training time. The parallel computations of transformer width allowed scaling the number of parameters by a factor of 16, while only increasing training time by a factor of approximately 3. Therefore, the optimal transformer depth and width depend on a combination of available memory and training time.

The PMT-fication method was found to be an effective reduction method, especially for events with an energy over 10 TeV. Models trained on PMT-fied input data were able to reconstruct events with a large number of pulses well, while the opening angle for the model trained on pulse maps exploded past the truncation point. At low energies, a 20% loss was observed for the PMT-fied models compared to the pulse models, likely since most of these events have less than 256 pulses, so the aggregation of pulses led to an unnecessary reduction. Still, for transformers covering a broad energy range of IceCube events, PMT-fication seems to be a successful approach.

A large-scale final model was trained with 6 encoder layers, an embedding dimension of 512, and 4 parallel multi-heads, totaling 27 million trainable parameters. It was trained for 100 epochs, taking roughly three weeks. During inference, the transformer reconstructs about 1000 events per second. The transformer is able to match or outperform the traditional likelihood-based SplineMPE reconstruction method for starting tracks with a detector contained track length of less than 700 m, especially doing best in the TeV energy range. Moreover, the transformer improves the reconstruction for downgoing events. These findings highlight the potential of the model for the reconstruction of HESE tracks from the southern sky, contributing to the search for neutrino sources near the galactic center. For through-going tracks, the transformer can not match the resolution by SplineMPE (its median opening angle is about $0.3° - 0.5°$ larger than SplineMPE's across the entire energy spectrum). The transformer reconstructs the neutrino direction of through-going tracks at PeV energies with a median opening angle of 0.61°. For starting tracks, this is 0.66°. A $\sim 0.15° - 0.4°$ bias in under-predicting the zenith angle was found in the model, an effect strongest for high energy downgoing events.

8.2 Recommendations for future work

The results in this thesis show great potential for the application of transformers for the reconstruction of high energy neutrinos in the IceCube detector. However, there remain a lot of opportunities for continued development. This includes improvements to the existing model to enhance the angular resolution, but also the extension to other tasks and the reconstruction of real data. These ideas could no longer be implemented within the time frame of this work, but can hopefully serve as inspiration for future work.

The following points are suggestions of things to try to improve the angular resolution of the transformer:

- **correcting for the bias in zenith:** Even though the exact cause of the transformer bias in zenith prediction remained unclear, a $-0.4°$ offset for the highest energy dataset is a rather significant error to the median opening angle. It is worthwhile exploring machine learning strategies to prevent or mitigate this bias, ranging from data-level to model-level interventions. These could include re-weighting of events for which the bias is known to be strongest, to force the model to focus on these events, or modifying the loss function to penalize errors depending on their sign or by adding a bias regularization term.
- **changing the aggregation layer:** Instead of the `MeanPool` layer that is used at the output of the transformer for aggregation along the sequence dimension, one could explore if there are benefits to using a combination of mean, minimum, maximum and sum pooling, like implemented in the GraphNeT DynEdge model. Alternatively, it could be considered to add a special *classification token* to the sequence before the input is fed to the transformer, like done in the BERT model [125]. This way, the aggregation becomes a learnable part of the attention mechanism.
- **changing the regression layer:** Instead of the single linear layer currently used for the prediction of a (x, y, z) vector-head, it could be worthwhile using an MLP with non-linear

activation functions. For example, an MLP going from d_{emb} to a hidden layer of $4\,d_{emb}$ to d_{output} could be used to, hopefully, better translate the transformer encoder output to the neutrino direction. Alternatively, the MLP could be used to predict flow parameters, using *conditional normalizing flows* for a posterior prediction of the neutrino direction [126].

- **speeding up training with specialized libraries:** The implementation of libraries like `FlashAttention` [127] or `xformers` [128] could optimize the self-attention layer by making the computations faster and more memory efficient. This would allow for longer training, using bigger batch sizes or using more training events (if available), which could improve the angular resolution.
- **finetuning after bulk training:** The study of different event selections showed how much the model performance can depend on the quality of the training events. Possibly, this phenomenon could be used to *finetune* the model for performance on specific events; use a first stage training a model on bulk data with a loose selection cut, then apply a second stage training on a specific event type (like high-quality starting or through-going tracks) to improve the reconstruction of these events specifically. A similar method could be applied with the loss function, starting training with a loss function focused on stability (like the Euclidean distance), but switching loss function after a while to one more tailored to the target (like a cosine similarity loss for angular reconstruction).

In addition to these improvements, future research could extend this work with:

- **track reconstruction in real data:** Although the results on Monte Carlo data in this work are promising, there are no guarantees that they will generalize to real data. Therefore, an essential next step is to apply the transformer to the reconstruction of real tracks, including their noise, and get an estimate of the angular resolution on real data. However, this is easier said than done, as it requires a sample of real data events for which the direction is known. Possibly, one could use the moon shadow of cosmic ray induced muons for this [129], or use the *MINIONS* sample developed by Daniel Durnford at the University of Alberta (a sample of muon events with two bright *anchor DOMs*).
- **integrating the PMT-fication pipeline into GraphNeT:** As PMT-fication seems to be an effective approach to reduce the size of high energy IceCube events as input to machine learning models, it would be valuable to integrate this pipeline into GraphNeT. This facilitates other machine learning researchers working on high energy neutrino reconstruction to apply this method to their models, hopefully accelerating the overall improvement of high energy neutrino reconstruction algorithms.
- **cascade/energy reconstruction or PID classification:** This work has shown the potential of the transformer architecture for the angular reconstruction of high energy track-like neutrino events. However, as the transformer does not use any detector geometry as input, but rather learns these patterns from the training data, nothing about the transformer architecture makes it specific to this task. In general, the transformer architecture could be suitable for the reconstruction of arbitrary event topologies. It would therefore be interesting to see how the architecture performs on different tasks like the angular reconstruction of cascades, energy reconstruction of tracks/cascades, or the classification of signal/noise or neutrino flavors.

Appendix

A. Model details

This section presents complete details on the model training parameters and data used for the results that are presented in the central part of the thesis. Moreover, the `Jupyter notebook` used to create all of the figures can be found on GitHub at `analysis.ipynb`. The descriptions are short and without further motivation for parameters, since their main purpose is to allow for easier reproduction of the results.

Figures 7.1, 7.2, 7.3, 7.5, 7.4, 7.6

Details on the training data

Model name	Dataset ID	Training events	Selection cut train	Directory
Thorsten Cut	22011	1 417 728	Thorsten Cut	{5}
SplineMPE	22011	980 864	SplineMPE $< 2°$	{8}
TC + SplineMPE	22011	932 992	TC + SplineMPE	{8}

Table A.1

Training parameters

Model name	batch size	d_{emb}	layers	heads	seq len	max lr	epochs	dropout	target
Thorsten Cut	64	128	6	4	256	5e-4	100	0.1	lepton dir
SplineMPE	64	128	6	4	256	5e-4	100	0.1	lepton dir
TC+SplineMPE	64	128	6	4	256	5e-4	100	0.1	lepton dir

Table A.2

Results (validation)

Model name	Dataset ID	Validated on (part)	Data dir	Model path
Thorsten Cut	22011	4	{8}	{10}
SplineMPE	22011	4	{8}	{11}
TC + SplineMPE	22011	4	{8}	{12}

Table A.3

Figure 7.7

Details on the training data

Model name	Dataset ID	Selection cut train	Directory
100k	22011	Thorsten Cut	{5}
300k	22011	Thorsten Cut	{5}
500k	22011	Thorsten Cut	{5}
1M	22011	Thorsten Cut	{5}
2M	22011	Thorsten Cut	{5}

Table A.4

Training parameters

Model name	batch size	d_{emb}	layers	heads	seq len	max lr	epochs	dropout	target
100k	64	128	6	4	256	5e-4	100	0.1	neutrino dir
300k	64	128	6	4	256	5e-4	100	0.1	neutrino dir
500k	64	128	6	4	256	5e-4	100	0.1	neutrino dir
1M	64	128	6	4	256	5e-4	100	0.1	neutrino dir
2M	64	128	6	4	256	5e-4	100	0.1	neutrino dir

Table A.5

Results (validation)

Model name	Dataset ID	Validated on (part)	Data dir	Model path
100k	22011	4	{5}	{13}
300k	22011	4	{5}	{14}
500k	22011	4	{5}	{15}
1M	22011	4	{5}	{31}
2M	22011	4	{5}	{16}

Table A.6

Figure 7.9, 7.10

Details on the training data

Model name	Dataset ID	Training events	Selection cut train	Directory
All	22011	500 032	Thorsten Cut	{5}

Table A.7

Training parameters

batch size	d_{emb}	layers	heads	seq len	max lr	epochs	dropout	target	total params
64	128	4	4	256	5e-4	100	0.1	neutrino dir	815k
64	128	6	4	256	5e-4	100	0.1	neutrino dir	1.2M
64	128	8	4	256	5e-4	100	0.1	neutrino dir	1.6M
64	256	4	4	256	5e-4	100	0.1	neutrino dir	3.2M
64	256	6	4	256	5e-4	100	0.1	neutrino dir	4.7M
64	256	8	4	256	5e-4	100	0.1	neutrino dir	6.3M
64	512	4	4	256	5e-4	100	0.1	neutrino dir	12.5M
64	512	6	4	256	5e-4	100	0.1	neutrino dir	18.7M
64	512	8	4	256	4e-4	100	0.1	neutrino dir	24.9M

Table A.8

Results (validation)

Model name	Dataset ID	Validated on (part)	Data dir	Model path
emb128 layer4	22011	4	{5}	{17}
emb128 layer6	22011	4	{5}	{18}
emb128 layer8	22011	4	{5}	{19}
emb256 layer4	22011	4	{5}	{20}
emb256 layer6	22011	4	{5}	{21}
emb256 layer8	22011	4	{5}	{22}
emb512 layer4	22011	4	{5}	{23}
emb512 layer6	22011	4	{5}	{24}
emb512 layer8	22011	4	{5}	{25}

Table A.9

Table 7.4

Details on the training data

Model name	Dataset ID	Training events	Selection cut train	Directory
1 head	22011	500 032	Thorsten Cut	{5}
4 head	22011	500 032	Thorsten Cut	{5}
8 head	22011	500 032	Thorsten Cut	{5}

Table A.10

Training parameters

batch size	d_{emb}	layers	heads	seq len	max lr	epochs	dropout	target
64	512	8	1	256	4e-4	100	0.1	neutrino dir
64	512	8	4	256	4e-4	100	0.1	neutrino dir
64	512	8	8	256	4e-4	100	0.1	neutrino dir

Table A.11

Results (validation)

Model name	Dataset ID	Validated on (part)	Data dir	Model path
1 head	22011	4	{5}	{26}
4 head	22011	4	{5}	{25}
8 head	22011	4	{5}	{27}

Table A.12

Figure 7.11, 7.12, 7.13, 7.14

Details on the training data

Data type	Dataset ID	Training events	Selection cut train	Directory
Pulse map	22011	1 000 000	Thorsten Cut	{2}
PMT-fied	22011	1 000 000	Thorsten Cut	{5}

Table A.13

Training parameters

model	batch size	d_{emb}	layers	heads	seq len	max lr	epochs	dropout	target
Pulse	64	256	6	4	256	5e-4	100	0.1	neutrino dir
PMT	64	128	6	4	256	5e-4	100	0.1	neutrino dir

Table A.14

Results (validation)

Model name	Dataset ID	Validated on (part)	Data dir	Model path
Pulse	22011	4	{2}	{30}
PMT	22011	4	{5}	{31}

Table A.15

Figure 7.15, 7.16, 7.17

Details on the training data

Data type	Dataset ID	Training events	Selection cut train	Directory
Pulse map	22010	1 000 000	Thorsten Cut	{1}
PMT-fied	22010	1 000 000	Thorsten Cut	{4}

Table A.16

Training parameters

model	batch size	d_{emb}	layers	heads	seq len	max lr	epochs	dropout	target
Pulse	64	256	6	4	256	5e-4	100	0.1	neutrino dir
PMT	64	256	6	4	256	5e-4	100	0.1	neutrino dir

Table A.17

Results (validation)

Model name	Dataset ID	Validated on (part)	Data dir	Model path
Pulse	22011	4 (first 500 000)	{1}	{28}
PMT	22011	4 (first 500 000)	{4}	{29}

Table A.18

Figure 7.18, 7.19, 7.20, 7.21, 7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33, 7.34, 7.35

Details on the training data

Model name	Dataset ID	Training events	Selection cut train	Directory
Final model	{22010, 22011, 22012}	3 944 233	Thorsten Cut	{4},{5},{6}

Table A.19

Training parameters

batch size	d_{emb}	layers	heads	seq len	max lr	epochs	dropout	target
64	512	6	4	256	4e-4	100	0.1	neutrino dir
64	256	6	4	256	5e-4	100	0.1	neutrino dir

Table A.20

Results (test)

Figure	Dataset ID	Tested on (part)	Selection	Data dir	Model path
7.18	22010	10 (first 500 000)	TC & lvl 3	{4}	{32}
7.19	22011	9	TC & lvl 3	{5}	{32}
7.20	22012	10	TC & lvl 3	{6}	{32}
7.21	{22010, 22011, 22012}	{10, 9, 10}	TC & lvl 3	{4},{5},{6}	{32}
7.22	{22010, 22011, 22012}	{10, 9, 10}	TC & lvl 3 & through	{4},{5},{6}	{32}
7.23	{22010, 22011, 22012}	{10, 9, 10}	TC & lvl 3 & starting	{4},{5},{6}	{32}
7.24	{22010, 22011, 22012}	{10, 9, 10}	TC & lvl 3	{4},{5},{6}	{32}
7.25	{22010, 22011, 22012}	{10, 9, 10}	TCt & lvl 3 & through	{4},{5},{6}	{32}
7.26	{22010, 22011, 22012}	{10, 9, 10}	TC & lvl 3 & starting	{4},{5},{6}	{32}
7.27	22010	10 (first 500 000)	TC & lvl 3	{4}	{32}
7.28	22011	9	TC & lvl 3	{5}	{32}
7.29	22012	10	TC & lvl 3	{6}	{32}
7.30	{22010, 22011, 22012}	{10, 9, 10}	TC	{4},{5},{6}	{32}
7.31	{22010, 22011, 22012}	{10, 9, 10}	TC	{4},{5},{6}	{32}
7.32	{22010, 22011, 22012}	{10, 9, 10}	TC	{4},{5},{6}	{32}
7.33	{22010, 22011, 22012}	{10, 9, 10}	TC	{4},{5},{6}	{32}
7.34	{22010, 22011, 22012}	{10, 9, 10}	TC	{4},{5},{6}	{32}
7.35	{22011, 22012}	{9, 10}	TC	{5},{6}	{32}

Table A.21

B. Statistical uncertainty on the median

This section serves as a detailed explanation of how to calculate the statistical uncertainty on the median of a distribution of arbitrary shape. This method has been used various times in this thesis to represent uncertainty bands on median values, which were binned as a function of some other value. The same method can be applied to calculate uncertainty bounds on any other percentile of a distribution. As an example, consider the distribution of the opening angle of the blue and red dataset in figure B.1. What these datasets entail exactly is not important for this explanation. Dashed lines indicate the medians of the distributions.

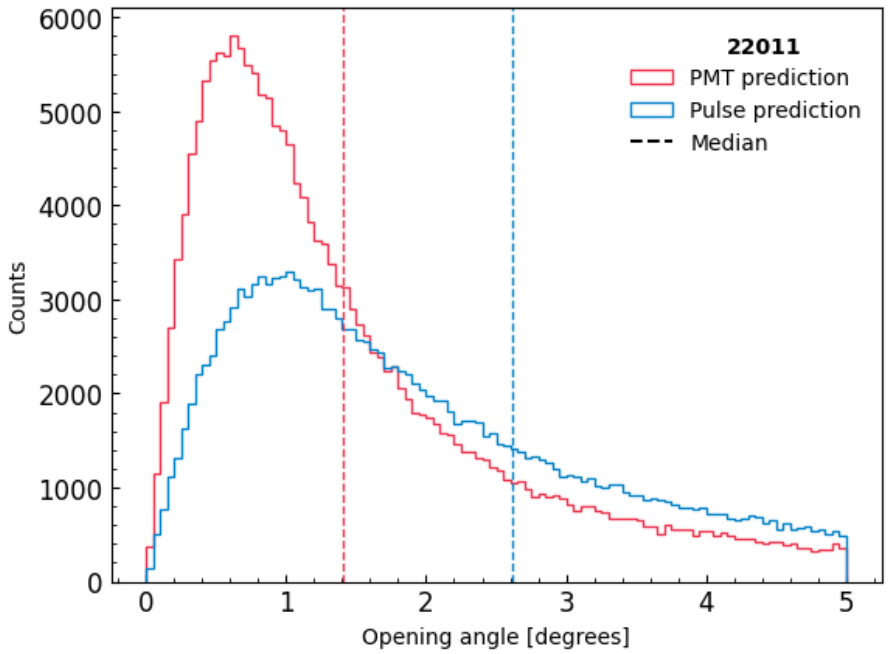


Figure B.1: Example distributions of the opening angle of two different models (blue and red)

The goal is to investigate how the median opening angle changes as a function of some other parameter. This is done by binning the entire distribution as a function of this parameter and calculating a median within each bin. An example is given in figure B.2, where the opening angles of the blue and red distributions are binned in nine bins of a variable number of pulses per event. The problem with this figure is that there are no uncertainties given on these medians. The medians of the models seem to diverge at a higher number of pulses per event, but is this statistically significant or just an artifact of the strongly decreasing number of events per bin at higher pulse per event?

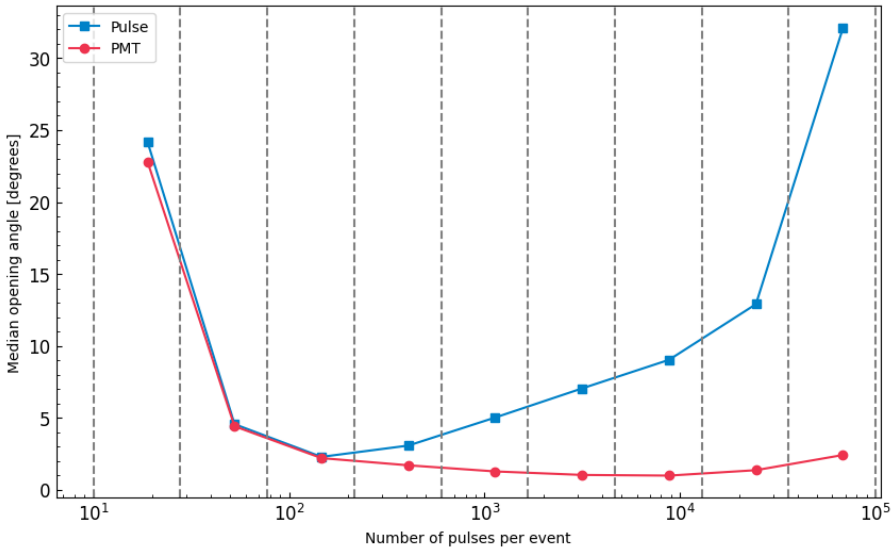


Figure B.2: Example of the median of a distribution as a function of some other variable, by binning the full distribution.

But how does one calculate the uncertainty on the median of an arbitrary (non-Gaussian) distribution? To understand this further, figure B.3 shows the distributions of the opening angle in the first and last bin of figure B.2, left and right, respectively. The medians of the distributions are once again indicated by dashed lines. Intuitively, one expects the medians in the left figure to be more rigorous than those in the right figure since the sample size is very small in the latter. This intuition is correct, but how does this translate to actual uncertainty bounds, especially for a distribution as sparse as on the right of figure B.3?

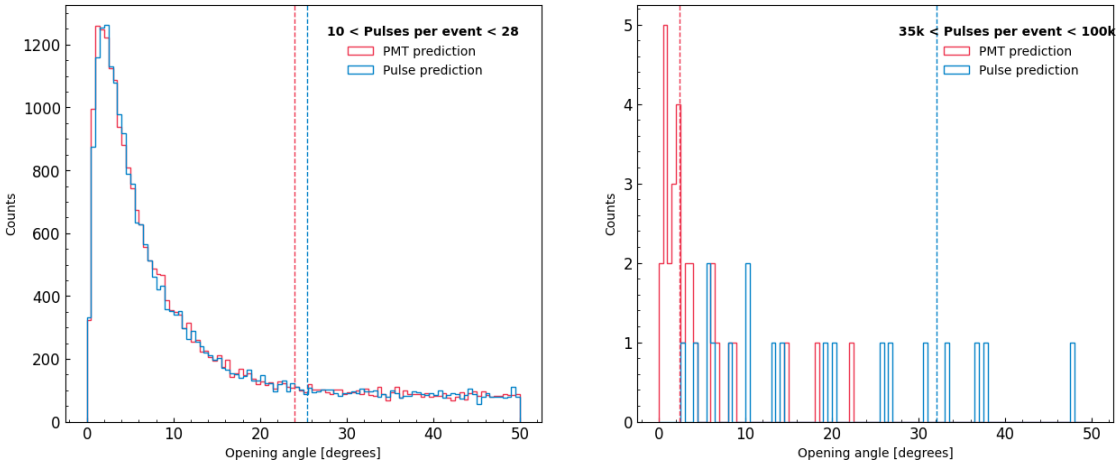


Figure B.3: Example of the distributions of the opening angle for the first and last bin in figure B.3. The median of the distributions is indicated with a dashed line.

The key to answering this question is the realization that the expected index of the true median follows a binomial distribution, independent of the underlying distribution. By definition, without knowing the true shape of a PDF, independently sampled points from this distribution each have a probability of $p = 0.5$ to be below the true median of the PDF. If every time a sampled point is below the true median is defined as a 'success', the number of successes is binomially distributed

with success probability $p = 0.5$ and total number of sampled points N, described by:

$$P(X = k) = \binom{N}{k} p^k (1-p)^{N-k} \tag{8.1}$$

An example of a binomial distribution with $p = 0.5$ and $N = 100$ is shown in figure B.4. Unsurprisingly, the expected number of samples with a value below the true median is centered around 50. This number is equal to the index of the observed median of an arbitrary distribution. However, the binomial distribution also defines the uncertainty of this number. In this way, the 16% and 84% confidence intervals of the binomial distribution can be used to present a 1σ confidence bound on the sample median. This is indicated in red in figure B.4. These intervals can be found by integrating the analytical expression of the binomial distribution. For large N, the binomial distribution can be approximated by a Gaussian, and the 1σ confidence can be found by

$$\sigma \approx \sqrt{Np(1-p)} \tag{8.2}$$

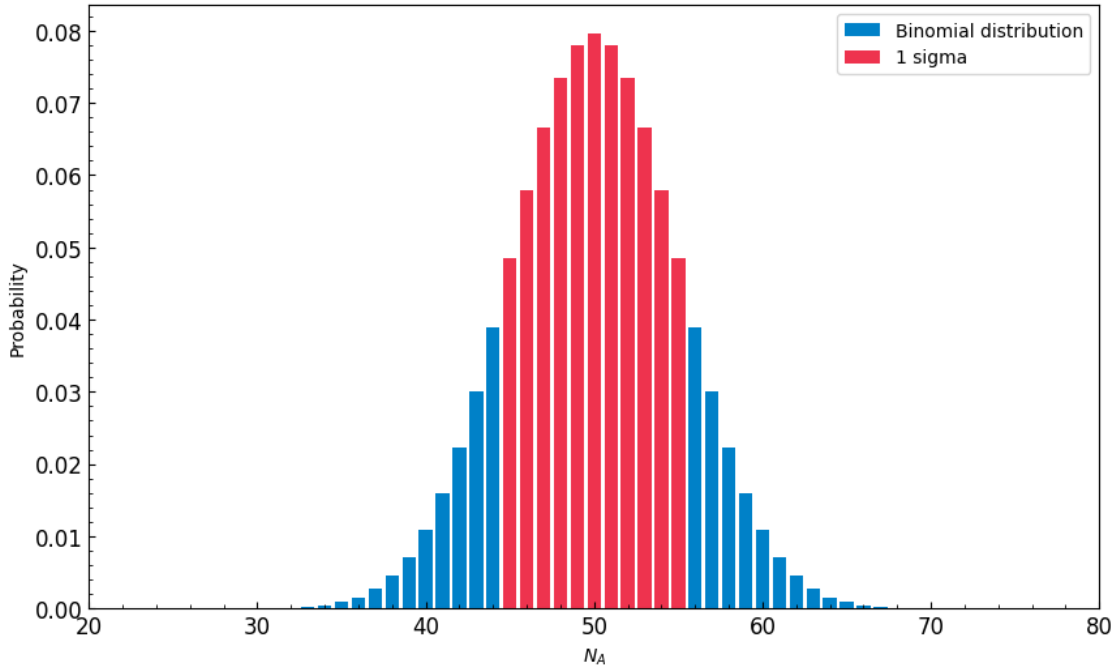


Figure B.4: Example of a binomial distribution with success probability $p = 0.5$ and number of trials $N = 100$. The red area indicates the 1σ confidence interval of the distribution.

Under the assumption that the observed samples are sampled independently from a single underlying true PDF, the values at 16% and 84% of the binomial distribution represent the index of the lower and upper uncertainty bound on the median in the ordered set of observations. Applying this logic to the distributions in figure B.3, the lower and upper uncertainty bounds on the median can be found as indicated by the shaded areas in figure B.5. Note that these uncertainties do not have to be symmetric, since they only depend on the index of the ordered sample.

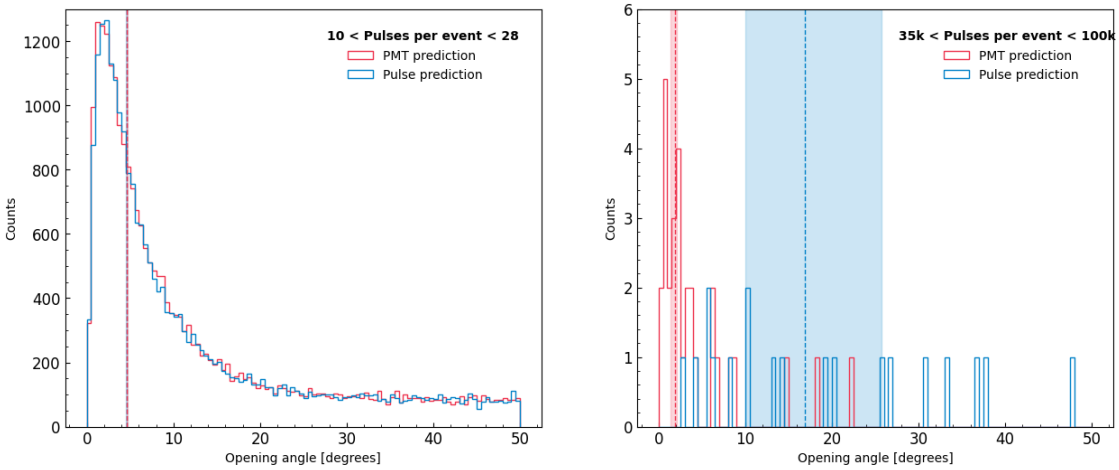


Figure B.5: The same distributions as shown in figure B.3, this time including a shaded area indicating the uncertainty on the median. This uncertainty bound is determined by using the 1σ confidence interval of the binomial as indices of the lower and upper bound in the ordered sample.

Repeating this calculation for each of the binned opening angle distributions, figure B.2 can be remade with statistical uncertainties on the median, as shown in figure B.6. The uncertainty on the median is small for bins with a high number of events, while for bins with a low number of events, it is likely to be larger. The exact uncertainties depend on the distributions of the samples.

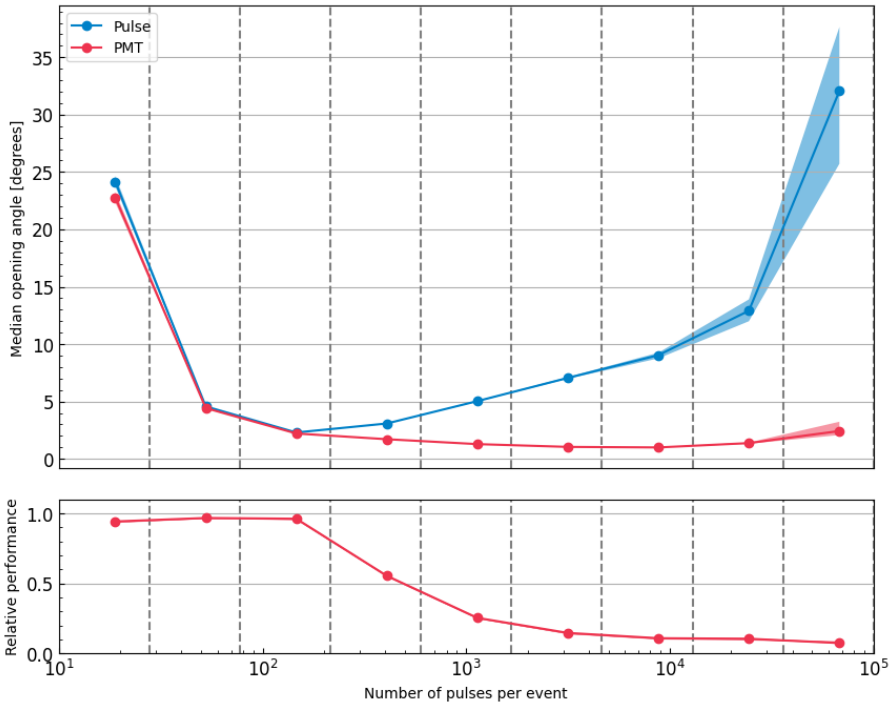


Figure B.6: The same as figure B.1, this time including a statistical uncertainty on the median as a shaded area.

C. List of file locations on the HPC cluster

Data directories

{1} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/sqlite_pulses/Snowstorm/22010

{2} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/sqlite_pulses/Snowstorm/22011

{3} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/sqlite_pulses/Snowstorm/22012

{4} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/PMTfied_filtered/Snowstorm/CC_CRclean_MuonLike/22010

{5} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/PMTfied_filtered/Snowstorm/CC_CRclean_MuonLike/22011

{6} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/PMTfied_filtered/Snowstorm/CC_CRclean_MuonLike/22012

{7} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/PMTfied/Snowstorm/22010

{8} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/PMTfied/Snowstorm/22011

{9} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/PMTfied/Snowstorm/22012

Trained models

{10} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/event_selection_tc.ckpt

{11} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/event_selection_splinempe.ckpt

{12} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/event_selection_tc_and_splinempe.ckpt

{13} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/100k_training_size.ckpt

{14} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/300k_training_size.ckpt

{15} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/500k_training_size.ckpt

{16} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/2mil_training_size.ckpt

{17} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb128_layer4.ckpt

{18} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb128_layer6.ckpt

{19} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb128_layer8.ckpt

{20} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb256_layer4.ckpt

{21} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb256_layer6.ckpt

{22} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb256_layer8.ckpt

{23} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb512_layer4.ckpt

{24} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb512_layer6.ckpt

{25} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb512_layer8.ckpt

{26} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb512_layer8_head1.ckpt

{27} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/emb512_layer8_head8.ckpt

{28} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22010/pulse/1mil_training_size.ckpt

{29} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22010/PMTfied/1mil_training_size.ckpt

{30} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/pulse/1mil_training_size.ckpt

{31} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/22011/PMTfied/1mil_training_size.ckpt

{32} /lustre/hpc/project/icecube/HE_Nu_Aske_Oct2024/trained_models/angular_reconstruction/tracks/all_energies/PMTfied/final_transformer_model_luc.ckpt

D. Additional plots

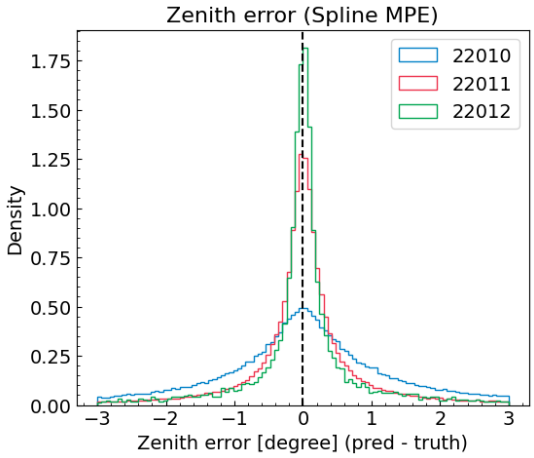


Figure D.1: Distribution of the error between predicted and true zenith angle for the SplineMPE algorithm, for the 22010, 22011 and 22012 datasets. The black dashed line indicates an error of 0°.

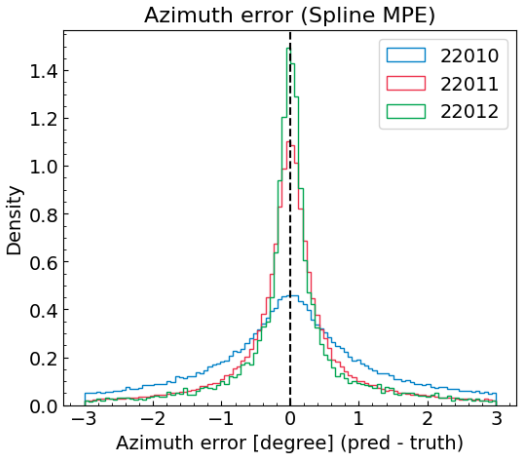


Figure D.2: Distribution of the error between predicted and true azimuth angle for the SplineMPE algorithm, for the 22010, 22011 and 22012 datasets. The black dashed line indicates an error of 0°.

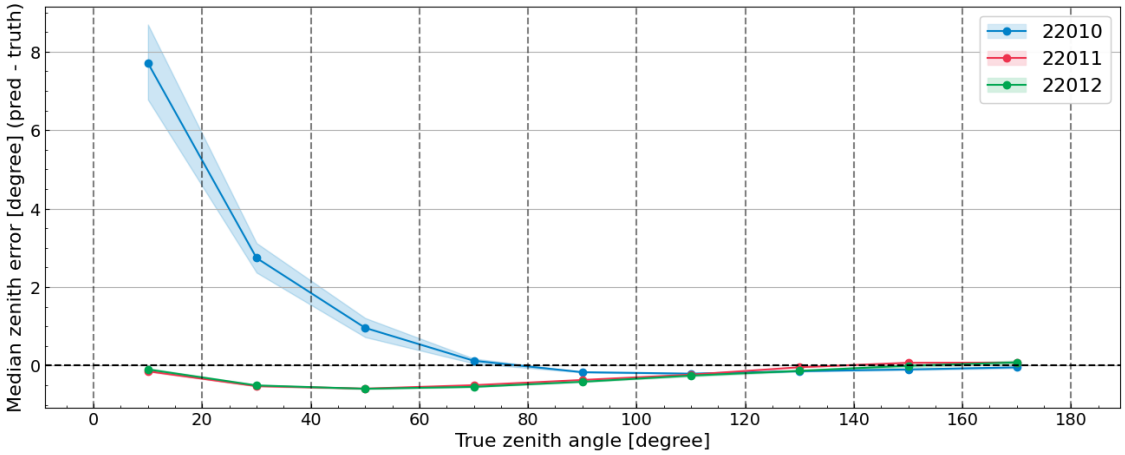


Figure D.3: Median error in zenith prediction as a function of true zenith direction for the 22010, 22011 and 22012 dataset.

Glossary

activation function Non-linear mathematical function determining the output of an artificial neuron, applied after the affine transformation. Popular activation functions include the rectified linear unit (ReLU) and the sigmoid.

active galactic nucleus (AGN) Compact region at the center of a galaxy that emits large amounts of electromagnetic radiation.

adamW Variant of the Adam optimizer used in machine learning, decoupling weight decay from the optimization step, separating it from the gradient update.

adam Adaptive moment estimation, a popular machine learning optimizer that combines the momentum method and an adaptive per-parameter learning rate.

after-pulse Photon detection at a delay of $\sim 300 - 1100\,\text{ns}$ as a result of ionized gas molecules inside the photomultiplier tube (PMT) striking a dynode, causing a second cascade of electrons.

air shower Cascades of subatomic particles and ionized nuclei produced by the interaction of a primary cosmic ray with the Earth's atmosphere.

amanda Antarctic Muon and Neutrino Detector Array, the predecessor to the IceCube detector.

analog transient waveform digitizer (ATWD) One of the waveform digitizers used in IceCube digital optical modules (DOMs) together with the fast analog digital converter (fADC). The ATWD digitizes waveforms at a resolution of $\sim 3\,\text{ns}$, covering an interval of $\sim 420\,\text{ns}$.

apache parquet Open-source column-oriented data storage format, optimized for bulk and parallel reads.

artificial neuron The elementary unit of an artificial neural network; a mathematical function conceived as a model of a biological neuron.

astrophysical neutrinos Neutrinos in the energy range $100\,\text{GeV} - 100\,\text{PeV}$ originating from cosmic sources like supernovae or AGNs.

atmospheric neutrinos Neutrinos in the $\text{MeV} - \text{PeV}$ range resulting from cosmic ray interactions with the Earth's atmosphere.

attention is all you need A 2017 landmark paper in machine learning by google researchers introducing the transformer architecture for language processing purposes.

attention mask Boolean matrix used to selectively ignore the attention weights between certain sequence elements during the attention calculation in a transformer. Examples are a causal attention mask or a padding attention mask.

attention A machine learning method that determines the relative importance of a sequence element to all other elements in the sequence. This mechanism is key to the transformer architecture.

backpropagation A method to estimate the gradient of a neural network efficiently using the chain rule. This gradient is used to determine parameter updates during training of a machine learning model.

baryon Type of hadron made of an odd number of quarks, like the proton and the neutron.

batch size Number of events in a batch that are processed together in one forward and/or backward pass of the model.

batch A subset of the training dataset used to compute one update of the model's weights during training, used to balance training efficiency and stability.

blazar Active galactic nucleus with a relativistic jet directed towards an observer.

cache Hardware or software component in computing used for temporary storage of data such that future access to that data is faster.

cascade template Template used for the reconstruction of cascade events in IceCube.

cascade Type of event detected in the IceCube detector caused by an ν_e, ν_τ or any neutral current (NC) interaction.

causal attention mask Attention mask used in a transformer decoder to prevent tokens from attending to future tokens in the sequence, crucial for tasks like language generation.

charged current (CC) interaction Type of weak interaction under the exchange of a W boson.

cherenkov angle Angle of emission of Cherenkov photons, depending on the velocity of the charged particle and the refractive index of the medium. For IceCube, the Cherenkov angle is $\sim 41°$.

cherenkov radiation Electromagnetic radiation emitted when a charged particle passes through a dielectric medium (like water or ice) at a speed larger than the phase velocity of light in that medium.

classification Type of supervised machine learning task where the goal is to assign input data to one or multiple predefined, discrete classes.

color confinement Phenomenon that color-charged particles (like quarks) cannot be isolated.

convolutional neural network (CNN) A type of feed forward neural network designed to process data with a grid-like structure, like images, by applying learnable filters through convolutional layers.

corsika A Monte Carlo simulation program used to simulate air shower events from cosmic ray interactions with the Earth's atmosphere. Mainly used for the simulation of downgoing atmospheric muons.

cosmic microwave background (CMB) Microwave radiation that fills all space in the observable universe, relic from the Big Bang. Also known as 'relic radiation'.

cosmic rays High energy particles or clusters of particles that move through space at nearly the speed of light.

cosmogenic neutrinos Ultra high energy neutrinos originating from cosmic ray interactions with the cosmic microwave background. Their detection has not been confirmed yet.

cosmological neutrinos Very low energy neutrinos that are a relic of the Big Bang. Also called 'relic neutrinos'.

cowan-reines neutrino experiment First experimental detection of the (anti)neutrino by observing the interaction of $\bar{\nu}_e$ with protons.

cross section Measure of probability in particle physics for a specific process to take place in a collision of two particles.

cross-attention Type of machine learning attention occurring between two sequences, typically using the relevant information from one sequence to transform the second sequence. Used in the decoder half of the transformer architecture.

decoder Second half of the transformer architecture, tasked with generating output sequences based on the output of the encoder.

deep inelastic scattering Type of collision between two high energy leptons and nucleons. The struck nucleus is shattered and causes a hadronic cascade of particles.

deep learning Machine learning with deep neural networks.

deepcore A denser detector region in the middle of the IceCube detector, focused on researching lower energy neutrinos.

detector signature IceCube detector response to different types of neutrino interactions, divided into tracks and cascades.

digital optical module (DOM) Glass sphere that houses the photomultiplier tube (PMT) and required electronics to detect events in IceCube.

dirac fermions A fermion which is different from its antiparticle. All fermions in the standard model fall in this category, possibly except for the neutrinos.

dom launcher IceCube module responsible for the simulation of digital optical module (DOM) electronics, including the discriminator, digitization and effects of electronic noise.

double bang Detector signature of a ν_τ, with a spatially separate charged current (CC) interaction vertex and tau lepton decay vertex causing a hadronic cascade and electromagnetic cascade respectively.

dropout Regularization technique used in training machine learning models. During training, a fraction of the model weights are randomly zeroed out on the forward pass, used to improve generalization and prevent overtraining.

dust layer Layer of optical impurities in the ice of the IceCube detector at $2000 - 2100\,\mathrm{m}$ depth.

dynedge A graph neural network (GNN) from GraphNeT using dynamic edge convolutions, used in IceCube for tasks like low energy event classification and reconstruction.

early stopping Technique used when training machine learning models to stop training the model when the validation set's performance no longer improves. Used to prevent model overtraining.

empirical loss Normalized loss on a subset of the dataset, often used as the metric to minimize during training. Often also referred to as 'training loss'.

encoder block The combination of the multi-head attention and feed-forward layer - including residual connections and layer normalization- in the transformer encoder. Repeated multiple times to form the encoder.

encoder First half of the transformer architecture, tasked with processing the input sequences and learning dependencies and patterns between input elements.

epoch A complete pass through the entire training dataset while training a machine learning model. Typically, models are trained for multiple epochs.

event selection The process of selecting a subset of well-reconstructible events from the dataset to use as the training set for a machine learning model.

event trigger Algorithms in IceCube that trigger the recording of all waveform readouts for a certain time window. Various triggers exists.

event All hits recorded within an event window combined, including hard local coincidence and soft local coincidence hits.

exploding gradient Problem during training of a machine learning model when gradients of the loss function become excessively large, causing the weights to grow uncontrollably, resulting in model instability.

fast analog digital converter (FADC) One of the waveform digitizers used in IceCube digital optical modules (DOMs), together with the analog transient waveform digitizer (ATWD). Compared to the ATWD, the FADC digitizes waveforms at a lower resolution of $\sim 25\,\mathrm{ns}$, but covering a longer interval of $\sim 6400\,\mathrm{ns}$.

feature space The domain of the input data in machine learning. A multi-dimensional mathematical space where each dimension represents a feature (or variable) used to describe the data points, often denoted by X.

feed forward network Subset of neural network functions where data flows in one direction, from input to output, without cycles or loops. A term often used interchangeably with multilayer perceptron (MLP).

fermion A half-integer spin subatomic particle that follows Fermi-Dirac statistics.

feynman diagram Graphical representation of the behavior and interaction of subatomic particles, named after Richard Feynman.

flasher Short for flasher boards; every digital optical module (DOM) in IceCube has twelve LED flashers, used for calibration of geometry, ice properties and photo multiplier tube (PMT) properties.

flavor Species of an elementary particle. The standard model has six flavors of quarks and six flavors of leptons.

flops Floating point operations per second, a measure of computational performance or efficiency.

gamma-rays High energy electromagnetic radiation.

gauge boson A bosonic elementary particle that acts as the force carrier for fermions.

generalization Machine learning term referring to the model's ability to perform on unseen data, as compared to the training data.

generative pre-trained transformer (GPT) Type of large language model (LLM) based on the transformer machine learning architecture. Trained on large amounts of text to generate human-like content.

generator Software responsible for the creation of the primary particle during simulation. Different generators exist for the creation of low energy neutrinos (Genie), high energy neutrinos (NuGen) or cosmic rays (Corsika).

genie A program used for the detailed Monte Carlo simulation of low energy neutrino interactions in IceCube.

geo antineutrinos See terrestrial antineutrinos.

glashow resonance The resonant formation of the W boson in antineutrino-electron collisions at 6.3 PeV.

gluon A massless particle mediating the strong force, binding quarks together into hadrons.

gradient descent First-order iterative mathematical optimization method that can be used to update model weights when training a machine learning model.

graph neural network (GNN) A type of neural network architecture that is designed to take graphs as input.

graphics processing unit (GPU) A specialized processor designed for handling parallel computations, originally developed for graphics rendering but later found to be useful for non-graphic parallel calculations like training machine learning models.

graphnet An open-source Python framework providing functionality to perform reconstruction tasks at neutrino telescopes using deep learning.

hadron Composite subatomic particle made of two or more quarks, held together by the strong force.

hard local coincidence Quality flag to an IceCube hit indicating if neighboring digital optical modules (DOMs) also registered a hit within a certain time window.

helicity suppression Suppression of decay rates or interaction cross sections in certain weak interactions due to helicity of the particles involved. Conservation laws can make certain helicity states unfavorable or forbidden, lowering the probability of the interaction taking place.

helicity Projection of the spin (chirality) of a particle onto its linear momentum. Also called 'handedness', with left-handed negative helicity and right-handed positive helicity.

higgs mechanism Generation mechanism of mass for the gauge bosons through electroweak symmetry breaking.

high energy starting event (HESE) Type of high energy neutrino event in IceCube with its interaction vertex within a sub-region of the detector.

hit A pulse detected by the anode of the photomultiplier tube (PMT) which exceeds the discriminator threshold of 0.25 pe.

hyperparameter Tunable parameter in machine learning that is set before training the model which controls the learning process, such as model size or learning rate.

i3 file IceCube custom data file type, split in various frames with information about the detector, calibration, run, or detected events.

ic22 Partial detector configuration during building of IceCube named after the number of strings at the time of data collection (22).

ic40 Partial detector configuration during building of IceCube named after the number of strings at the time of data collection (40).

ic59 Partial detector configuration during building of IceCube named after the number of strings at the time of data collection (59).

ic79 Partial detector configuration during building of IceCube named after the number of strings at the time of data collection (79).

ic86 Current IceCube detector configuration, representing the configuration with 78 main array strings and 8 DeepCore strings.

icecube neutrino observatory (IceCube) A neutrino detector developed by the University of Wisconsin–Madison and constructed at the Amundsen–Scott South Pole Station in Antarctica.

icecube upgrade The installation of 700 new digital optical modules (DOMs) on 7 new strings within the current DeepCore array, planned for late 2025.

icecube-gen2 Generation two collaboration of IceCube, hoping to add another 120 strings with detectors to the IceCube array for the detection of astrophysical neutrinos, increasing the instrumented volume to $\sim 8\,\mathrm{km}^3$.

icetop Collection of detectors at the surface of the ice, above the IceCube main array. IceTop consists of 81 sets of two tanks with ice, each tank containing two digital optical modules (DOMs).

icetray The IceCube software framework responsible for managing interactions between i3 modules, used for simulation, processing and analysis of data files.

kaggle Data science competition platform.

l2 regularization See weight decay.

label space The set of all possible output labels or target values that a machine learning model can predict. Can be both discrete (classification) or continuous (regression).

label Machine learning term for an element of the label space. Often used as the prediction target in supervised learning.

late-pulse Photon detection at a delay of $\sim$ 60 ns as a result of the first photo electron being reflected on the first dynode, only triggering the dynode with the second impact.

layer normalization Mathematical function used in machine learning that normalizes the data across all features within a single data sample; key in transformer models.

learning bias The set of assumptions made when training a machine learning model, including both assumptions on the feature/label space and model design decisions.

learning rate scheduler Method to adjust the learning rate during training a machine learning model according to a predefined schedule. Can help with model convergence.

learning rate Tuning parameter used in machine learning to determine the step size at each iteration of the optimization algorithm.

lepton number Conserved quantum number representing the difference between the number of leptons and antileptons. Leptons have lepton number $+1$, while antileptons have lepton number -1.

lepton A half-integer spin elementary particle that does not undergo strong interactions.

level IceCube term to indicate the data processing and filtering stage, from raw, uncleaned events to events filtered for a specific scientific goal.

line fit Simple first-guess directional reconstruction algorithm in IceCube using a χ^2 minimization.

loss function A mathematical formula to calculate the loss of a machine learning model.

loss A measure for the quality of a machine learning model, describing the difference between a model's predicted output and the true target value. The goal of training machine learning models is to minimize their loss.

machine learning (ML) Field of study in artificial intelligence developing statistical algorithms that can learn from data and generalise to unseen data.

majorana fermions A fermion which is its own antiparticle. None of the fermions in the standard model fall in this category, possibly except for the neutrinos.

meson Type of hadron made of an even number of quarks (equal amount of quarks and antiquarks), like the pion or kaon.

model weights The collection of parameters within a machine learning model that define the mapping from input to output. These parameters are 'learned' during training.

momentum method A machine learning optimization technique updating weights using a momentum term; the moving average of past gradients. Can help stabilize and speed up training.

multi-head attention Attention mechanism in transformer models using multiple parallel smaller single attention heads. Allows the model to focus on different aspects of the input data simultaneously and learn more effectively.

muon filter Common name used for the IceCube filter `MuonFilter_13`, aimed to select events whose detector signal looks like a muon neutrino.

network depth Number of layers with learnable parameters between the input and output in a machine learning model.

network width The (maximum) number of nodes in a single network layer in a machine learning model.

neural network (NN) A network of interconnected artificial neurons, used as a mathematical model to approximate non-linear functions.

neutral current (NC) interaction Type of weak interaction under the exchange of a Z boson.

neutrino flavor eigenstates The three leptonic flavors in which a neutrino can be created: electron neutrino (ν_e), muon neutrino (ν_μ) and tau neutrino (ν_τ).

neutrino interaction angle Angle between the incoming neutrino direction and the trajectory of the outgoing particles produced in a neutrino interaction.

neutrino mass eigenstates The three discrete neutrino masses, label '1', '2', and '3'.

neutrino mass hierarchy The ordering of the three neutrino mass eigenstates, with two possible configurations. In the 'normal hierarchy', mass 2 is lighter than mass 3, while in the 'inverted hierarchy', the opposite would be true..

neutrino oscillation Quantum mechanical phenomenon in which a neutrino created in a specific flavor eigenstate can later be measured in another flavor eigenstate.

northern sky The northern celestial hemisphere. Term to indicate neutrino events in IceCube that traveled through the Earth's core.

northern track Type of high energy neutrino event in IceCube, coming from the northern hemisphere (traveling through the Earth).

nugen Neutrino generator, a Monte Carlo simulation program used in IceCube to simulate high energy neutrino events.

one cycle Type of learning rate scheduler used in machine learning. It starts with a short warm-up to allow for exploration of the loss landscape, followed by a gradual decrease of learning rate to improve the chance of convergence.

opening angle Angle between the predicted and true neutrino direction vectors in degrees. Measure angular reconstruction accuracy.

optimizer An algorithm used in machine learning that determines how the weights are updated based on the gradients of the loss functions with respect to the model parameters. Popular optimizers include stochastic gradient descent (SGD), Adam and AdamW.

overtraining Problem in machine learning when the model has learned features from the training set that do not generalize to an unseen dataset, like the validation or test set.

padding attention mask Attention mask used in a transformer encoder that uses input sequences of variable length, which are padded to a constant length. Ensures that padded sequence elements do not contribute to the attention weight matrix.

pandelMPE Likelihood-based directional reconstruction algorithm in IceCube using the Pandel function as an approximation for the residual photon time PDFs.

pe Photoelectron, unit of charge used in IceCube, relating the recorded charge (in Coulombs) to the number of photons that hit the digital optical module (DOM).

photomultiplier tube (PMT) Tube with a photocathode, multiple dynodes and an anode, used to amplify the signal from the absorption of a single photon.

photon propagation code (PPC) IceCube software that computes the propagation of (Cherenkov) photons inside the detector.

photon A massless particle mediating the electromagnetic force.

pmns matrix Neutrino mixing matrix described by Pontecorvo, Maki, Nakagawa and Sakata as a model for neutrino oscillations.

pmt response simulator IceCube module used to simulate the photomultiplier tube (PMT) response to detected photons. Includes generation of pre-pulses, late-pulses and after-pulses, as well as applying jitter and saturation effect..

pmt-fication Data aggregation method by statistically summarizing IceCube pulses on a photomultiplier tube (PMT) basis.

polyplopia IceCube software used for injecting atmospheric shower background events on top of primary events in the simulation chain.

pooling layer Type of layer in a neural network that aggregates information dispersed among many vectors into fewer vectors. Popular pooling layers include max pooling and mean pooling.

pre-pulse Early photon detection as the result of the photon passing the photocathode in the photomultiplier tube (PMT) and striking the first dynode directly.

processing and filtering (PnF) Algorithm run at the South Pole, responsible for the bulk of the data reduction before transmission via satellite..

proposal Software used for the propagation of leptons in IceCube, parameterizing energy losses in the ice.

pulse map The collection of hits of an event after waveform unfolding.

pulse series See pulse map.

pytorch Machine learning (ML) library based on Torch developed by Meta AI. One of the most popular ML frameworks to use in Python.

quark Type of elementary particle and fundamental constituent of matter.

random access memory (RAM) Form of temporary, fast computer memory that can be read and changed in any order, typically used to store working data.

rectified linear unit (ReLU) Popular activation function used in machine learning, defined as the non-negative part of its argument.

recurrent neural network (RNN) A type of neural network designed to take sequential data as input, such as text or a time series, utilizing recurrent connections to capture temporal dependencies.

regression Type of supervised machine learning task where the goal is to predict one or multiple continuous target values based on the input data.

scalar boson A boson with zero spin. The only fundamental scalar boson is the Higgs boson.

seeded radius time (SRT) cleaning Iterative IceCube hit cleaning algorithm using hard local coincidence (HLC) hits as seeds for the selection of valuable soft local coincidence (SLC) hits to be added to the event.

segmented spline reco Improvement to the splineMPE reconstruction method. Splits the muon track into segments, considering stochastic energy losses for each of the segments.

self-attention Type of machine learning attention occurring in a single sequence, used in the encoder half of the transformer architecture.

sigmoid Popular activation function used in machine learning with a characteristic S-shaped curve.

simple multiplicity trigger (SMT) Primary event trigger in IceCube, looking for 8 hard local coincident (HLC) hits within 5 μs in the main array or 3 HLC hits within 2.5 μs within DeepCore.

skip connection A residual connection in neural networks where the input of a layer is directly added to its output, bypassing the layer. This helps with gradient flow, reducing the problem like vanishing gradients.

snowstorm ensemble set SnowStorm event dataset using a combination of different nuisance parameters based on their allowed phase space. By reweighting the ensemble set, an 'old style' discrete simulation set can be reproduced.

snowstorm Simulation method developed for IceCube to handle systematic uncertainties on physics parameters during Monte Carlo simulations.

softmax Popular mathematical function used in machine learning, converting a vector of K real numbers into a probability distribution of K possible outcomes, often used as the last layer in a regression model.

solar neutrinos Electron (anti)neutrinos from the sun produced during fusion.

south pole ice (SPICE) models Collection of four generations of ice models describing the optical properties of the IceCube detector.

southern sky The southern celestial hemisphere. Term to indicate neutrino events in IceCube that come from the top of the detector.

splineMPE cut Event selection criterion, only selecting events that the SplineMPE reconstruction algorithm can reconstruct within 2°.

splineMPEmax SplineMPE directional reconstruction method at max settings, including extra hit cleaning and noise modeling.

splineMPE Likelihood-based directional reconstruction algorithm in IceCube using a B-spline approximation for the residual photon time PDFs.

standard model of particle physics The theory describing three of the four known fundamental forces and classifying all known elementary particles.

stochastic gradient descent (SGD) Extension of 'gradient descent' optimization algorithm, using a stochastic approximation of the true gradient calculated from the entire dataset by calculating the gradient on a subset (batch) of the dataset.

string Support cables in IceCube to which the digital optical modules (DOMs) are connected.

strong scaling Notion of scalability in high performance computing; how the solution time scales with the number of processors for a fixed total problem size.

supernova early warning system (SNEWS) Global network of neutrino experiments sensitive to supernova neutrinos, aiming to provide the astronomical community with a prompt alert of a supernova event.

supernova neutrinos Neutrinos radiated by a massive star during a supernova.

supervised learning Branch of machine learning using labeled training data.

terrestrial antineutrinos Electron antineutrinos produced during radioactive beta decay in the Earth.

test set Subset of the total dataset in machine learning that is kept aside for the entire training process. When model development and training are finished, this subset is used to evaluate the final model's performance on unseen data.

thorsten cut Collection of event selection criteria for high energy angular reconstruction of neutrinos inspired by the work of Thorsten Glüsenkamp, only using charged current (CC) events passing the muon filter.

tidal disruption event (TDE) Astronomical transient produced when a star is pulled apart by a black hole's tidal force.

token Individual element of the input sequence of a transformer model. In the case of language models, these are often words or subwords in a larger text context.

torch tensor Core data structure in the PyTorch library, a multi-dimensional array with added capabilities for GPU acceleration.

track Type of event detected in the IceCube detector caused by a ν_μ charged current (CC) interaction.

training loss Normalized empirical loss on the training set, used to update the model parameters during training of a machine learning model.

training set Subset of the total dataset used to train a machine learning model. The model is tasked with learning patterns in the training set, with the goal of those patterns generalizing to unseen data.

training The search for the optimal model weights in machine learning by iteratively updating them to minimize the loss.

transformer Machine learning architecture developed by Google researchers based on the attention mechanism.

trigger sim IceCube module used for the simulation of the event triggers.

validation loss Normalized empirical loss on the validation set, used as a measure of how well a machine learning model generalizes to unseen data during training.

validation set Subset of the total dataset in machine learning that is not directly used to train the model. Used to evaluate the model's performance on unseen data to tune hyperparameters during training.

vanishing gradient Problem during training of a machine learning model when gradients of the loss function become very small, especially problematic in deep models.

virtual particle Theoretical transient particle with characteristics of an ordinary particle, allowed to spontaneously emerge from vacuum.

vuvuzela IceCube module used for the generation of thermal and radioactive decay noise during simulation.

w boson An electrically charged particle with charge ± 1, mediating the charged current (CC) weak interactions..

waveform The analogue signal from the photomultiplier tube (PMT). When exceeding the discriminator threshold, it is digitized.

weak scaling Notion of scalability in high performance computing; how the solution time scales with the number of processors for a fixed problem size per processor.

weight decay Technique in machine learning used to prevent overfitting by adding a term to the loss function proportional to the square of the model weights.

z boson An electrically neutral particle mediating the neutral current (NC) weak interactions..

Bibliography

[1] Markus Ahlers and Francis Halzen. "Opening a new window onto the universe with IceCube". In: *Progress in Particle and Nuclear Physics* 102 (2018), pp. 73–88. ISSN: 0146-6410. DOI: https://doi.org/10.1016/j.ppnp.2018.05.001. URL: https://www.sciencedirect.com/science/article/pii/S0146641018300346 (cit. on pp. 1, 10, 11, 19).

[2] "Evidence for High-Energy Extraterrestrial Neutrinos at the IceCube Detector". In: *Science* 342.6161 (Nov. 2013). ISSN: 1095-9203. DOI: 10.1126/science.1242856. URL: http://dx.doi.org/10.1126/science.1242856 (cit. on p. 1).

[3] Mark Aartsen et al. "Multimessenger observations of a flaring blazar coincident with high-energy neutrino IceCube-170922A". In: *Science* 361.6398 (July 2018). ISSN: 1095-9203. DOI: 10.1126/science.aat1378. URL: http://dx.doi.org/10.1126/science.aat1378 (cit. on pp. 1, 11).

[4] R. Abbasi et al. "Evidence for neutrino emission from the nearby active galaxy NGC 1068". In: *Science* 378.6619 (Nov. 2022), pp. 538–543. ISSN: 1095-9203. DOI: 10.1126/science.abg3395. URL: http://dx.doi.org/10.1126/science.abg3395 (cit. on pp. 1, 11).

[5] R. Abbasi et al. "Observation of high-energy neutrinos from the Galactic plane". In: *Science* 380.6652 (June 2023), pp. 1338–1343. ISSN: 1095-9203. DOI: 10.1126/science.adc9818. URL: http://dx.doi.org/10.1126/science.adc9818 (cit. on pp. 1, 24).

[6] Robert Stein et al. "A tidal disruption event coincident with a high-energy neutrino". In: *Nature Astronomy* 5.5 (2021), pp. 510–518. ISSN: 2397-3366. DOI: 10.1038/s41550-020-01295-8. URL: https://doi.org/10.1038/s41550-020-01295-8 (cit. on p. 1).

[7] Simeon Reusch et al. "Candidate Tidal Disruption Event AT2019fdr Coincident with a High-Energy Neutrino". In: *Phys. Rev. Lett.* 128 (22 June 2022), p. 221101. DOI: 10.1103/PhysRevLett.128.221101. URL: https://link.aps.org/doi/10.1103/PhysRevLett.128.221101 (cit. on p. 1).

[8] M. G. Aartsen et al. "Time-Integrated Neutrino Source Searches with 10 Years of IceCube Data". In: *Physical Review Letters* 124.5 (Feb. 2020). ISSN: 1079-7114. DOI: 10.1103/physrevlett.124.051103. URL: http://dx.doi.org/10.1103/PhysRevLett.124.051103 (cit. on p. 1).

[9] R. Abbasi et al. "A muon-track reconstruction exploiting stochastic losses for large-scale Cherenkov detectors". In: *Journal of Instrumentation* 16.08 (Aug. 2021), P08034. ISSN: 1748-0221. DOI: 10.1088/1748-0221/16/08/p08034. URL: http://dx.doi.org/10.1088/1748-0221/16/08/P08034 (cit. on pp. 1, 2, 39, 40).

[10] M.G. Aartsen et al. "Improvement in fast particle track reconstruction with robust statistics". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 736 (Feb. 2014), pp. 143–149. ISSN: 0168-9002. DOI: 10.1016/j.nima.2013.10.074. URL: http://dx.doi.org/10.1016/j.nima.2013.10.074 (cit. on pp. 2, 38).

[11] R. Abbasi et al. "Graph Neural Networks for low-energy event classification reconstruction in IceCube". In: *Journal of Instrumentation* 17.11 (Nov. 2022), P11003. DOI: 10.1088/1748-0221/17/11/P11003. URL: https://dx.doi.org/10.1088/1748-0221/17/11/P11003 (cit. on pp. 2, 36).

[12] Habib Bukhari et al. *IceCube – Neutrinos in Deep Ice The Top 3 Solutions from the Public Kaggle Competition*. 2023. arXiv: 2310.15674 [astro-ph.HE]. URL: https://arxiv.org/abs/2310.15674 (cit. on pp. 2, 36, 45, 54).

[13] Philipp Eller. *Public Kaggle Competition "IceCube – Neutrinos in Deep Ice"*. 2023. arXiv: 2307.15289 [astro-ph.HE]. URL: https://arxiv.org/abs/2307.15289 (cit. on pp. 2, 36).

[14] J. J. Thomson. "XL. Cathode Rays". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 44.269 (1897), pp. 293–316. DOI: 10.1080/14786449708621070. URL: https://doi.org/10.1080/14786449708621070 (cit. on p. 4).

[15] Albert Einstein. "Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt". In: *Annalen der Physik* 322.6 (1905), pp. 132–148. DOI: 10.1002/andp.19053220607. URL: https://doi.org/10.1002/andp.19053220607 (cit. on p. 4).

[16] R. A. Millikan. "A Direct Photoelectric Determination of Planck's "h"". In: *Phys. Rev.* 7 (3 Mar. 1916), pp. 355–388. DOI: 10.1103/PhysRev.7.355. URL: https://link.aps.org/doi/10.1103/PhysRev.7.355 (cit. on p. 4).

[17] C. M. G. Lattes et al. "Processes Involving Charged Mesons". In: *Nature* 159.4047 (1947), pp. 694–697. ISSN: 1476-4687. DOI: 10.1038/159694a0. URL: https://doi.org/10.1038/159694a0 (cit. on p. 4).

[18] Seth H. Neddermeyer and Carl D. Anderson. "Note on the Nature of Cosmic-Ray Particles". In: *Phys. Rev.* 51 (10 May 1937), pp. 884–886. DOI: 10.1103/PhysRev.51.884. URL: https://link.aps.org/doi/10.1103/PhysRev.51.884 (cit. on p. 4).

[19] S. Navas et al. "Review of Particle Physics". In: *Phys. Rev. D* 110 (3 Aug. 2024), p. 030001. DOI: 10.1103/PhysRevD.110.030001. URL: https://link.aps.org/doi/10.1103/PhysRevD.110.030001 (cit. on pp. 4, 5).

[20] F. Englert and R. Brout. "Broken Symmetry and the Mass of Gauge Vector Mesons". In: *Phys. Rev. Lett.* 13 (9 Aug. 1964), pp. 321–323. DOI: 10.1103/PhysRevLett.13.321. URL: https://link.aps.org/doi/10.1103/PhysRevLett.13.321 (cit. on p. 4).

[21] Peter W. Higgs. "Broken Symmetries and the Masses of Gauge Bosons". In: *Phys. Rev. Lett.* 13 (16 Oct. 1964), pp. 508–509. DOI: 10.1103/PhysRevLett.13.508. URL: https://link.aps.org/doi/10.1103/PhysRevLett.13.508 (cit. on p. 4).

[22] G. S. Guralnik, C. R. Hagen and T. W. B. Kibble. "Global Conservation Laws and Massless Particles". In: *Phys. Rev. Lett.* 13 (20 Nov. 1964), pp. 585–587. DOI: 10.1103/PhysRevLett.13.585. URL: https://link.aps.org/doi/10.1103/PhysRevLett.13.585 (cit. on p. 4).

[23] Ziro Maki, Masami Nakagawa and Shoichi Sakata. "Remarks on the Unified Model of Elementary Particles". In: *Progress of Theoretical Physics* 28.5 (Nov. 1962), pp. 870–880. ISSN: 0033-068X. DOI: 10.1143/PTP.28.870. eprint: https://academic.oup.com/ptp/article-pdf/28/5/870/5258750/28-5-870.pdf. URL: https://doi.org/10.1143/PTP.28.870 (cit. on pp. 5, 7).

[24] Dan Hooper. "What is the Standard Model of particle physics, and why are scientists looking beyond it?" In: *Astronomy Magazine* (2022). Accessed: 2025-02-23. URL: https://www.astronomy.com/science/what-is-the-standard-model-of-particle-physics-and-why-are-scientists-looking-beyond-it/ (cit. on p. 5).

[25] CERN. *The matter-antimatter asymmetry problem*. Accessed: 2025-02-23. 2025. URL: https://home.cern/science/physics/matter-antimatter-asymmetry-problem (cit. on p. 5).

[26] M.C. Gonzalez-Garcia and Michele Maltoni. "Phenomenology with massive neutrinos". In: *Physics Reports* 460.1–3 (Apr. 2008), pp. 1–129. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2007.12.004. URL: http://dx.doi.org/10.1016/j.physrep.2007.12.004 (cit. on p. 5).

[27] C. Giganti, S. Lavignac and M. Zito. "Neutrino oscillations: The rise of the PMNS paradigm". In: *Progress in Particle and Nuclear Physics* 98 (Jan. 2018), pp. 1–54. ISSN: 0146-6410. DOI: 10.1016/j.ppnp.2017.10.001. URL: http://dx.doi.org/10.1016/j.ppnp.2017.10.001 (cit. on p. 5).

[28] Nobel Prize Outreach. *The Nobel Prize in Physics 2015*. Accessed: 2025-02-23. 2025. URL: https://www.nobelprize.org/prizes/physics/2015/summary/ (cit. on p. 5).

[29] David J. Griffiths. *Introduction to Elementary Particles*. 2nd ed. See Section 1.5 for details. Weinheim, Germany: Wiley-VCH, 2008. ISBN: 978-3-527-40601-2 (cit. on p. 6).

[30] Laurie M. Brown. "The idea of the neutrino". In: *Physics Today* 31.9 (Sept. 1978), pp. 23–28. ISSN: 0031-9228. DOI: 10.1063/1.2995181. URL: https://doi.org/10.1063/1.2995181 (cit. on p. 6).

[31] C. L. Cowan et al. "Detection of the Free Neutrino: a Confirmation". In: *Science* 124.3212 (1956), pp. 103–104. DOI: 10.1126/science.124.3212.103. eprint: https://www.science.org/doi/pdf/10.1126/science.124.3212.103. URL: https://www.science.org/doi/abs/10.1126/science.124.3212.103 (cit. on p. 6).

[32] Gui-Jun Ding and Stephen F King. "Neutrino mass and mixing with modular symmetry". In: *Reports on Progress in Physics* 87.8 (July 2024), p. 084201. DOI: 10.1088/1361-6633/ad52a3. URL: https://dx.doi.org/10.1088/1361-6633/ad52a3 (cit. on p. 7).

[33] Ivan Esteban et al. "NuFit-6.0: updated global analysis of three-flavor neutrino oscillations". In: *Journal of High Energy Physics* 2024.12 (Dec. 2024). ISSN: 1029-8479. DOI: 10.1007/jhep12(2024)216. URL: http://dx.doi.org/10.1007/JHEP12(2024)216 (cit. on p. 7).

[34] Stephen F. King, Danny Marfatia and Moinul Hossain Rahat. "Toward distinguishing Dirac from Majorana neutrino mass with gravitational waves". In: *Phys. Rev. D* 109 (3 Feb. 2024), p. 035014. DOI: 10.1103/PhysRevD.109.035014. URL: https://link.aps.org/doi/10.1103/PhysRevD.109.035014 (cit. on p. 8).

[35] A S Barabash. "Double beta decay experiments: present and future". In: *Journal of Physics: Conference Series* 1390.1 (Nov. 2019), p. 012048. DOI: 10.1088/1742-6596/1390/1/012048. URL: https://dx.doi.org/10.1088/1742-6596/1390/1/012048 (cit. on p. 8).

[36] Kevork N. Abazajian and Julian Heeck. "Observing Dirac neutrinos in the cosmic microwave background". In: *Phys. Rev. D* 100 (7 Oct. 2019), p. 075027. DOI: 10.1103/PhysRevD.100.075027. URL: https://link.aps.org/doi/10.1103/PhysRevD.100.075027 (cit. on p. 8).

[37] J. A. Formaggio and G. P. Zeller. "From eV to EeV: Neutrino cross sections across energy scales". In: *Reviews of Modern Physics* 84.3 (Sept. 2012), pp. 1307–1341. ISSN: 1539-0756. DOI: 10.1103/revmodphys.84.1307. URL: http://dx.doi.org/10.1103/RevModPhys.84.1307 (cit. on pp. 8, 12).

[38] Abhik Jash. "Studies on the Physics of Resistive Plate Chambers in Relation to the INO Experiment". PhD thesis. HOMI BHABHA NATIONAL INSTITUTE, Apr. 2018, pp. 4–7 (cit. on pp. 8, 9).

[39] S Al Kharusi et al. "SNEWS 2.0: a next-generation supernova early warning system for multi-messenger astronomy". In: *New Journal of Physics* 23.3 (Mar. 2021), p. 031201. DOI: 10.1088/1367-2630/abde33. URL: https://dx.doi.org/10.1088/1367-2630/abde33 (cit. on p. 8).

[40] T. Araki et al. "Experimental investigation of geologically produced antineutrinos with KamLAND". In: *Nature* 436.7050 (2005), pp. 499–503. ISSN: 1476-4687. DOI: 10.1038/nature03980. URL: https://doi.org/10.1038/nature03980 (cit. on p. 8).

[41] S. Aiello et al. "Observation of an ultra-high-energy cosmic neutrino with KM3NeT". In: *Nature* 638.8050 (2025), pp. 376–382. DOI: 10.1038/s41586-024-07194-5 (cit. on p. 8).

[42] Kiran Munawar. "Identifying Cosmic Ray Induced Cascade Events with IceTop". MA thesis. Christchurch, New Zealand: University of Canterbury, Dec. 2017, pp. 28–32. URL: https://ir.canterbury.ac.nz/server/api/core/bitstreams/519f9e02-b979-4c8c-94ff-ae7502dc884a/content (cit. on pp. 9, 10).

[43] Yasaman Farzan and Alexei Yu. Smirnov. "Coherence and oscillations of cosmic neutrinos". In: *Nuclear Physics B* 805.1–2 (Dec. 2008), pp. 356–376. ISSN: 0550-3213. DOI: 10.1016/j.nuclphysb.2008.07.028. URL: http://dx.doi.org/10.1016/j.nuclphysb.2008.07.028 (cit. on p. 10).

[44] M. G. Aartsen et al. “Observation of High-Energy Astrophysical Neutrinos in Three Years of IceCube Data”. In: *Phys. Rev. Lett.* 113 (10 Sept. 2014), p. 101101. DOI: 10.1103/PhysRevLett.113.101101. URL: https://link.aps.org/doi/10.1103/PhysRevLett.113.101101 (cit. on p. 10).

[45] Robert Stein et al. “A tidal disruption event coincident with a high-energy neutrino”. In: *Nature Astronomy* 5.5 (Feb. 2021), pp. 510–518. ISSN: 2397-3366. DOI: 10.1038/s41550-020-01295-8. URL: http://dx.doi.org/10.1038/s41550-020-01295-8 (cit. on p. 11).

[46] Simeon Reusch et al. “Candidate Tidal Disruption Event AT2019fdr Coincident with a High-Energy Neutrino”. In: *Physical Review Letters* 128.22 (June 2022). ISSN: 1079-7114. DOI: 10.1103/physrevlett.128.221101. URL: http://dx.doi.org/10.1103/PhysRevLett.128.221101 (cit. on p. 11).

[47] Kate Scholberg. *Supernova Neutrinos: Astrophysics and Particle Physics*. Presented at the Niels Bohr International Academy, University of Copenhagen. Accessed: 2025-02-23. June 2023. URL: https://indico.nbi.ku.dk/event/1046/contributions/7626/attachments/2618/3812/KateScholberg1.pdf (cit. on p. 11).

[48] Raj Gandhi et al. “Ultrahigh-energy neutrino interactions”. In: *Astroparticle Physics* 5.2 (Aug. 1996), pp. 81–110. ISSN: 0927-6505. DOI: 10.1016/0927-6505(96)00008-4. URL: http://dx.doi.org/10.1016/0927-6505(96)00008-4 (cit. on p. 12).

[49] Sheldon L. Glashow. “Resonant Scattering of Antineutrinos”. In: *Phys. Rev.* 118 (1 Apr. 1960), pp. 316–317. DOI: 10.1103/PhysRev.118.316. URL: https://link.aps.org/doi/10.1103/PhysRev.118.316 (cit. on p. 12).

[50] Kai Schatto. “Stacked searches for high-energy neutrinos from blazars with IceCube”. PhD thesis. Mainz U., June 2014 (cit. on pp. 13–15, 17–19, 22, 37–40).

[51] Juliana Stachurska. *First Double Cascade Tau Neutrino Candidates in IceCube and a New Measurement of the Flavor Composition*. 2019. arXiv: 1908.05506 [astro-ph.HE]. URL: https://arxiv.org/abs/1908.05506 (cit. on p. 13).

[52] John Learned and Kai Mannheim. “High-Energy Neutrino Astrophysics”. In: *Annual Review of Nuclear and Particle Science* 50 (Dec. 2000), pp. 679–749. DOI: 10.1146/annurev.nucl.50.1.679 (cit. on p. 13).

[53] Juan Pablo Yáñez Garza. “Measurement of neutrino oscillations in atmospheric neutrinos with the IceCube DeepCore detector”. PhD thesis. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät I, 2014. DOI: http://dx.doi.org/10.18452/17016 (cit. on p. 13).

[54] P. A. Čerenkov. “Visible Radiation Produced by Electrons Moving in a Medium with Velocities Exceeding that of Light”. In: *Phys. Rev.* 52 (4 Aug. 1937), pp. 378–379. DOI: 10.1103/PhysRev.52.378. URL: https://link.aps.org/doi/10.1103/PhysRev.52.378 (cit. on p. 14).

[55] J. Ahrens et al. “Muon track reconstruction and data selection techniques in AMANDA”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 524.1–3 (May 2004), pp. 169–194. ISSN: 0168-9002. DOI: 10.1016/j.nima.2004.01.065. URL: http://dx.doi.org/10.1016/j.nima.2004.01.065 (cit. on p. 14).

[56] Christian Spiering. “Towards high-energy neutrino astronomy: A historical review”. In: *The European Physical Journal H* 37.3 (July 2012), pp. 515–565. ISSN: 2102-6467. DOI: 10.1140/epjh/e2012-30014-2. URL: http://dx.doi.org/10.1140/epjh/e2012-30014-2 (cit. on p. 15).

[57] IceCube Collaboration. *Measurement of atmospheric neutrino oscillation parameters using convolutional neural networks with 9.3 years of data in IceCube DeepCore*. 2024. arXiv: 2405.02163 [hep-ex]. URL: https://arxiv.org/abs/2405.02163 (cit. on p. 15).

[58] R. Abbasi et al. "Search for an eV-Scale Sterile Neutrino Using Improved High-Energy ν_μ Event Reconstruction in IceCube". In: *Physical Review Letters* 133.20 (Nov. 2024). ISSN: 1079-7114. DOI: 10.1103/physrevlett.133.201804. URL: http://dx.doi.org/10.1103/PhysRevLett.133.201804 (cit. on p. 15).

[59] The IceCube Collaboration. *Search for dark matter from the center of the Earth with ten years of IceCube data*. 2024. arXiv: 2412.12972 [astro-ph.HE]. URL: https://arxiv.org/abs/2412.12972 (cit. on p. 15).

[60] Donghwa Kang. "Recent results of cosmic-ray studies with IceTop at the IceCube Neutrino Observatory". In: *Advances in Space Research* 72.10 (2023), pp. 4613–4624. ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2023.09.027. URL: https://www.sciencedirect.com/science/article/pii/S0273117723007494 (cit. on p. 15).

[61] Martin Rongen. *The IceCube Neutrino Observatory as an Instrument for Glaciology*. Seminar presented at VUB IIHE. Accessed: 2025-02-23. Mar. 2018. URL: https://indico.iihe.ac.be/event/1169/attachments/251/268/IceCubeGlaciology.pdf (cit. on p. 15).

[62] Markus Ahlers, Klaus Helbing and Carlos Pérez de los Heros. "Probing particle physics with IceCube". In: *The European Physical Journal C* 78.11 (2018), p. 924. ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-018-6369-9. URL: https://doi.org/10.1140/epjc/s10052-018-6369-9 (cit. on pp. 15, 16).

[63] R. Abbasi et al. "IceTop: The surface component of IceCube". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 700 (Feb. 2013), pp. 188–220. ISSN: 0168-9002. DOI: 10.1016/j.nima.2012.10.067. URL: http://dx.doi.org/10.1016/j.nima.2012.10.067 (cit. on p. 15).

[64] Aya Ishihara. *The IceCube Upgrade – Design and Science Goals*. 2019. arXiv: 1908.09441 [astro-ph.HE]. URL: https://arxiv.org/abs/1908.09441 (cit. on p. 15).

[65] M G Aartsen et al. "IceCube-Gen2: the window to the extreme Universe". In: *Journal of Physics G: Nuclear and Particle Physics* 48.6 (Apr. 2021), p. 060501. ISSN: 1361-6471. DOI: 10.1088/1361-6471/abbd48. URL: http://dx.doi.org/10.1088/1361-6471/abbd48 (cit. on p. 15).

[66] Spencer R. Klein. "IceCube: A Cubic Kilometer Radiation Detector". In: *IEEE Transactions on Nuclear Science* 56.3 (June 2009), pp. 1141–1147. ISSN: 0018-9499. DOI: 10.1109/tns.2009.2015300. URL: http://dx.doi.org/10.1109/TNS.2009.2015300 (cit. on p. 17).

[67] R. Abbasi et al. "The design and performance of IceCube DeepCore". In: *Astroparticle Physics* 35.10 (May 2012), pp. 615–624. ISSN: 0927-6505. DOI: 10.1016/j.astropartphys.2012.01.004. URL: http://dx.doi.org/10.1016/j.astropartphys.2012.01.004 (cit. on p. 17).

[68] R. Abbasi et al. "Calibration and characterization of the IceCube photomultiplier tube". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 618.1–3 (June 2010), pp. 139–152. ISSN: 0168-9002. DOI: 10.1016/j.nima.2010.03.102. URL: http://dx.doi.org/10.1016/j.nima.2010.03.102 (cit. on p. 17).

[69] Martin Rongen and The IceCube Collaboration. *Calibration of the IceCube Neutrino Observatory*. Presented at the Workshop on the Evaluation of Advanced Electronics and Instrumentation for Water Cherenkov Experiments. Apr. 2022. URL: https://indico.in2p3.fr/event/26412/contributions/107686/attachments/69882/98866/IceCube_Calibration_WCW22.pdf (cit. on pp. 18, 23).

[70] J. Lundberg et al. "Light tracking through ice and water—Scattering and absorption in heterogeneous media with Photonics". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 581.3 (Nov. 2007), pp. 619–631. ISSN: 0168-9002. DOI: 10.1016/j.nima.2007.07.143. URL: http://dx.doi.org/10.1016/j.nima.2007.07.143 (cit. on pp. 18, 23).

[71] M. Aartsen et al. "South Pole Glacial Climate Reconstruction from Multi-Borehole Laser Particulate Stratigraphy". In: *Journal of Glaciology* 59 (Sept. 2013), pp. 1117–1129. DOI: 10.3189/2013JoG13J068 (cit. on pp. 18, 23).

[72] Étienne Bourbeau. *The Ice in IceCube: A Historical Journey*. Presented at the University of Copenhagen. Nov. 2017. URL: https://wiki.icecube.wisc.edu/images/e/e2/Ice_summary.pdf (cit. on p. 18).

[73] Ryan Bay, Robert Rohde and Nathan Bramall. "South Pole paleowind from automated synthesis of ice core records". In: *J. Geophys. Res* 115 (July 2010). DOI: 10.1029/2009JD013741 (cit. on p. 18).

[74] The IceCube Collaboration. "Evidence of Optical Anisotropy of the South Pole Ice". In: *Proceedings of the International Cosmic Ray Conference*. ICRC, 2013. URL: https://arxiv.org/pdf/1309.7010 (cit. on p. 18).

[75] M.G. Aartsen et al. "Measurement of South Pole ice transparency with the IceCube LED calibration system". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 711 (May 2013), pp. 73–89. ISSN: 0168-9002. DOI: 10.1016/j.nima.2013.01.054. URL: http://dx.doi.org/10.1016/j.nima.2013.01.054 (cit. on p. 18).

[76] Dmitry Chirkin and Martin Rongen. *An improved mapping of ice layer undulations for the IceCube Neutrino Observatory*. 2023. arXiv: 2307.13951 [astro-ph.HE]. URL: https://arxiv.org/abs/2307.13951 (cit. on pp. 18, 23, 42).

[77] R. Abbasi et al. "IceCube high-energy starting event sample: Description and flux characterization with 7.5 years of data". In: *Physical Review D* 104.2 (July 2021). ISSN: 2470-0029. DOI: 10.1103/physrevd.104.022002. URL: http://dx.doi.org/10.1103/PhysRevD.104.022002 (cit. on pp. 19, 76).

[78] J. L. Kelley and IceCube Collaboration. "Event triggering in the IceCube data acquisition system". In: *AIP Conference Proceedings* 1630.1 (Nov. 2014), pp. 154–157. ISSN: 0094-243X. DOI: 10.1063/1.4902795. eprint: https://pubs.aip.org/aip/acp/article-pdf/1630/1/154/12124858/154_1_online.pdf. URL: https://doi.org/10.1063/1.4902795 (cit. on p. 19).

[79] Nathan Whitehorn. "A Search for High-Energy Neutrino Emission from Gamma-Ray Bursts". Ph.D. Dissertation. University of Wisconsin–Madison, 2012. URL: https://docushare.icecube.wisc.edu/dsweb/Get/Document-60879/thesis.pdf (cit. on pp. 19, 20).

[80] John Kelley. *The IceCube Detector, Part II: DAQ, Triggers, and Filters*. IceCube Virtual Bootcamp, 2020-06-15. With thanks to Dave Glowacki, Naoko K. Neilson, Erik Blaufuss. 2020. URL: https://events.icecube.wisc.edu/event/123/contributions/6461/attachments/5474/6272/Bootcamp-Detector-2020_v2.pdf (cit. on p. 21).

[81] M.G. Aartsen et al. "The IceCube Neutrino Observatory: instrumentation and online systems". In: *Journal of Instrumentation* 12.03 (Mar. 2017), P03012–P03012. ISSN: 1748-0221. DOI: 10.1088/1748-0221/12/03/p03012. URL: http://dx.doi.org/10.1088/1748-0221/12/03/P03012 (cit. on p. 21).

[82] Jorge Prado González. "Machine Learning-Driven Event Selection for Neutrino Physics Analysis Using the IceCube Upgrade". MA thesis. University of Copenhagen, Faculty of Science, May 2023. URL: https://nbi.ku.dk/english/research/experimental-particle-physics/icecube/group_theses/JorgeGonzalezMScThesis_UpgradeEventSelection_.pdf (cit. on pp. 21, 22).

[83] Juan Carlos Díaz-Vélez. *Intro to IceCube Simulation*. IceProd/SimProd Workshop, Madison, WI, USA. June 2024. URL: https://events.icecube.wisc.edu/event/227/contributions/10230/attachments/7877/10252/Simulation%20-%20IceProd.pdf (cit. on pp. 22, 23).

[84] S. Iyer Dutta et al. "Propagation of muons and taus at high energies". In: *Phys. Rev. D* 63 (9 Apr. 2001), p. 094020. DOI: 10.1103/PhysRevD.63.094020. URL: https://link.aps.org/doi/10.1103/PhysRevD.63.094020 (cit. on p. 23).

[85] R. Abbasi et al. "Measurement of the high-energy all-flavor neutrino-nucleon cross section with IceCube". In: *Physical Review D* 104.2 (July 2021). ISSN: 2470-0029. DOI: 10.1103/physrevd.104.022001. URL: http://dx.doi.org/10.1103/PhysRevD.104.022001 (cit. on p. 23).

[86] J.-H. Koehne et al. "PROPOSAL: A tool for propagation of charged leptons". In: *Computer Physics Communications* 184.9 (2013), pp. 2070–2090. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2013.04.001. URL: https://www.sciencedirect.com/science/article/pii/S0010465513001355 (cit. on p. 23).

[87] R. Abbasi et al. "Calibration and characterization of the IceCube photomultiplier tube". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 618.1 (2010), pp. 139–152. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2010.03.102. URL: https://www.sciencedirect.com/science/article/pii/S0168900210006662 (cit. on p. 23).

[88] D. Tosi and C. Wendt for the IceCube Collaboration. "Calibrating the photon detection efficiency in IceCube". In: *Technology and Instrumentation in Particle Physics 2014 (TIPP2014)*. Amsterdam, the Netherlands: Proceedings of Science, June 2014. URL: https://pos.sissa.it/213/157/pdf (cit. on p. 23).

[89] M.G. Aartsen et al. "Efficient propagation of systematic uncertainties from calibration to analysis with the SnowStorm method in IceCube". In: *Journal of Cosmology and Astroparticle Physics* 2019.10 (Oct. 2019), pp. 048–048. ISSN: 1475-7516. DOI: 10.1088/1475-7516/2019/10/048. URL: http://dx.doi.org/10.1088/1475-7516/2019/10/048 (cit. on pp. 23, 42).

[90] Erik Ganster. *SnowStorm: A Summary of the Status of SnowStorm Simulations for a GlobalFit of IceCube's Neutrino Data*. Diffuse GlobalFit Workshop 2021, Madison (virtual). May 2021. URL: https://events.icecube.wisc.edu/event/137/contributions/7643/attachments/6056/7234/SnowStorm_Simulations.pdf (cit. on p. 23).

[91] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210 (cit. on p. 24).

[92] Nobel Prize Outreach AB. *Scientific Background to the Nobel Prize in Physics 2024: "For foundational discoveries and inventions that enable machine learning with artificial neural networks."*. https://www.nobelprize.org/prizes/physics/2024/advanced-information/. The Nobel Prize in Physics 2024: Advanced Information. 2024. URL: https://www.nobelprize.org/prizes/physics/2024/advanced-information/ (cit. on p. 24).

[93] G. Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1 (Sept. 2012), pp. 1–29. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.020. URL: http://dx.doi.org/10.1016/j.physletb.2012.08.020 (cit. on p. 24).

[94] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 303–314. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274 (cit. on p. 26).

[95] Matus Telgarsky. *Representation Benefits of Deep Feedforward Networks*. 2015. arXiv: 1509.08101 [cs.LG]. URL: https://arxiv.org/abs/1509.08101 (cit. on p. 27).

[96] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0 (cit. on p. 29).

[97] John Duchi, Elad Hazan and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12 (July 2011), pp. 2121–2159 (cit. on p. 29).

[98] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980 (cit. on p. 29).

[99] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG]. URL: https://arxiv.org/abs/1609.04747 (cit. on p. 29).

[100] Alexander Amini et al. *Spatial Uncertainty Sampling for End-to-End Control*. May 2018. DOI: 10.48550/arXiv.1805.04829 (cit. on p. 29).

[101] Olga Mula and Mark Peletier. *Mathematics of Neural Networks*. Lecture notes, Master course, Department of Mathematics, TU Eindhoven. Last edited on Saturday, 16th December 2023. Dec. 2023 (cit. on p. 29).

[102] Andrew L. Maas. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: 2013. URL: https://api.semanticscholar.org/CorpusID:16489696 (cit. on p. 30).

[103] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762 (cit. on pp. 32–35, 50, 62, 65).

[104] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 32).

[105] Nal Kalchbrenner et al. *Neural Machine Translation in Linear Time*. 2017. arXiv: 1610.10099 [cs.CL]. URL: https://arxiv.org/abs/1610.10099 (cit. on p. 32).

[106] Fakultit Informatik et al. "Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies". In: *A Field Guide to Dynamical Recurrent Neural Networks* (Mar. 2003) (cit. on p. 32).

[107] Josep Ferrer. *How Transformers Work: A Detailed Exploration of Transformer Architecture*. Jan. 2024. URL: https://www.datacamp.com/tutorial/how-transformers-work (cit. on pp. 33, 34).

[108] Andreas Søgaard et al. *GraphNeT: Graph neural networks for neutrino telescope event reconstruction*. 2022. arXiv: 2210.12194 [astro-ph.IM]. URL: https://arxiv.org/abs/2210.12194 (cit. on p. 36).

[109] R. Abbasi et al. "A convolutional neural network based cascade reconstruction for the IceCube Neutrino Observatory". In: *Journal of Instrumentation* 16.07 (July 2021), P07041. ISSN: 1748-0221. DOI: 10.1088/1748-0221/16/07/p07041. URL: http://dx.doi.org/10.1088/1748-0221/16/07/P07041 (cit. on p. 37).

[110] Rasha Abbasi et al. "Conditional normalizing flows for IceCube event reconstruction". In: *Proceedings of 38th International Cosmic Ray Conference — PoS(ICRC2023)*. Vol. 444. 2023, p. 1003. DOI: 10.22323/1.444.1003 (cit. on p. 37).

[111] Mirco Huennefeld et al. "Combining Maximum-Likelihood with Deep Learning for Event Reconstruction in IceCube". In: *Proceedings of 37th International Cosmic Ray Conference — PoS(ICRC2021)*. Vol. 395. 2021, p. 1065. DOI: 10.22323/1.395.1065 (cit. on p. 37).

[112] J. Ahrens et al. "Muon track reconstruction and data selection techniques in AMANDA". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 524.1–3 (May 2004), pp. 169–194. ISSN: 0168-9002. DOI: 10.1016/j.nima.2004.01.065. URL: http://dx.doi.org/10.1016/j.nima.2004.01.065 (cit. on pp. 37–39).

[113] Christian Spiering. "Neutrino Detectors Under Water and Ice". In: *Particle Physics Reference Library: Volume 2: Detectors for Particles and Radiation*. Ed. by Christian W. Fabjan and Herwig Schopper. Cham: Springer International Publishing, 2020, pp. 785–822. ISBN: 978-3-030-35318-6. DOI: 10.1007/978-3-030-35318-6_17. URL: https://doi.org/10.1007/978-3-030-35318-6_17 (cit. on p. 39).

[114] Haack, Christian, Lu, Lu and Yuan, Tianlu. "Improving the directional reconstruction of PeV hadronic cascades in IceCube". In: *EPJ Web Conf.* 207 (2019), p. 05003. DOI: 10.1051/epjconf/201920705003. URL: https://doi.org/10.1051/epjconf/201920705003 (cit. on p. 40).

[115] R. Abbasi et al. "A convolutional neural network based cascade reconstruction for the IceCube Neutrino Observatory". In: *Journal of Instrumentation* 16.07 (July 2021), P07041. ISSN: 1748-0221. DOI: 10.1088/1748-0221/16/07/p07041. URL: http://dx.doi.org/10.1088/1748-0221/16/07/P07041 (cit. on p. 40).

[116] R. Abbasi et al. "Improved modeling of in-ice particle showers for IceCube event reconstruction". In: *Journal of Instrumentation* 19.06 (June 2024), P06026. ISSN: 1748-0221. DOI: 10.1088/1748-0221/19/06/p06026. URL: http://dx.doi.org/10.1088/1748-0221/19/06/P06026 (cit. on p. 40).

[117] Apache Arrow Development Team. *PyArrow Documentation*. Accessed: 2025-03-27. 2025. URL: https://arrow.apache.org/docs/python/index.html (cit. on p. 48).

[118] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG]. URL: https://arxiv.org/abs/1912.01703 (cit. on p. 49).

[119] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org (cit. on p. 50).

[120] Tyler R. Scott, Andrew C. Gallagher and Michael C. Mozer. *von Mises-Fisher Loss: An Exploration of Embedding Geometries for Supervised Learning*. 2021. arXiv: 2103.15718 [cs.LG]. URL: https://arxiv.org/abs/2103.15718 (cit. on p. 54).

[121] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101 (cit. on p. 55).

[122] Sashank J. Reddi, Satyen Kale and Sanjiv Kumar. "On the Convergence of Adam and Beyond". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=ryQu7f-RZ (cit. on p. 55).

[123] R. Abbasi et al. "Low energy event reconstruction in IceCube DeepCore". In: *The European Physical Journal C* 82.9 (2022), p. 807. DOI: 10.1140/epjc/s10052-022-10721-2. URL: https://doi.org/10.1140/epjc/s10052-022-10721-2 (cit. on p. 59).

[124] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361 (cit. on pp. 62–65).

[125] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805 (cit. on p. 83).

[126] Thorsten Glüsenkamp. *Conditional normalizing flows for IceCube event reconstruction*. 2023. arXiv: 2309.16380 [astro-ph.HE]. URL: https://arxiv.org/abs/2309.16380 (cit. on p. 84).

[127] Tri Dao. *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning*. 2023. arXiv: 2307.08691 [cs.LG]. URL: https://arxiv.org/abs/2307.08691 (cit. on p. 84).

[128] Meta AI. *xFormers: A modular and efficient library for Transformer models*. https://github.com/facebookresearch/xformers. Accessed: 2025-05-07. 2021 (cit. on p. 84).

[129] Saskia Philippen, Thorsten Glüsenkamp and Sebastian Schindler. *Testing the Pointing of IceCube using the Moon Shadow in Cosmic-Ray Induced Muons*. 2021. arXiv: 2108.04093 [astro-ph.HE]. URL: https://arxiv.org/abs/2108.04093 (cit. on p. 84).