Front page

Exam information
NFYK10020E - Physics Thesis 60 ECTS, Niels Bohr Institute - Contract:128712 (Kaare Endrup Iversen)

Handed in by
Kaare Endrup Iversen
nvc889@alumni.ku.dk

Exam administrators
Eksamensteam, tel 35 33 64 57
eksamen@science.ku.dk

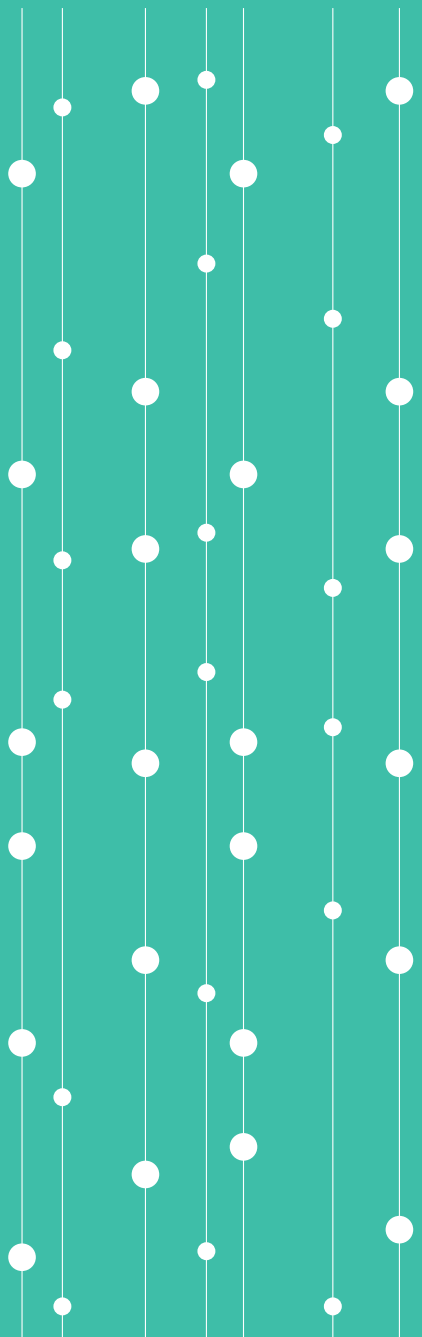
Assessors
Troels Christian Petersen
Examiner
petersen@nbi.ku.dk
+4535325442

Hand-in information

Title: The Good, the Bad & the Noisy - Characterisation of IceCube Neutrino Events Using Graph Neural Networks
Title, english: The Good, the Bad & the Noisy - Characterisation of IceCube Neutrino Events Using Graph Neural Networks
The sworn statement: Yes
Does the hand-in contain confidential material: No

The Good, the Bad & the Noisy

Characterisation of IceCube Neutrino Events Using Graph Neural Networks



Master's Thesis in Computational Physics

Written by *Kaare Endrup Iversen*

2021-2022

Supervised by
Troels Christian Petersen

University of Copenhagen

Name of Institute:	Faculty of Science
Name of Department:	The Niels Bohr Institute
Author(s):	Kaare Endrup Iversen
Email:	nvc889@alumni.ku.dk
Title and subtitle:	The Good, the Bad & the Noisy - Characterisation of IceCube Neutrino Events Using Graph Neural Networks
Supervisor(s):	Troels Christian Petersen
Handed in:	2021-2022
Defended:	2021-2022

The Good, the Bad & the Noisy

Master's Thesis in Computational Physics

First printing, May 2022

Abstract

The study of neutrino oscillations using low energy events from the IceCube detector is dependant on reliable event selection and accurate event reconstruction. The current methods are slow and computationally expensive and are also hard to update and improve upon.

Machine learning methods are well suited for many of these tasks, as they are fast and computationally inexpensive but require large amounts of training data, of which there is plenty in the IceCube experiment. While most machine learning methods struggle with the irregular IceCube data, graph neural networks (GNNs) – machine learning algorithms based on mathematical graph theory – hold especially great potential, as they take advantage of the sparse and non-Euclidean geometry of IceCube data and can tackle the great variation in signal input size.

In this thesis, methods based on GNNs are applied for a series of classification and reconstruction tasks on low energy IceCube events.

Reconstruction of the event interaction time is tested using different methods of transformations of the target variable, finding that a non-linear QuantileTransformer is necessary to obtain satisfying performance on the irregularly distributed variable. Separation of events by detector noise events and particle events and separation by track- and cascade-like events is also implemented. The classifiers are compared with the current event cleaning pipeline, finding that the GNN approach produces samples of higher purity with higher efficiency.

Pulse level cleaning of the events from the noisier IceCube Upgrade detector is demonstrated, showing that the sophisticated GNN outperforms competing methods and produces an event sample with satisfying purity and efficiency. The cleaned pulses are used for reconstruction of energy and direction, demonstrating that the pulse cleaning counteracts the increased noise rates in all regimes except very low energy events and some cascade-like events.

All these improvements should enable IceCube to take the lead in precision on the measurements of θ_{23} and Δm_{23} in the next half decade.

Acknowledgments

I would like to thank Troels Petersen, my supervisor for the opportunity to work on this project, for his ability to attract and gather many of the coolest and cleverest people I have ever had the pleasure of meeting, for his endless stream of ideas, and for never letting any of us forget that data is fun and exciting.

I would also like extend my thanks the members GraphNeT group, particularly Rasmus Ørsøe and Andreas Søgaard for the opportunity to use and work on the framework developed by them, but more so for their irreplaceable guidance and for always being willing and happy to help master students learn more about GNNs, coding practicies, github and so much more. And to the extended NBI IceCube family, Tanya, James, Jason, Aske, Morten and Amalie for good discussion, many laughs and for a work environment that made me look forward to going to the office every day. A special and heartfelt thanks to Tom Stuttard for making all of the IceCube Upgrade simulations used in this work.

In addition, I want to thank my family and my second family, the Troglodytes of Studentergården, as well as Astrid for their support and patience, for listening me to talk about graphs and neutrinos before disappearing to Blegdamsvej yet again.

And lastly, the biggest thanks to Leon Bozianu, the best unexpected office partner one could wish for, for the news reading, danish learning, scone consuming, Netto going, meme exchanging, countless of ours of banter, and the bit of physics that went on in between.

Statement of Contributions

This thesis is part of larger project with many working pieces and hard-working people. It is a product of collaboration on multiple scales from the entirety of the IceCube Collaboration, over the GraphNeT group spanning University of Copenhagen and Technical University of Munich, over the Niels Bohr Institute High Energy Physics group, to the middle office on the 2nd floor of the M building where Leon Bozianu and I have worked together for the past nine months (and Morten Holm for the last four). The work presented here could not exist without any of these groups.

A significant amount of my time working on this project has been spent contributing to the development of the GraphNeT framework. This work will not feature in this thesis as I have deemed it too code-specific and not relevant for the physics research and results published here. In the end, the hours spent working on the framework have been returned tenfold by its implementation in my work on this thesis and by the generous help and guidance of the member of the GraphNeT group.

The contents of the following chapters are the results of studies carried out be me, with offset in the work of others before me, and ideas that have been discussed within our group. The exceptions are as follows: The results in chapter 6, which started with common model development by Morten Holm and me, after which we carried on in our separate directions. The results in Chapter 5 which is is the product of a collaboration with Leon Bozianu which has lasted the entire time of our theses, resulting in models trained and used by both of us, but with completely separate final analyses.

Introduction

The goal of this thesis is to explore the possibilities of using Graph Neural Networks (GNNs) for reconstruction of low energy neutrino events in the IceCube detector. When neutrinos interact in the ice, their charged particle products produce Cherenkov radiation which is observed by the 5,160 Digital Optical Models (DOMs). Due to the irregular geometry of the IceCube detector array and the sparse nature of low-energy events, GNNs are believed to be suited for event analysis, and previous works have already obtained results that are superior to current methods for a variety of tasks. Additionally, with the current methods, reconstructing all events recorded by IceCube takes 3 months using 1000 CPUs. With GNNs, this can be done in 8 hours on a single CPU. Better and faster event reconstruction will allow for better understanding of neutrinos and their behavior beyond the Standard Model of Particle Physics, such as the oscillation parameters.

The first chapter will give a brief overview of the Standard Model, the history of neutrinos and the concept of neutrino oscillations, and an introduction to other phenomena important to IceCube.

The second chapter describes the IceCube experiment, the current and future detector array and the current reconstruction methods.

The third chapter introduces machine learning and GNNs, and also introduces the IceCube GNN group, the GraphNeT group.

The fourth chapter is the first chapter of results and analysis. Here, the pitfalls and possibilities of regressing the interaction time variable using feature scaling is examined.

The fifth chapter shows how GNNs can be used to replace parts or all of the OscNext event cleaning process.

The sixth chapter concerns the IceCube Upgrade detectors, and training a GNN to separate noise from physics on a pulse level to mitigate the increased noise associated with it.

In the seventh chapter, the cleaned Upgrade pulses from chapter six are used to perform reconstruction, in order to properly quantify the effect of the noise cleaning.

In the end, all of the above is discussed along with future ideas, dreams and challenges.

Contents

1 Challenges in Particle Physics

Contents

This thesis concerns the neutrinos, their fascinating behavior and the study of them through the IceCube detector. To preface these investigations, this sections provides and overview of the current landscape of particle physics and some of the answered questions that occupy particle physicist today, as well as an introduction to the phenomena that makes it possible to study of neutrinos through the IceCube detector array.

1.1 The Standard Model

Since the ancient Greeks and most likely earlier, scientists have sought to understand the fundamental building blocks of the world we live in. Through modern particle physics, a coherent description of matter and the interactions that govern its behavior is presented in the **Standard Model of Particle Physics**. In the Standard Model, the matter and forces of our universe are reduced to combinations of 17 particles with corresponding antiparticles and four fundamental forces. The particles are arranged in two main groups, the **fermions** which have a spin of $\frac{1}{2}$ and represent the building blocks of matter and the **bosons** which have integer spin numbers and mediate the interactions through the four fundamental forces.

The fermions are further categorised in two subgroups, known as the **quarks** which have an electric charge of either $\frac{2}{3}$ or $-\frac{1}{3}$[1] and bind together through the strong force to form hadrons like protons and neutrons (which are baryons) and pions and kaons (which are mesons) and the **leptons** which again consist of the 3 "electon-like" particles with an electric charge of -1, the **electron**, the **muon** and the **tauon**, and the neutrally charged **neutrinos** of which there exist one for each of the electron-like particles.

The three of the four fundamental forces are explained by the Standard model: The electromagnetic force mediated by the **photon**, the weak force mediated by the Z and $W^{\pm}$ bosons, and strong force mediated by the **gluon**. The final boson, the **Higgs boson** H^0 which represent the field that gives mass to all the particles[2] as part of the electroweak symmetry breaking. The fourth fundamental force, gravity is not included in the Standard Model[3].

All the matter fermions and force carrying bosons are shown in Figure 1.1, with the quarks in green, the leptons in blue, the gauge bosons in orange, and the Higgs in yellow. Thin outlines indicate which fermions interact through which forces.

[1] With the corresponding antiquarks having charges of $-\frac{2}{3}$ and $\frac{1}{3}$, respectively.

[2] Except the neutrinos, which will be discussed in Section 1.2.

[3] It is not yet known, if a corresponding particle (like a graviton) exists to mediate the gravitational force. If it was found, it would mean the discovery of quantum gravity.

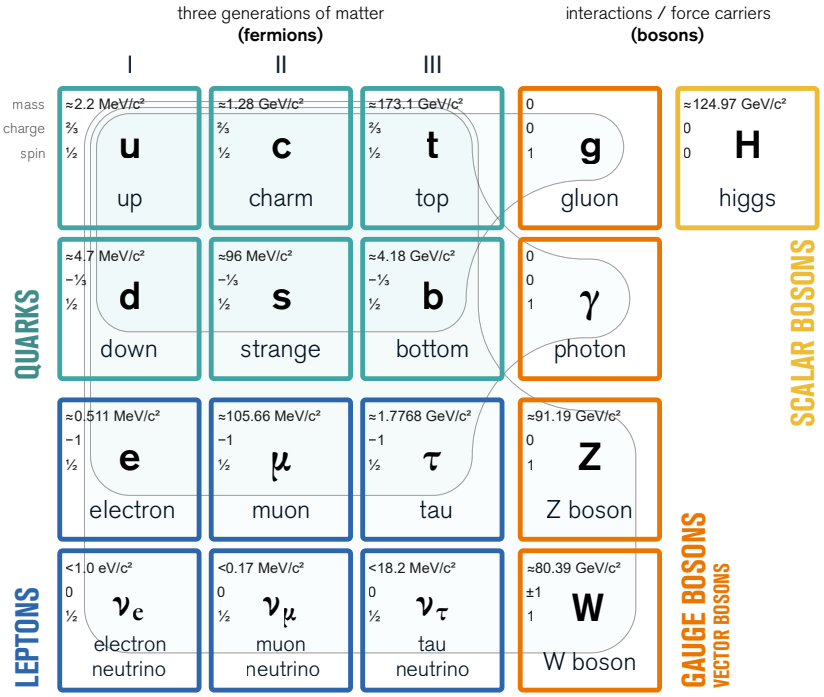


Figure 1.1: The Standard Model of Particle Physics.
Image made in Adobe Illustrator with inspiration from [1].

The particles of the lepton family only interact through the the electromagnetic force and the weak force, while the quacks also interact through the strong force[2]. This relationship is visualised in Figure

1.2 with blue lines connecting particles that interact with each other.

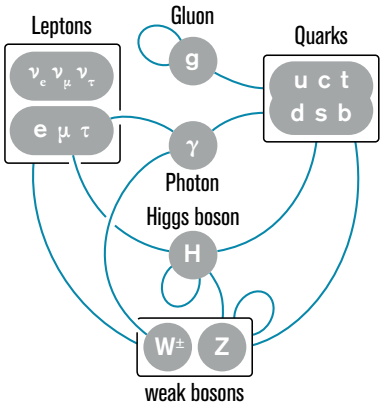


Figure 1.2: Schematic of the interactions of the particles of the Standard Model. *Image made in Adobe Illustrator with inspiration from* [3].

1.2 Neutrinos and Oscillations

Perhaps the strangest and most elusive group of the 17 fundamental particles are the neutrinos. Initially thought to be massless, these uncharged leptons interact exclusively through gravitation and the weak force, the first of which is very weak and the second of very short range. The weakness of their interactions makes them virtually undetectable despite them being among the most abundant particles in the universe. About 100 trillion solar neutrinos pass through our bodies every second.

Many neutrinos originate from nuclear fusion processes in the Sun, while others are the product of radioactive decays, and others still are created constantly in cosmic ray interactions in our atmosphere [4]. On Earth, high intensity neutrino beams are created and studied using particle accelerators. While the neutrinos' unwillingness to interact makes them hard to study, it also makes them the excellent cosmic messengers. As the trajectories of neutrinos are not disturbed by astrophysical magnetic fields and dust, reconstructing their paths can allow us to trace their source and probe the farthest reaches of the universe[4].

[4] See Section 1.3.

1.2.1 The Mystery Particle

The neutrinos were initially theorised as a solution to experiments in the early nighteenhundreds that suggested that the energy spectra of electrons and positrons produced in β^+ and β^- decays[5]

$$p^+ \rightarrow n^0 + e^+ \tag{1.1}$$

$$n^0 \rightarrow p^+ + e^- \tag{1.2}$$

were continuous rather than discrete as would be expected with the electron inheriting the bulk of the energy difference as kinetic energy[5].

[5] While Eq. 1.1 looks like proton decay, but can only happen inside nuclei. A free proton will never decay this way.

In a lecture in Zürich in 1957, German physicist Wolfgang Ernst Pauli suggested that the beta decays might not be two-body problems at all, but that a third unseen particle was involved and responsible for the distribution of electron energy. Conservation of charge and angular momentum restricted this particle to neutral charge and spin $\frac{1}{2}$. In addition, this mystery particle had to have no mass and to interact only through the weak force[6].

This description fits that of the neutrinos in Section 1.1[6], and indeed the particle responsible for the continuous energy problem turned out

[6] Except for not having any mass. This controversy is a central part modern research on neutrinos and the work done by IceCube, and will be addressed in Section 1.2.2.

to be what is today known as the neutrino. Its existence was confirmed for the first time in 1956 by Cowan and Reines, by studying the hypothesised mechanism of the inverse beta decay in which an electron antineutrino collides with a proton and form a positron and a neutron

$$\bar{\nu}_e + p^+ \rightarrow e^+ + n \tag{1.3}$$

[7] These will be important later.

Cowan and Reines used photomultiplier tubes (PMTs[7]) to detect gamma rays from the interacting products of the inverse beta decay to confirm that the decay had occurred and consequently that the electron antineutrino had to have existed[7, 8].

1.2.2 Oscillating Neutrinos

In addition to the electron neutrino, the existence of the muon neutrino was confirmed in 1962 by Lederman, Schwartz, and Steinberger. The tau flavour was not theorised until 1975 when its charged counterpart, the tau lepton, was discovered. The first hints of the existence of a third neutrino flavour came from the now famous Homestake Experiment. Officially named *Brookhaven Solar Neutrino Experiment*, the goal of the experiment was to measure the flux of of neutrinos emitting from fusion processes in the Sun, and indeed it was the first to successfully do so. It was in continuous operation from 1970 until 1994 and was headed by astrophysicists John N. Bahcall who performed the theoretical calculations prior to the experiment, and Raymond Davis, Jr. who designed the experiment. When comparing the predicted rates of neutrino detection to the experimental observations, only about a third of the expected amount of neutrinos were detected. The deficit was eventually established to be caused by the phenomenon of **neutrino oscillations**, which were first suggested by Bruno Pontecorvo in 1957[9, 10, 11].

Neutrino oscillations are, according to current understanding, a consequence of the fact that the eigenstates of neutrino propagation differ from the eigenstates of the weak neutrino interactions[8] which are the associated with the flavour of the neutrino. We will represent the flavour of a neutrino α as a superposition of mass eigenstates k:

[8] Elaborated in Section 1.5.

$$|\nu_\alpha\rangle = \sum_{k\in 1,2,3} U^*_{\alpha k} |\nu_k\rangle \tag{1.4}$$

where $U_{\alpha k}$ are elements of the Pontecorvo-Maki-Nakagawa-Sakata (**PMNS**) matrix

$$\begin{pmatrix} U_{e1} & U_{e2} & U_{e3} \\ U_{\mu 1} & U_{\mu 2} & U_{\mu 3} \\ U_{\tau 1} & U_{\tau 2} & U_{\tau 3} \end{pmatrix}, \tag{1.5}$$

a unitary[9] matrix that specifies the mixing strength of each flavour and mass state combination. An $n \times n$ unitary matrix will have n^2 free parameters, which in the case of the PMNS matrix[10] becomes 3 mixing angles and 6 complex phases, 5 of which can be absorbed as phases of the lepton fields resulting in 4 free parameters. The most common parameterisation of the PMNS matrix is the three mixing angles θ_{12}, θ_{13} and θ_{23} and the complex phase δ_{CP} which should be zero if neutrino oscillations are to obey charge-parity symmetry[11].

Using the shorthand $c_{ij} = \cos(\theta_{ij})$ and $s_{ij} = \sin(\theta_{ij})$, the resulting matrix is

[9] Under the assumptions of the Standard Model. Unitarity is defined by $U^\dagger U = I$.

[10] This is under the assumption that the neutrino is a Dirac particle. Treating it as a Majorana particle, where their mass term is not invariant under rephasings of the neutrino field, 2 additional CP phases are needed.

[11] This is theorised to not be the case. CP violating neutrinos could help explain the asymmetric distribution of matter and antimatter in the universe[12].

$$\begin{pmatrix} U_{e1} & U_{e2} & U_{e3} \\ U_{\mu 1} & U_{\mu 2} & U_{\mu 3} \\ U_{\tau 1} & U_{\tau 2} & U_{\tau 3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & s_{23} \\ 0 & -s_{23} & c_{23} \end{pmatrix} \begin{pmatrix} c_{13} & 0 & s_{13}e^{-i\delta_{\mathrm{CP}}} \\ 0 & 1 & 0 \\ -s_{13}e^{i\delta_{\mathrm{CP}}} & 0 & c_{13} \end{pmatrix} \begin{pmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.6}$$

The matrix has solutions on the form

$$|\psi(t)\rangle = e^{-iEt} |\psi(0)\rangle \tag{1.7}$$

with ψ being the quantum state and E the energy of the neutrino. Adjusting Eq. 1.4 accordingly, the time evolution of neutrino of flavor α can be written as

$$|\nu_\alpha(t)\rangle = \sum_{k\in 1,2,3} U^*_{\alpha k} |\nu_k(t)\rangle \tag{1.8}$$

To obtain the probabilities of oscillation, the mass eigenstates in Eq. 1.8 must be expanded in the weak eigenbasis, by inverting the PMNS matrix using the unitary condition

$$|\nu_k\rangle = \sum_{\beta\in e,\nu,\tau} U_{\beta k} |\nu_\beta\rangle \tag{1.9}$$

The probability of a neutrino oscillating from flavour α to flavour β at time t, can be found by inserting Eq. 1.7 and Eq. 1.9 in Eq. 1.8

$$\begin{aligned} P(\nu_\alpha \to \nu_\beta) &= \left|\langle \nu_\beta | \nu_\alpha \rangle\right|^2 \\ &= \left| \langle \nu_\beta | \left(\sum_{k\in 1,2,3} U^*_{\alpha k} e^{-iE_k t} \sum_{\lambda\in e,\nu,\tau} U_{\lambda k} \right) |\nu_\lambda\rangle \right|^2 \\ &= \left| \sum_{k\in 1,2,3} U^*_{\alpha k} U_{\beta k} e^{-iE_k t} \right|^2 \\ &= \sum_{k,l\in 1,2,3} U^*_{\alpha k} U_{\beta k} U_{\alpha l} U^*_{\beta l} e^{-i(E_k - E_l)t} \end{aligned} \tag{1.10}$$

Instead of treating the neutrinos as a wave packet[12], we assume that

[12] Which has been done in [13] and is fortunately phenomenologically equivalent to the result obtained here.

that the neutrino states propagate as plane waves with equal momenta, $p_k = p_l = p$. Using $E^2 = p^2 + m^2$, we rewrite the energy difference

$$E_k - E_j = p\left(\sqrt{1+\frac{m_k^2}{p^2}} - \sqrt{1+\frac{m_l^2}{p^2}}\right) \approx \frac{m_k^2 - m_l^2}{2p} \tag{1.11}$$

where the square roots have been Taylor expanded keeping only the first terms. Converting to units of the speed of light (and approximating that the neutrinos propagate a this speed, covering a distance L in time t), we get $t = L$ and $E \approx p$, and using $\Delta m_{kl}^2 = m_k^2 - m_l^2$ we can rewrite the phase terms of 1.10

$$e^{-i(E_k - E_l)t} \approx 1 - 2\sin^2\left(\frac{\Delta m_{kl}^2 L}{4E}\right) + i\sin^2\left(\frac{\Delta m_{lk}^2 L}{2E}\right) \tag{1.12}$$

which leaves the full oscillation probability

$$P(\nu_\alpha \to \nu_\beta) = \delta_{\alpha\beta} - 2\sum_{k,l\in 1,2,3} \mathrm{Re}\left(U_{\alpha k}^* U_{\beta k} U_{\alpha l} U_{\beta l}^*\right)\sin^2\left(\frac{\Delta m_{kl}^2 L}{4E}\right) + \sum_{k,l\in 1,2,3} \mathrm{Im}\left(U_{\alpha k}^* U_{\beta k} U_{\alpha l} U_{\beta l}^*\right)\sin^2\left(\frac{\Delta m_{lk}^2 L}{2E}\right) \tag{1.13}$$

From Eq. 1.13 it can be concluded that the probability of oscillation depends on the energy and the distance traveled by the neutrino, the square of the mass differences between the flavour states, and the elements of PMNS matrix. An example of this relation is shown in Figure 1.3, where the probability of oscillation from ν_μ to ν_τ for ν_μ neutrinos created in Earths atmosphere and detected in the IceCube detector is plotted as a function of their energy and their zenith angle θ_{zenith}, which corresponds to the distance traveled from the atmosphere, $L = 2R\cos(\theta_{\text{zenith}})$, where R is the radius of the Earth. This relation is why accurate reconstruction of energy and angle are of great importance for determining the elements of the PMNS matrix[14, 15, 16].

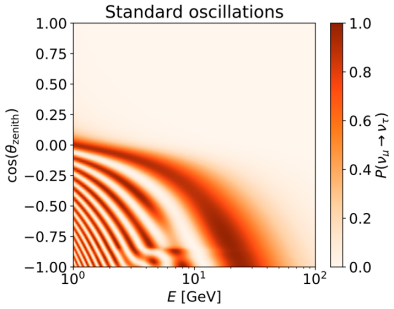


Figure 1.3: The probability of a ν_μ neutrino created in Earth's atmosphere and detected in the IceCube detector oscillating to a ν_τ before reaching the detector as a function of its energy and zenith angle, which is is a proxy for distance traveled.
Image from [17]

1.2.3 Neutrino Mass and Sterile Neutrinos

Since the neutrino oscillations require the flavour states to be linear combinations of the mass states, a direct consequence of the discovery of neutrino interactions is the indications that neutrinos must have mass. As mentioned in Section 1.1, in the Standard Model, neutrinos are described as massless point particles. In the current framework, all fermions except for the neutrinos have been found to exist in both left-handed and right-handed states, while the left-handed neutrino states

has been observed. This lack of right-handed neutrino states means that the neutrinos cannot be massive. One of the simplest extensions of the Standard Model that would explain this[13] is to add the missing right-handed neutrino (and left-handed antineutrino) singlets. These neutrino states would mix with the other three massive states, but not interact through the weak interactions, earning them the name **sterile neutrinos**. An additional theory, the **see-saw mechanism** describes how the currently known neutrinos acquire their mass by mixing with a very heavy iteration of the sterile neutrinos called as **heavy neutral leptons** (HNLs)[18, 19].

[13] And other well-established observational phenomena such as the matter-antimatter asymmetry of our Universe[18].

1.3 Cosmic Rays

Every second, the atmosphere of the Earth is bombarded by charged particles from the cosmos. Their origins range from relatively nearby locations like the Sun and our own galaxy to distant galaxies. The energy spans a large range from MeVs to EeVs and its distribution is shown in Figure 1.4. Upon contact with the Earth's atmosphere, the particles interact with the with the atomic nuclei in the atmosphere (e.g. ^{16}O, ^{14}N, and ^{40}Ar) producing a shower of particles, consisting mostly of light and unstable **mesons** that decay further into secondary products. The radiation from these **cosmic rays** are detectable on the surface of the earth[20, 21].

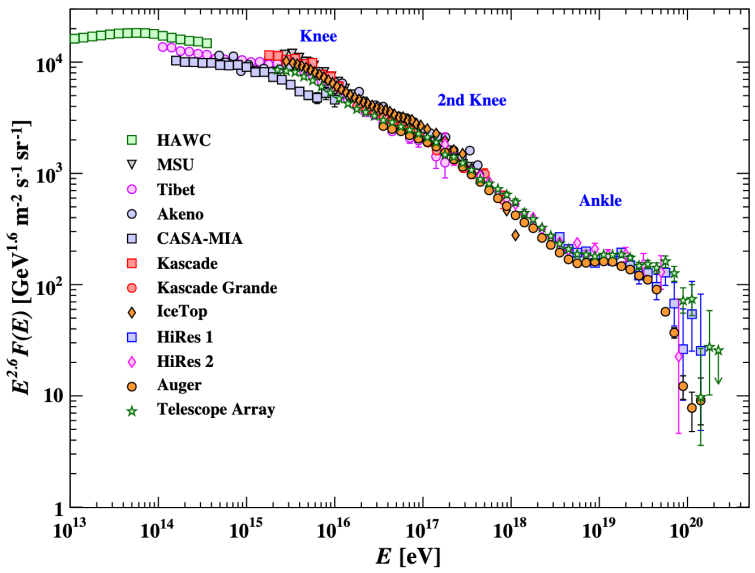


Figure 1.4: The energy distribution of cosmic rays.
Image from [22]

Being the lightest of the family, the **pions** are by far the most numerous of the mesons produced in the showers. They appear in both

their neutral (π^0) and charged (π^+ and π^-) variants, which primarily undergo the decay processes

$$\pi^0 \rightarrow \gamma + \gamma \quad (98.82\% \text{ of the time}) \tag{1.14}$$

$$\pi^+ \rightarrow \mu^+ + \nu_\mu \quad (99.99\% \text{ of the time}) \tag{1.15}$$

$$\pi^- \rightarrow \mu^- + \bar{\nu}_\mu \quad (99.99\% \text{ of the time}) \tag{1.16}$$

and thus are the principal contributor to the flux of neutrinos in the energy range relevant to the study of neutrino oscillations. In addition, the muons produced in these decays will also[14] contribute to the neutrino flux as they decay into electrons

$$\mu^- \rightarrow e^- + \bar{\nu}_e + \nu_\mu \quad (100\% \text{ of the time}) \tag{1.17}$$

A schematic of the pion decay chain is shown in Figure 1.5.

At larger energies, other mesons like the **kaon** (K^+ and K^-) become increasingly prevalent. These are also of interest to neutrino studies as they decay as

$$K^+ \rightarrow \mu^+ + \nu_\mu \quad (63\% \text{ of the time}) \tag{1.18}$$

$$K^+ \rightarrow \pi^0 + e^+ + \nu_e \quad (5\% \text{ of the time}) \tag{1.19}$$

and similarly for the K^- meson. The decay of Eq. 1.19 accounts of a significant amount of electron neutrinos. At higher energies still, the **D mesons** appear. These are of interest due to their large rest mass which allows them to decay into **tau leptons**, which again produce tau neutrinos during hadronic decay

$$\tau^- \rightarrow \mu_\tau + (1 \text{ or more } \pi)^- \quad (64\% \text{ of the time}) \tag{1.20}$$

[14] Given that they don't make it all the way to the detector without decaying, which they often will as explained in Section 1.4.

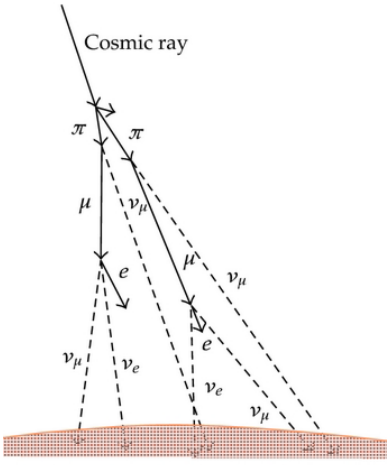


Figure 1.5: Schematic of a cosmic ray pion decay.
Image from [23]

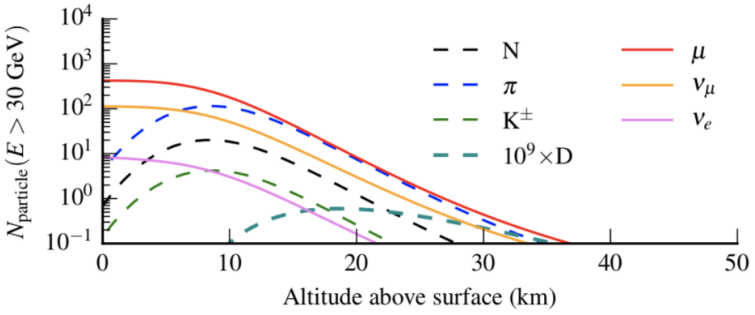


Figure 1.6: Distribution of cosmic ray secondary products (mesons) and the resulting muons and neutrinos. The D meson frequency is scaled up by 10^9 to illustrate the negligibility of its contribution and explain the absence of the tau neutrino.
Image from unpublished work by A. Fedynitch and R. Engel [24]

This means that the overall neutrino flux is also expected to contain tau neutrinos. In practice though, the tau contribution to the atmospheric neutrino flux is so many orders of magnitude lower than its muon and electron counterparts, that it has never been measured. A summary plot of the distributions of cosmic ray secondary products and the resulting muon and neutrino flux is shown in Figure 1.6, with the D meson flux scaled up by 10^9 and the tau neutrinos completely absent[22, 25, 26].

1.4 Muons

Aside from the neutrino, the muon is a strong contender for most relevant particle, to anyone involved with the IceCube detector. As seen in Section 1.3, the atmospheric muons are the dominating byproduct of almost every step of the meson decay process. These muons travel through the atmosphere at relativistic speeds giving them a long decay time that makes them abundant on the surface of the Earth and allows them to travel several kilometers before stopping due to energy loss and decaying. This makes them the primary source of (non-instrumental) noise in the IceCube detector, and a major target in the efforts described in Section 2.6.1 as well as in this work. It also makes them a possible calibration tool, especially muons that stop and deposit all their energy inside this detector[15].

Their long decay time also make muons special as a product of neutrino interactions. Its sibling, the τ lepton, has a lifetime $\tau_\tau = 0.29 \cdot 10^{-12}$ s and will in the low energy regime have a characteristic travel length of $c\tau_\tau \approx 87 \cdot 10^{-6}$ m[16]. The stable electron on the other hand, is too light to travel far in the detector array. Employing $E = \gamma m_0$, the energy loss can be described as a function of particle mass using the generalised Larmor formula descried in Section 1.6 in the relativistic limit

$$\frac{dE}{dx} \propto m^{-6} \tag{1.21}$$

which results in the electron quickly depositing all of its energy as Bremsstrahlung. Consequently, the muon has the perfect balance of long decay time and mass to travel furthest in the detector, which – as we shall see – provides the muon neutrino charged current interactions with a unique event signature[27, 28].

[15] A possibility that will be explored in efforts related to this work, but is left untreated in this thesis except for consideration in the event selection process.

[16] Ignoring relativistic time-dilating effects.

1.5 Charged/Neutral Current Decays

As hinted in Sections 1.1 and 1.2, the neutrinos interact only through the weak nuclear force[17], mediated by the Z^0 and $W^\pm$ bosons. The two force carriers give rise to two different types of interaction, the

[17] And gravity, as all particles do.

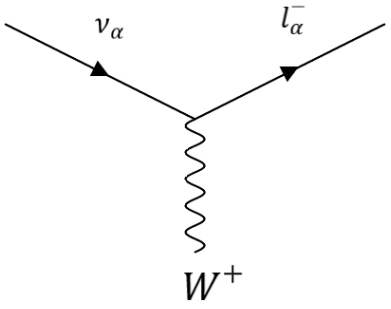


Figure 1.7: Feynman diagram of the interaction vertex of a charged current weak interaction in which a neutrino is transformed into a charged lepton of the same flavour (α) through the exchange of a charged W boson.

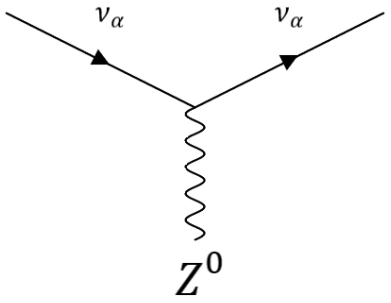


Figure 1.8: Feynman diagram of the interaction vertex of a neutral current weak interaction in which a neutrino is transformed into a neutrino of the same flavour through the exchange of a neutral Z boson.

charged current (CC) weak interaction and the **neutral current** (NC) weak interaction. The interactions can be visualised with the use of **Feynman diagrams**, a pictorial representation of the mathematical expressions that describe particle interactions. The interaction vertices that hold the most interest in experimental particle physics are shown in Figures 1.7 and 1.8.

Figure 1.7 depicts a CC process, in which a neutrino is transformed into a charged lepton of the same flavour (α) through the exchange of a charged W-boson. The reverse process is also allowed, transforming a lepton into a neutrino of the same flavour. Figure 1.8 shows the interaction vertex of an NC interaction, in which a neutrino is transformed into a neutrino of the same flavour through the exchange of a neutral Z-boson, depositing about half of its energy in the Z-boson. It is common to distinguish between CC and NC events in neutrino experiments, as the primary lepton in the CC interaction is generally detectable, while in the NC interaction the primary neutrino is generally not detectable, decreasing the total amount of detectable light from the interaction by a factor 2[29].

1.6 Energy Loss and Cherenkov Radiation

Particles in the IceCube detector are not observed directly, but are instead detected through the phenomenon of Cherenkov radiation, which travels in their wake as a clue of their presence. When a charged particle moves trough a dielectric medium, it loses energy as is travels in the form of photons, due to the disruption in the polarised surroundings caused by the displacement of the particle. Figure 1.9 shows the energy loss of a muon travelling in copper as a function of its momentum. While relationship between energy loss and and energy depends on the medium, and the IceCube detector uses ice and not copper as stopping medium, the two have similar stopping power and can be used interchangeably for illustration purposes.

In the central part of Figure 1.9, below 100 GeV, is the Bethe (or Bethe-Bloch) regime, where the primary source of energy loss is **ionisation**, the energy loss is described by the Bethe relation

$$\left\langle -\frac{\mathrm{d}E}{\mathrm{d}x} \right\rangle = Kz^2 \frac{Z}{A} \frac{1}{v^2} \left(\frac{1}{2} \ln \frac{2m_e v^2 \gamma^2 W_{\max}}{I^2} - v^2 \right) \tag{1.22}$$

where z is the charge number of the particle, Z is the atomic number, A the molar mass and I the mean excitation energy of the medium, $W_{\max}$ is the maximum energy loss of a single collision and $K = 4\pi N_A r_e^2 m_e$. Muons with energy below 50 GeV lose their energy quickly due to Coulomb interactions and are thus rarely seen in the IceCube detector, except when they are the product of neutrino interactions.

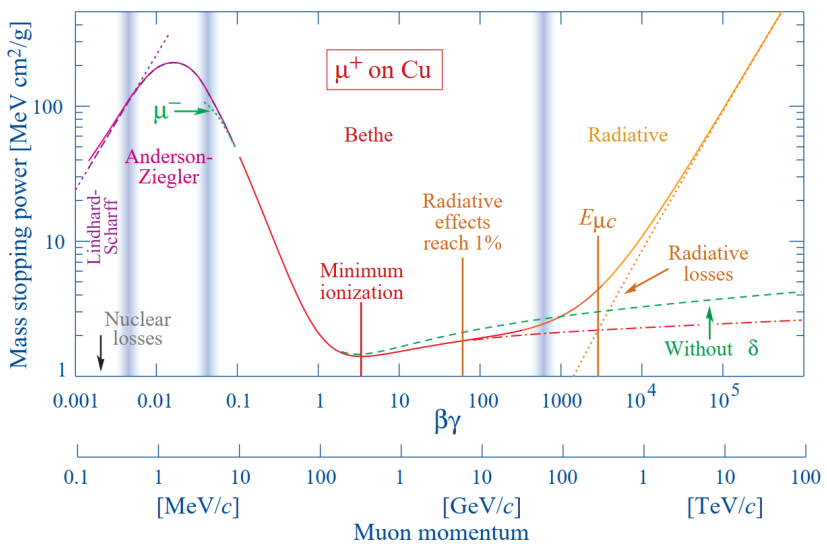


Figure 1.9: The energy loss of a muon travelling through copper as a function of particle momentum, showing the ionisation (Bethe) regime in the central part of the image (red) and the radiative regime to the right (yellow).
Image from [30]

For energies greater than 100 GeV, the energy loss is dominated by **radiative** effects such as **Bremsstrahlung** which, when derived classically, takes the form[18]

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{2}{3}e^2\gamma^6\left(\dot{v}^2 - (v \times \dot{v})^2\right) \tag{1.23}$$

where e is the elementary charge, λ is the Lorentz facor $\lambda = (1 - v^2/c^2)^{-1/2}$ and $\dot{v}$ is the time derivative of the particle velocity. To compare this to Eq. 1.22, which is position derived rather than time derived, one must simply take the muons to be in the relativistic limit (where $t \approx x$ in units of c)[30, 31].

[18] A generalisation of the Larmor formula[31].

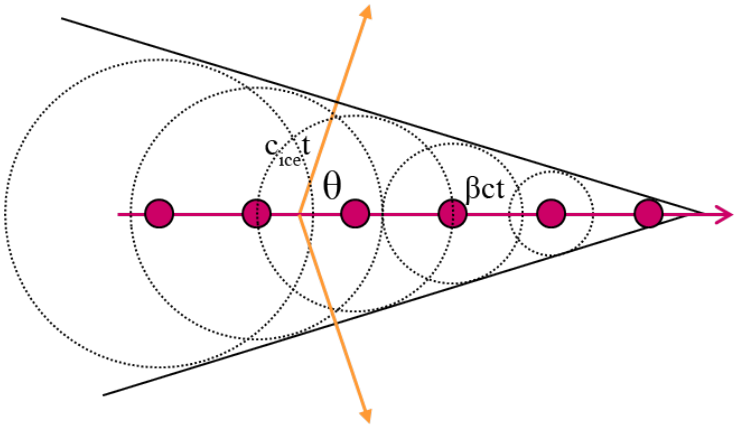


Figure 1.10: Diagrammatic representation of the Cherenkov radiation, showing the particle trajectory in red from left to right and the direction of the propagation of constructively interfering light in orange. The angle theta can be calculated from the right triangle defined by the travelling distances βct and $c_{\mathrm{ice}}t$.
Image from [32]

When a charged superluminal[19] particle travels through a dielectric medium, the photons emitted travel slower than the particle, leading to

[19] A particle travelling at a greater velocity than the speed of light in the medium in question ($v > c_{\mathrm{medium}}$). This does not contradict special relativity, as $v < c_{\mathrm{vacuum}}$.

a buildup of constructively interfering wavefronts at a fixed angle with respect to the trajectory of the particle as shown in Figure 1.10. From the geometry of figure one can deduce the angle from the distance travelled by the particle, βct[20], and the distance propagated by the wave with phase velocity c_{ice}[21] emitted at $t = 0$, which will be $c_{\text{ice}}t$. The angle θ is described by

$$\cos\theta = \frac{c_{\text{ice}}t}{\beta ct} = \frac{\frac{c}{n}}{\beta c} = \frac{1}{n\beta} \tag{1.24}$$

where $n = \frac{c}{c_{\text{ice}}}$ relates the refractive index n to the speed of light in vacuum and medium (ice).

As the result, the light emitted will appear as a moving cone travelling at the speed of the particle. When wieved in the plane perpendicular to the particle trajectory, the light will be detectable as well-defined circles as shown in Figure 1.11. This phenomenon is called **Cherenkov radiation**[22], and any particle physics experiment involving a transparent dielectric medium can use Cherenkov radiation to observe and describe particle interactions. It is used in Imaging Atmospheric Cherenkov Telescopes (IACTs) to detect high energy gamma radiation with the atmosphere as detection medium, whereas neutrino detectors like IceCube, as mentioned, use large volumes of water in either frozen or liquid state to detect Cherenkov light from neutrino-nucleus interactions[35, 32].

[20] Where β is a common way of describing the particle velocity in the relativistic regime, as a fraction of the speed of light in vacuum $\beta = v_{\text{particle}}/c$.

[21] Here denoted c_{ice} since ice is the medium in question.

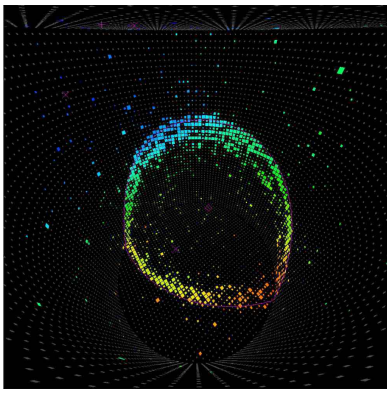


Figure 1.11: Cross section of a cone of Cherenkov radiation from a muon event observed in the Super-Kamiokande experiment, clearly showing the circular shape.
Image from [33]

[22] After Soviet scientist Pavel Cherenkov, who received the 1958 Nobel Prize for being the first to detect it experimentally in 1934, although Marie Curie had observed it in a highly concentrated radium solution in 1910[34, 35].

2 IceCube Neutrino Observatory

Contents

As the name suggests, IceCube is shaped as a hexagonal cylinder.
Jespersen, Schauser, Severin & Vinther, 2021

The studies featured in this thesis concern the IceCube Neutrino Observatory, and the effort to study netrino oscillations through low energy IceCube neutrino events. This chapter describes the experiment and the detector array with a focus on the current event cleaning and reconstruction methods, event signatures, and the upcoming IceCube Upgrade.

2.1 The Largest Neutrino Telescope in the World

Encompassing a cubic kilometer of the antarctic ice sheet near the Amundsen–Scott South Pole Station, the IceCube Neutrino Observatory is a one-of-a-kind detector, designed to study the universe through the detection of neutrinos. With the first string being deployed in 2005 and construction finishing on December 17th 2010, by 2013, IceCube

had detected 28 neutrinos that likely originated outside our solar system[36]. While the main target of IceCube is to study extreme astrophysical phenomena such as supernovae, gamma-ray bursts, black holes and neutron stars, data from the detector also allows for study of the neutrino itself[37, 38]. This is the exact purpose of the IceCube OscNext group which this work is an extension of. With accurate reconstruction of the energy, direction and flavour composition of atmospherics neutrinos, the oscillation parameters described in Section 1.2.2 can be calculated more precisely than ever[39]. A visualisation of the IceCube detector array is shown in Figure A.2 in the Appendix, and a more schematic overview is shown here in Figure 2.1.

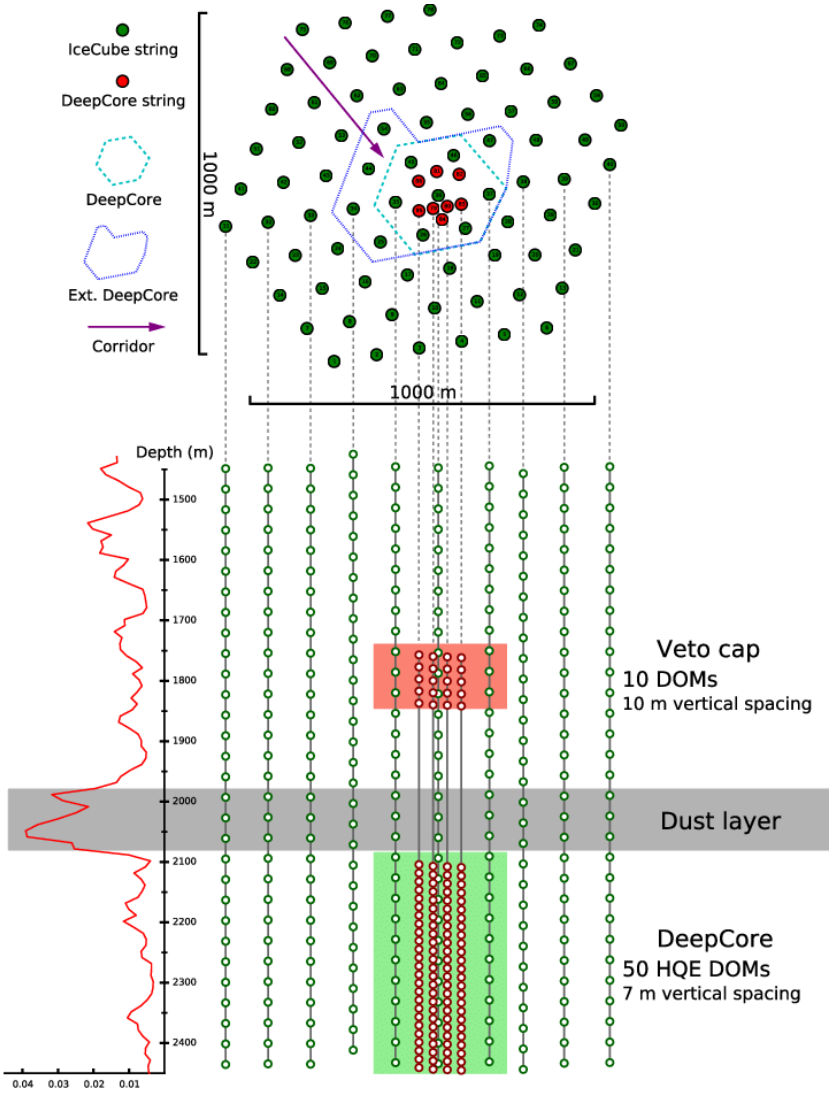


Figure 2.1: Overview of the IceCube detector array, depicting original IceCube strings and DOMs (green), DeepCore strings and DOMs (red) and the dust layer (gray band).
Image from [40]

2.2 DOMs

The original structure of IceCube consists of 5,160 spherical optical sensors called Digital Optical Modules (**DOMs**, shown in Figure 2.2), deployed evenly on 86 hexagonally distributed strings at depths between 1,450 and 2,450 meters as indicated by the green dots in Figure 2.1. Between the 2,000 and 2,100 meters lies the so-called dust layer, a band of less pure ice that distorts the light passing through it more than the rest of the ice sheet. The dust layer was unknown during the inception of IceCube, but newer generations of DOMs are placed with it in mind.

Each DOM is equipped with a photomultiplier tube (**PMT**) capable of detecting the Cherenkov Radiation of the superluminal particles travelling though the ice. When triggered, the DOM will record the interaction for $0.43\mu s$ or $6.4\mu s$ depending on the internal digitiser and transmit the digital data to the counting house on the surface above the array. The current generation of DOMs are built with a single sensor pointing down in order to primarily detect neutrinos[41].

Figure 2.2: Original IceCube Digital Optical Module (DOM).
Image from [42]

2.3 Extensions

2.3.1 DeepCore

One of the major flaws discovered after comission of the IceCube project, was the inadequacy of its hexagonal grid of sensors. The regular grid allows for specific angles where particles can tunnel unnoticed down the rows of DOMs as indicated by the purple arrow in Figure 2.1. In an effort to remedy this, the first upgrade to IceCube, **DeepCore** was constructed. Consisting of 480 new DOMS on 6 new strings in the central part of the detector primarily below the dust layer, DeepCore increases sensor density and breaks the discrete symmetry of the original structure[44]. The DeepCore strings are represented as red dots in Figure 2.1.

2.3.2 IceCube Upgrade

During the arctic summer of 2022/23, another 7 strings are scheduled to be installed in the DeepCore region. Carrying around 700 detectors, it will feature 2 new DOM types: The Multi-PMT Digital Optical Module (**mDOMs**), and the Dual Optical Modules (**D-Eggs**). The mDOM (shown in Figure 2.3) has 24 PMTs distributed evenly across its surface, and as such provides more information on the direction of the incoming signal. The D-Eggs have a PMT pointing down like the original DOMs as well as PMT pointing directly up, and are expected to shed more light on the properties of the hole ice, the *new* ice around

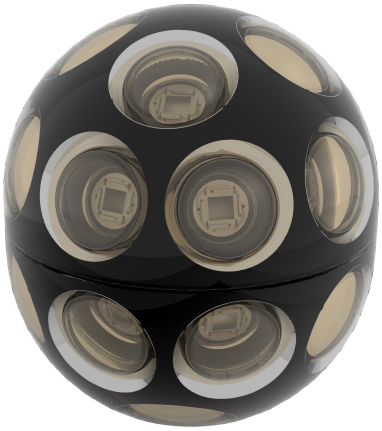


Figure 2.3: IceCube Upgrade mDOM.
Image from [43]

the strings that is a result of the drilling process.

With the IceCube upgrade, noise is an increasingly large problem, especially in the low energy regime. The main source of noise in IceCube is the decay of radioactive isotopes (^{40}K and isotopes from decay chains of ^{238}K, ^{235}K and ^{232}T) present in the DOM pressure vessel or the PMGs themselves. The decays both deposit energy in that glass that produces scintillation photons, and produce electrons which emit Cherenkov radiation, both of which trigger the PMTs. One of the tasks of this thesis is to filter signal from noise at the individual PMT level, to allow the upgrade data to be used for low energy reconstruction.

IceCube Upgrade is the forerunner of and even larger envisioned extension of IceCube, **IceCube Gen2**, proposed to double the number of DOMs over 8 cubic kilometers of ice[45].

2.4 Simulation

Similarly to many other fields of physics, IceCube uses simulations for a lot of its studies. Simulation is is very useful in the field of particle physics, which involves interactions where the outcome is based on probabilities. **Monte Carlo** (MC) simulation describes the approach of simulating chains of probabilistic events in order to gain knowledge of the distribution of the final outcome.

Since the actual data from the IceCube detector is unlabeled, and the supervised reconstruction algorithms used in this work requires large amounts of labeled data for training, the approach is to train of MC simulated examples and reconstruct the real data using the trained model. Consequently, it is crucial that the MC simulation accurately represents the complex interactions of neutrinos and other particles inside the detector.

The simulation of events in DeepCore and IceCube upgrade are divided into separate stages covering the full chain of interactions from cosmic rays generating showers in the atmosphere, to photons produced by neutrino interactions in the ice sheet. The library of algorithms is rather extensive, but in the field of low energy neutrinos reconstruction, the following three are the most important.

2.4.1 CORSIKA

The COsmic Ray SImulation for KAscade, or **CORSIKA**, is a relatively old algorithm that simulates background muons from from cosmic ray interactions in the atmosphere. It performs detailed simulations of primary particles including protons, light nuclei and photons and their interactions with the atmosphere which is represented in five layers with different densities and seasonal variations. The products of

the interactions are then propagated to the surface of the earth, taking into account the energy losses, scattering, and deflection in Earth's magnetic field. The interactions of these secondary particles and their products make of the final part of the **air shower**. IceCube uses CORSIKA by selecting muons with a direction that causes them to travel though the IceCube detector. It could be used to simulate neutrinos in the same way, but due to the small cross-section of neutrinos, this is extremely inefficient[46].

2.4.2 MuonGun

While detailed and accurate, the simulations of CORSIKA are computationally expensive and not very flexible. **MuonGun** is developed based on the results of CORSIKA simulations and bypasses the the simulation of the full shower. Instead, it starts the simulation on the very edge of the detector and has the ability to create large samples of data at low computational cost[47].

2.4.3 GENIE

The **GENIE** generator is used in IceCube to simulate neutrinos in the 1 GeV to 1 TeV energy range. The events are generated from a power law spectrum and forced to interact with electrons or nuclei inside the detector. Interactions are simulated with respect to neutrino flavour and energy, and covers elastic scattering, quasi-elastic scattering, resonance production, and deep inelastic scattering with deep inelastic scattering being the dominant interaction type for energies > 10 GeV. The results are propagated out of the nucleus while allowing hadrons to re-interact before exiting the nucleus[48].

2.5 Events in Data

Regardless of origin in Monte Carlo simulation or real data, or of passing level 2 or level 7, IceCube events are recorded and stored in **.i3 files**, a highly flexible (and consequently quite complex) file format unique to the IceCube collaboration. Each event in IceCube becomes a **frame** in an .i3 file, and each frames holds all recorded **features** of the event, information on whether the event has passed certain criteria, and information on its origins if known. It is common within the oscillation group to perform work directly on .i3 files, but owing to the predecessors of this work, the standard practice within the GraphNet group[1] is to extract the desired features from .i3 files and save them in SQLite[2] databases for faster access. For each event, any number of DOMs may record a signal, so the observed features are saved for each DOM, making for a feature matrix of fixed length (number of features)

[1] See Section 3.4.

[2] A fast and (very) popular SQL database engine written in C[49].

and variable height (number of DOM pulses). The most common features extracted and used for training of neural networks in this work are shown in Table 2.1[50].

Table 2.1: Important event features used for GNN reconstruction divided into currently available features and features that is only be available in Upgrade data.

Feature Variable	Description	Data Type
dom_x	The x coordinate of the DOM position	float
dom_y	The y coordinate of the DOM position	float
dom_z	The z coordinate of the DOM position	float
dom_time	The time of recording of the event	float
dom_charge	The charge recorded by the DOM	float
rde	Relative dom efficiency	float
pmt_area	The PMT surface area	float
	Upgrade only features	
string	Index unique to each string	integer
pmt_number	Index unique to each PMT on a given DOM	integer
dom_number	Index unique to each DOM on a given string	integer
pmt_x	The x unit vector of the PMT direction	float
pmt_y	The y unit vector of the PMT direction	float
pmt_z	The z unit vector of the PMT direction	float
pmt_type	The type of DOM	integer

Most of the features should be self-explanatory, but the DOM charge and relative DOM efficiency are somewhat specific to the IceCube data collection process:

DOM charge describes the total charge recorded by the DOM, and is determined by integrating the whole waveform measured in the interval defined by the **leading** and **tailing** edges. The leading edge is the slope between the bins where 10 and 90% of the first measured peak are reached extrapolated to the baseline. The trailing edge is the point where the waveform drops below 10% of the first peak for the final time. An illustration of the calculations can be seen in Figure A.3 in the Appendix. The figure is borrowed from the IceCube FeatureExtractor documentation[51] where a detailed explanation of the calculation can also be found.

Relative DOM efficiency is a measure of the DOM's electrical sensitivity to light, decribed by the quantum efficiency relative to the rest of the DOMs. The quantum efficiency is the ratio between the number of charge carriers collected at the PMT terminal and the number of photons hitting the PMT's photoreactive surface[52, 41].

2.6 Current Methods

The cleaning and reconstruction methods currently employed in IceCube are complex, reliable, thought-through and often based directly physical observations and models. Many of them are also slow, and with room for improvement in performance as well as speed. Based on previous work on this subject, this work is based on the hypothe-

sis that Neural Networks can improve or outperform several of these routines. This section describes what we are trying to beat.

2.6.1 Cleaning

Multiple levels of both hardware software filtering exist within the IceCube data stream to filter out noise and eventually unwanted particle such as muons. The data selection pipeline developed and utilised by the Oscillation Group which is also used in this work, is split into seven layers of cleaning from inital trigger to final selection. The rate of events at levels 2-7 are shown in Figure 2.4.

Level 1, Initial Trigger: When a DOM records a photoelectric signal above its baseline, the first limiter of what gets recorded is the **Hard Local Coincidence** (HCL), a pairwise comparison of the DOM and each of its neighbours. If two neighbouring DOMs report a signal within $\pm 1\mu s$, it is likely that the light detected stems from a physical event in the ice, such as Cherenkov radiation from a neutrino interaction. This process is common for the entire collaboration. For neutrino events, the **Single Multiplicity Trigger** (SMT) filter is used to keep all events with at least 8 instances of local coincidence.

Level 2, DeepCore Filter: The part of the selection process specific to the Oscillation Group begins with the Level 2 DeepCore filter. This filter is specifically designed to select events that have the potential to come from low-energy neutrino interactions, by looking for events with more than three single multiplicity triggers (SMT3). Then, the **Seeded Radius-Time** SRT cleaning algorithm, which is based on special relativity, is applied as well as a veto that eliminates events associated with atmospheric muons.

Level 3, Simple Muon & Noise Cuts: At this level, a number of predetermined cuts on low-level variables are applied to efficiently remove noise and muons. The cuts take into account mainly the number of hits and a pusle series and their distributions and time and space.

Level 4, BDTs: At this stage, the data is classified as either pure noise, muons or neutrino candidates using **Boosted Decision Trees** (BDTs). BDTs are a simple type ML classification algorithm, and it is possible to use machine learning at this stage because the cuts at level 3 ensure good agreement between data and MC.

Level 5, Muon Corridor Cuts: By this point, almost all of the noise and the majority of the muons will have been eliminated from the sample. The remaining muons are difficult to identify, because they usually the ones that pass through the corridors of IceCube described in Section 2.3.1 and indicated by the purple arrow in Figure 2.1. For this reason, a **corridor cut module** provides directional cuts that eliminate 97% of the muons at Level 4 at the cost of 48%of the neutrinos.

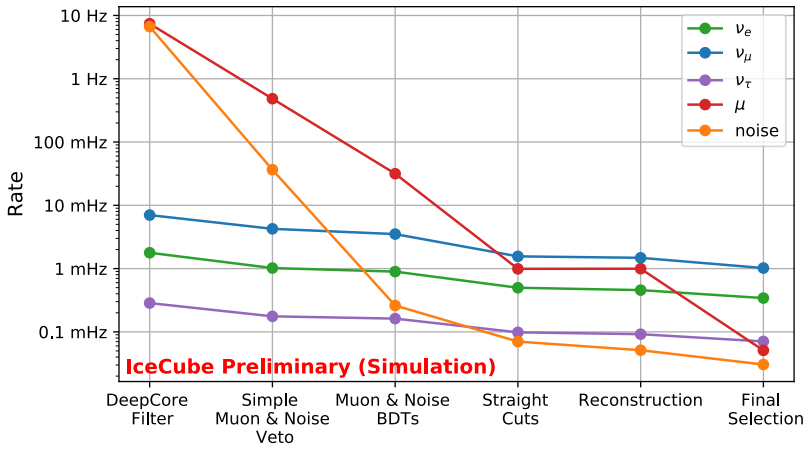


Figure 2.4: The event rate of each particle type and noise at each level of the OscNext data cleaning process.
Image from [25]

Level 6, Event Reconstruction: At Level 6, the data rates are finally low enough for reconstruction software to be applied. The current method for high-quality reconstruction is **RetroReco** and is described in Section 2.6.2.

Level 7, Final Selection: Using the reconstructed event features from level 7, the remaining muons are removed using another BDT[25].

2.6.2 RetroReco

*As seen from a spherical coordinate transformation of its interaction vertex.

Table 2.2: Important reconstructed variables in the OscNext analysis.

Truth Variable	Description	Data Type
energy	The energy of the particle	float
position_x	The x coordinate of the interaction vertex	float
position_y	The y coordinate of the interaction vertex	float
position_z	The z coordinate of the interaction vertex	float
azimuth	The azimuth angle of the particle's interaction vertex*	float
zenith	The zenith angle of the particle's interaction vertex*	float
pid	The type the particle using the convention from [53]	integer
interaction_type	The type of interaction (charged of neutral current)	integer
interaction_time	The time of the interaction	float
	Pulse level variables	
truth_flag	Pulse level noise/physics flag	integer

The current implemented method for event reconstruction is RetroReco, an algorithm based on **reconstruction tables**, large multidimensional tables constructed by simulating a particular light source at many locations and orientations in the array and running each simulation many times. A given event hypothesis can be compared to the tables by interpolating them to obtain a probability density function (pdf) of photon observation time for each individual DOM. This is summarised in the likelihood function that sums the probabilities of observing a number

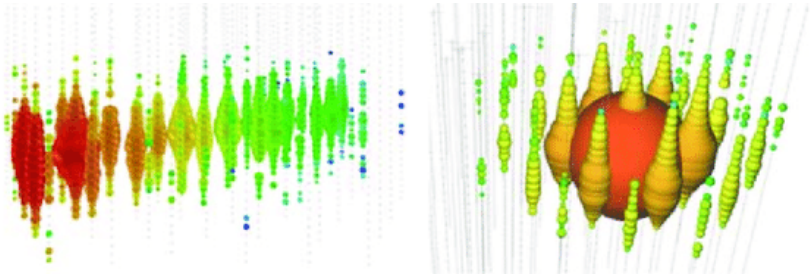


Figure 2.5: **Left:** Track-like event signature in the IceCube detector. **Right:** Cascade-like event signature in the IceCube detector.
Image from [55]

of photons in a certain time bin, given a corresponding **photon expectation** which is derived from the event hypothesis. Maximising this likelihood provides the most likely parameters describing the event. The most relevant variables reconstructed by RetroReco that also feature in this work, are shown in Table 2.2[54].

2.7 Event Signatures

It is common within IceCube to distinguish neutrino events by the physical shape of the trace their interactions make in the detector array, sometimes known as the **event signature**. The two major neutrino event signatures observed in IceCube are **track**-like and **cascade**-like events[3]. The event signature of a neutrino interaction is determined by the flavour of the neutrino and the type of interaction (charged or neutral current).

[3] Other, more complex signatures exist, but are not relevant to this work.

Track-like event signatures are left by the charged current interactions of muon neutrinos and a small fraction of tau neutrinos. In the ν_μ charged current interaction, a muon is created, and as discussed in Section 1.4, muons have a long enough lifetime to not decay immediately and sufficient mass to travel a significant distance in the detector before losing all of its energy to Bremsstrahlung. As a result, the Cherenkov radiation of the muon product can be observed along its trajectory in the detector following the interaction as a cylindrical light beam. The tau lepton produced in the tau neutrino charged current decay is massive enough to decay as

$$\tau^- \rightarrow \nu_\tau + \mu^- + \bar{\nu}_\mu \quad (17.39\% \text{ of the time}) \tag{2.1}$$

as shown in Figure 2.6, producing a muon with enough energy to leave a track-like signature. An example of the track-like event signature is displayed on the left in Figure 2.5[30, 57].

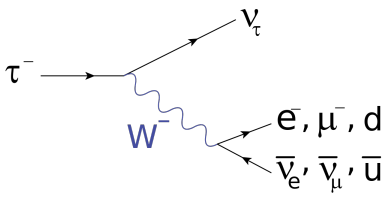


Figure 2.6: Decay modes of the tau lepton decay. Several pions may be created (as combinations of u's and d's) but here only the $\mu^- \bar{\nu}_m u$ mode is of interest.
Image from [56]

Cascade-like events include all neutral current interactions as well as the charged current interactions of electron neutrinos and the majority of tau neutrinos. These decays are very localised in the detec-

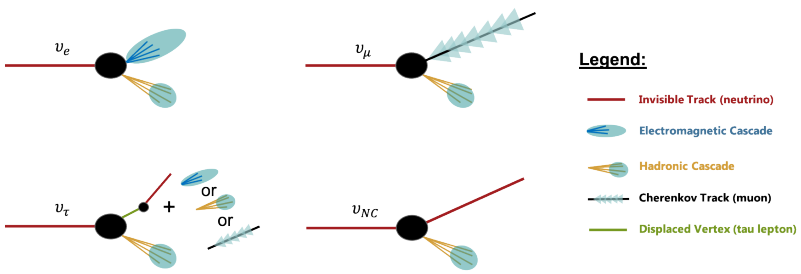


Figure 2.7: Overview of the possible neutrino interaction types and their event signatures.
Image from [25]

tor and produce hadronic and electromagnetic showers that appear as spherical light pattern in the detector. These are generally harder to reconstruct as they lack the clear directional asymmetry of track-like events[58]. A summary of the possible neutrino interactions and their signatures is shown in Figure 2.7.

3 Machine Learning in IceCube

Contents

The tools used in this work to improve the reconstruction of IceCube events is a part of the vast field of data analysis methods known as machine learning, which concerns models that are able learn from data, make decisions and discover patterns with minimal human interference. Not at all a knew scientific field[1], it has gained a lot of traction in recent years – with an estimated 100 articles per day on the on the popular public repository of research papers, Arxiv[61]. As many of the tools involve mainly matrix operations, machine learning also emerging within in the field of high energy physics, where data rates are ever increasing and the need for novel GPU based methods increasing with them[62].

Like many other types of algorithms, machine learning algorithms are designed to accept some input and produce an answer. The input could be an image, and the answer could be whether or not the

[1] The term was coined in 1959 by IBM researcher and computer science pioneer Arthur Samuel[59], and by 1963 a reinforcement learning *machine* had learned to play Tic-Tac-Toe[60].

image contains a car, or it could be a history of the stock market and probability of a certain stock rising or falling in value. What separates machine learning from other algorithms, is that instead of being programmed to produce the correct answer the machine learning model is programmed with a flexible architecture and allowed to *learn* how to treat data by exposure to examples. The foundation of machine learning is the process of feeding a model a sufficient amount of examples with corresponding correct answers – 10.000 images with and without cars and a label for each image – and allowing the computer to *learn* how to arrive at the correct answer from the examples alone. There are numerous ways to got about this, but one of the most common methods – and the one applied in this thesis – is neural networks[63].

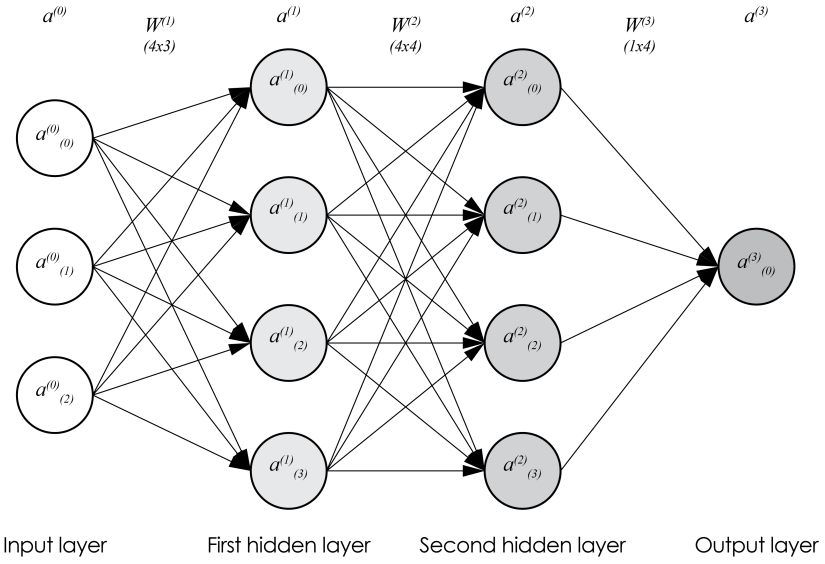


Figure 3.1: Representation of a simple neural network showing neurons as circles and connection as lines, with the input layer on the far right and the feed-forward direction from left to right. *Image made in Adobe Illustrator.*

3.1 Neural Networks

The complex and very powerful world of neural networks begins with a **neuron**. Simply put, a neuron is just a number. What makes it a neuron, is its relation to other numbers – other neurons in the network[2]. In neural networks, neurons are arranged in **layers**. Each layers represent a step through the machine learning algorithm from input to answer. In a simple neural network (an MLP, multi-layer-perceptron) each neuron in a layer, is connected to each neuron in the next layer. This connection is represented by another number a **weight**, with which the number from the neuron in question will be multiplied to obtain its contribution to the neuron in the next layer. The value of the neuron in the next layer, will consequently be a weighted sum of the neu-

[2] It is also common to refer to a neuron as a node, but here the term neuron is employed to avoid confusion with the nodes of graph theory.

rons in the previous layer. Sometimes, another number called a **bias** is added to the sum, and often, an **activation function** – a function that constrains the value as shown in Figures 3.2, 3.3 and 3.4 – is applied before assigning to next neuron its new value. The expression for the value a of the i$^{\text{th}}$ neuron in the k$^{\text{th}}$ layer will be

$$a_{(i)}^{(k)} = f\left(\sum_{j=0}^{N-1}\left(w_{(i,j)}^{(k)} \cdot a_{(j)}^{(k-1)}\right) + b_{(i)}^{(k)}\right) \tag{3.1}$$

with w being the weights, b the biases, f the activation function and N being the number of neurons in the $(k-1)^{\text{th}}$ layer.

With each neuron in one layer connected to each neuron in the next, and each connecting having its own unique weight, in a layer with 4 neurons connected to another layers with 4 neurons, each neuron will have 4 connections, giving 16 total connections and weights, as shown between the layers $a^{(1)}$ and $a^{(2)}$ in Figure 3.1. The first layers is called the **input layer**, and as the name suggest, is where the network receives the input, and its dimensions are defined by the input data structure. The last layer is the **output layer**, which is interpreted as the answer, and its dimensions are constrained by the type of answer desired. The layers between the input and output are the **hidden layers**, and there can be any number of these (modern deep learning methods us hundreds of layers) and they can have any number of neurons. When applying the algorithm to an input, the neurons in each layers receives a value based on the neurons in the previous layer, and the answer is determined from the values of the output layer. The weights and biases are the variable components of the model, and it is by changing their values that the network learns to produce correct answers. This process is addressed in Section 3.3[64].

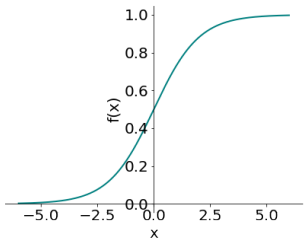


Figure 3.2: A sigmoid activation function

$$f(x) = \frac{1}{1+e^{-x}}$$

which keeps the output between 0 and 1.

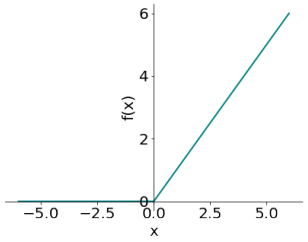


Figure 3.3: A Rectified Linear Unit (ReLU) activation function

$$f(x) = \max(0, x)$$

which sets values below 0 to 0.

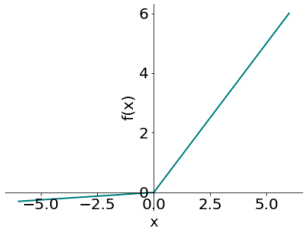


Figure 3.4: A Leaky ReLU activation function

$$f(x) = \max(0.05x, x)$$

which linearly scales values below 0.

3.2 Graph Neural Networks

Graph Neural Networks (GNNs) are a sub-type of neural networks based on the mathematical theory of graphs. It has arisen from a desire to incorporate information about the structure of, and relationships between components of data into machine learning. For problems where the data has clear non-euclidian structure, such as social networks, infrastructure models and molecular structures, the relational information has been shown to be an valuable asset for accurate predictions[65, 66].

3.2.1 Graphs

In the context of graph theory, a graph consist of a set of **nodes** and **edges** that connect two nodes. A graph is often denoted by $G = V, E$,

[3] V for verticies, since N will be used for neighbourhood.

where V is the set of nodes[3] and E is the set of edges. An edge $e_{ij} = (v_i, v_j)$ has two endpoints $v_i \in V$ and $v_j \in V$. v_i and v_j are then said to be **joined** by e_{ij} or that the two nodes are **adjacent**. Additionally, v_i and v_j are called each others **neighbours**. Generally, the **neighbourhood** of v is written as $N(v) = \{k \in V | (v, k) \in E\}$. One can also define the adjacency matrix A as a matrix where $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$. An edge in a graph can be **directed** or **undirected**. An **directed graph** is a graph where all edges are directed, and similarly, an **undirected graph** is a graph where all edges are undirected.

Additionally, a graph can be equipped with a set of **nodes features** X associated with each node, and possibly also a set of **edge features** X associated with each edge. An example of such a graph could be a collection of employees, with the edges features being variables like their salary and the time they been employed, and the criteria for a connecting being if the employees are collaborating on a project, with the edges features being how long the project has been going on or the amount of time the employees spend together working on it. Such a graph is often called an **attributed graph**. Figure 3.5 shows a schematic of an undirected graph on the left, a directed graph in the center, and an attributed graph on the right[67, 68].

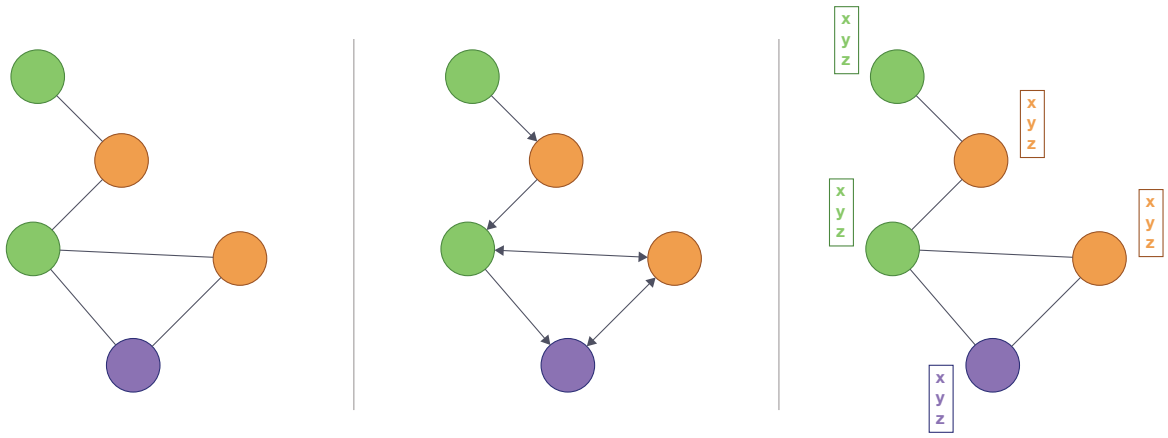


Figure 3.5: Schematics of different types of graphs. **Left:** An undirected graph. **Center:** A directed graph. **Right:** An attributed graph.
Image made with Adobe Illustrator.

As the world is not made up of nodes and edges, before one can use a GNN, one must first turn the data at hand into a graph. This can be done in a number of ways, for the examples used aboce, the choice a quite natural, but for IceCube events it is non-trivial since no DOMs are inherently *more connected* than others, and since there exists both spatial and temporal hierarchies among the pulses. In this work, the choice has been to use a simple k-nearest-neighbours (KNN) method, connecting each pulse to its 8 nearest neighbours in space-time. The number 8 is chosen as it has been found to work well in previous work on the subject, and while a few attempts at alternative

connection schemes have been explored within the group during the duration of this work, no compelling arguments have been found for switching from the 8 KNN method.

3.2.2 Message Passing

Instead of having multiple layers with different numbers of nodes, the layers of a GNN are typically new representations of the original graph with the same number of nodes. The nodes can have a different number of features and – as we will see later – the connections between the nodes may also change. Between each layer, the nodes are updated depending on the values of their neighbours, through the principle of **message passing**. Here, a node receives a *message* from each of its neighbours in the form of an array numbers, that has a length corresponding to the number of features in the new layer. When considering a node v_i that receives a message m_j from each of its neighbours $v_j \in N(v_i)$, the message can be written as

$$m_j = f(v_j)W \tag{3.2}$$

where W are a matrix of learnable weights just like in a regular neural network, and f is a function specific to the message passing, which is often its own small neural network. The new value of each feature of the node is then found using chosen aggregation (like the sum) of the messages

$$\tilde{m}_i = \underset{j \in N(i)}{\text{Aggr}}(m_j) \tag{3.3}$$

and updated using another function G specific to the message passing scheme

$$\tilde{v}_i = G(\tilde{m}_i) \tag{3.4}$$

A variety of complex operations can be applied to reach the message. The message passing operator used in this thesis, EdgeConv[69], uses a small neural network (here an MLP) to arrive at the message

$$\tilde{\boldsymbol{v}}_i = \sum_{j \in N(i)} \text{MLP}(\boldsymbol{v}_i || \boldsymbol{v}_j - \boldsymbol{v}_i) \tag{3.5}$$

where $\tilde{\boldsymbol{v}}_i$ denotes the new features of i, $N(i)$ are the neighbours of i and $(\boldsymbol{v}_i || \boldsymbol{v}_j - \boldsymbol{v}_i)$ means a concatenation of the vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j - \boldsymbol{v}_i$. This means the weights of the network are concealed in the MLP inside the operator.

3.2.3 GNNs as a Generalisation of CNNs

Convolutional neural networks (CNNs) are another subset of neural networks that are widely used in image recognition and processing.

They are specifically for processing pixel data and use a mathematical operation called a convolution in some of the layers in place of regular matrix multiplication as described in Section 3.1. A convolution is defined as an operation on two functions f and g that expresses how the shape of one is modified by the other. In CNNs, a **filter**[4] is applied to the data. The filter – smaller in dimensions than the input data – is applied to all regions of the input. It can be thought of as sliding across an input image and picking up important features along to way. This process is shown in Figure 3.6. On the left is shown a 3×3-filter operating on the top left region, and on the right is shown an example of features picked up by different filters operating on the same image[70].

[4] Sometimes known as a kernel.

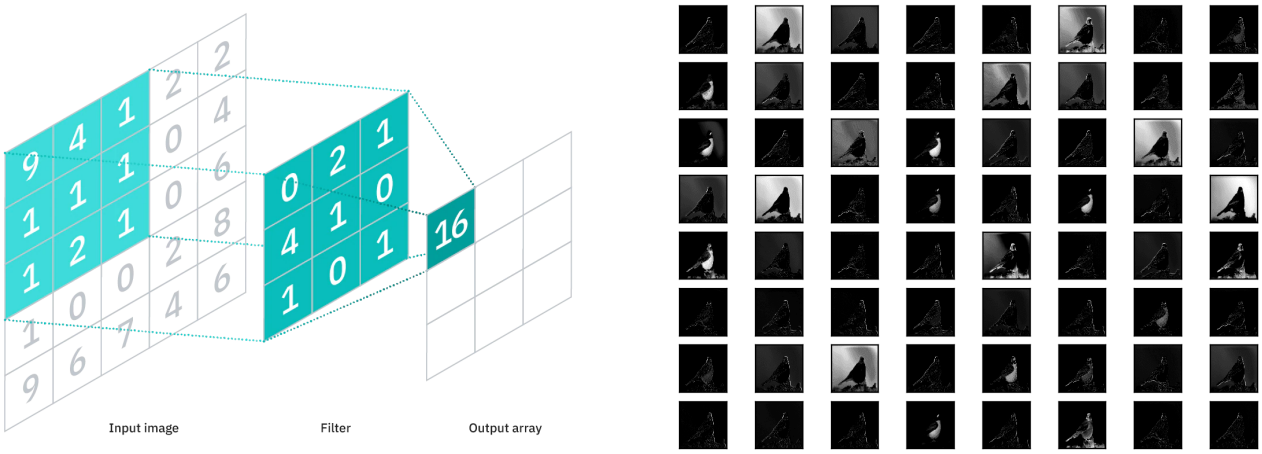


Figure 3.6: **Left:** A 3×3-filter operating on the top left region of a 2D array. *Image from* [71]
Right: An example of features picked up by different CNN filters operating on the same images.
Image from [72]

GNNs can be viewed as generalisations of CNNs in two aspects. Firstly, the CNNs require data to have a rigid grid structure. In the world of image recognition, this is natural, but for data of different kinds, like the IceCube events used in this thesis, a lot of constraints need to be applied a lot of choices need to be made on how to treat irregularities, pad empty areas and so on[5]. The GNNs do not require this structure but can still utilise the relational information of nodes that are adjacent in the graph space. Secondly, the many possible ways of performing message passing between nodes at each layer[6] can be viewed as a generalisation of CNN filters. Choices in message passing scheme and the use of edge features provide great flexibility on the passing of information between neighbouring nodes.

[5] This has been done for high energy events in IceCube with good results[73].

[6] Which is naturally known as a graph convolution.

3.3 Training Neural Networks

Sections 3.1 and 3.2 describe the structure of neural networks, and how they process data to produce an answer. But this is only the *machine*,

this section will describe the *learning*. As mentioned earlier, machine learning works by supplying the models with **training examples**, data with the correct answer attached, and allows the model to gradually update its weights and biases to fit the training data.

3.3.1 Loss Function

The first building block of this process is the **loss function**. When a training example is fed through the model, the result it produces is compared to the correct result using a loss function, a function that assigns a value – also known as a cost – to how wrong the model's prediction is. The choice in loss functions can shift the focus during training between examples where the prediction is very wrong and ones where it is almost correct. The **Mean Absolute Error** (MAE) Loss has a linear relation between loss and difference from the truth and may focus too much on already decent predictions, while the **Mean Square Error** (MSE) Loss has produces a loss that is the square of the difference between truth and prediction and may punish outliers too heavily. The **LogCosh** Loss is a mixture between the two approaches, approximating a power of two near zero and a linear relation further away. The behaviors of the three loss functions are shown in Figure 3.7.

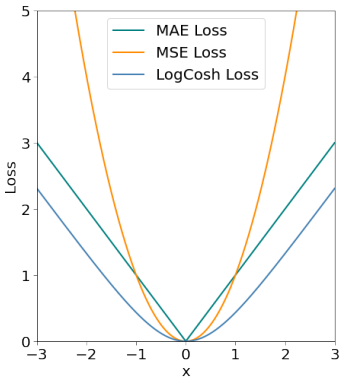


Figure 3.7: The behaviors of Mean Absolute Error (MAE), Mean Square Error, (MSE) and LogCosh Loss near zero.

Two additional loss functions are used extensively throughout this work, the **Binary Cross Entropy** Loss and the **von Mises-Fisher** Loss. These are important and complex enough to require individual explanation.

ID	Truth	Predicted Probability	Corrected Probability	Log
0	1	0.78	0.78	-0.1079
1	1	0.90	0.90	-0.0458
2	0	0.10	0.90	-0.0458
3	1	0.47	0.47	-0.3279
4	0	0.51	0.49	-0.3098

Table 3.1: Examples of true and predicted values for Binary Cross Entropy Loss, and the corresponding corrected probabilities.

Binary Cross Entropy Loss: As the name suggests, the BCE loss is used for binary classification[7]. The formal definition of the BCE loss is that it is the negative average of the log[8] of corrected predicted probabilities. The term predicted probabilities can be explained using Table 3.1 which show a selection of example truths and predictions for an imaginary classifier. The table shows how the corrected probability is equal to the predicted probability of a given event belonging to the correct group, or more precisely, when the truth is one, the corrected probability is equal to the predicted probability, and when the truth is zero, it is one minus the predicted probability. The total loss then

[7] A generalised version for multi-label classification exists, but will not be covered here[74].

[8] Log here is the natural logarithm.

becomes

$$\mathcal{L} = \frac{1}{N} \sum_{i=0}^{N-1} \log(p_i) \tag{3.6}$$

where N is the number of training examples and p_i is the corrected probability[75, 76].

Von Mises-Fisher Loss: The VMF loss is used for angles, and can in principle be used to regress direction in a space of any dimension. It is based on the von Mises-Fisher distribution, a probability distribution on a (n-1)-sphere[9] in $\mathbb{R}^n$. For a predicted direction $\boldsymbol{x}$ and true direction $\boldsymbol{\mu}$, denoted the mean direction of the probability distribution, the PDF is given by

$$p_n(\boldsymbol{x}|\boldsymbol{\mu}, \kappa) = C_n(\kappa) \exp\left(\kappa \boldsymbol{\mu}^T \boldsymbol{x}\right) \tag{3.7}$$

where κ is called the **concentration parameter**, which is analogous the term $\frac{1}{\sigma^2}$ in a normal distribution: When κ is zero, the distribution is uniform, and when it is large the distribution is concentrated around $\boldsymbol{\mu}$. Multiple ways exists of estimating if the full underlying distribution is known κ[78, 79], but this work employs the trick of allowing of allowing the model to output an estimation of κ alongside the predictions, which has been found to work well. Figure 3.8 shows samples drawn from distributions with different values of κ on a sphere. C is a normalisation constant, given by[80, 81]

[9] For mathematical definitions of n-spheres, see [77].

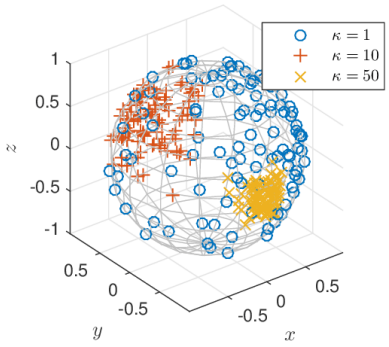


Figure 3.8: Samples from 3 different von Mises-Fisher distributions with $\kappa = 1$, 10 and 50, respectively on a 3D sphere (2-sphere).
Image from [82].

$$C_n(\kappa) = \frac{\kappa^{n/2-1}}{(2\pi)^{n/2} I_{n/2-1}(\kappa)} \tag{3.8}$$

where I_v is the **modified Bessel function** of the first kind at order v. In 3 dimensions, this reduces to

$$C_n(\kappa) = \frac{\kappa^n}{4\pi \sinh(\kappa)} \tag{3.9}$$

In this work, only one angle of direction is regressed at a time, and thus the 2 dimensional distribution is used. This is sometimes known just as the von Mises distribution or the **circular normal distribution**, and the normalisation constant becomes

$$C_n(\kappa) = \frac{1}{2\pi I_0(\kappa)} \tag{3.10}$$

The probability distribution can be used to create a log function by taking the negative log likelihood of Eq. 3.7, using the 2D normalisation from Eq. 3.10

$$\mathcal{L} = -\log p_n(\boldsymbol{x}|\boldsymbol{\mu}, \kappa) = -\log(\kappa) + \log(2\pi) + \log(I_0(\kappa)) \tag{3.11}$$

The final term of Eq. 3.11 is non-trivial but has been treated in [83] and [79].

3.3.2 Optimising

With a definite measure of the error of its prediction, the model can use this information to update its weights (and biases if present) through the process of **backpropagation**. If the neural network is be describes as a function f of its weights θ

$$f(\boldsymbol{x}, \theta) = \hat{\boldsymbol{y}} \tag{3.12}$$

mapping from data $\boldsymbol{x}$ to prediction $\hat{\boldsymbol{y}}$, and the loss is $\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$, then the objective is to fit θ to minimise $\mathcal{L}$ over all data in the training sample

$$\tilde{\theta} = \text{argmin}_\theta \sum_{\text{data}} \mathcal{L}(f(\boldsymbol{x}, \theta), \boldsymbol{y}) \tag{3.13}$$

Because all operations of the neural network can be expressed as matrix operations, the derivative of all the individual layer weights can be calculated using the chain rule. For each training example, these gradients are used by the **optimiser** along with the loss and the parameter called the **learning rate** to update each weight.

The optimiser used in this thesis is called Adam[84]. It uses **stochastic gradient descent** to navigate the highly non-convex loss landscape to find the optimal values for θ. Among the hyper-parameters of ADAM, the learning rate is crucial to the outcome of the gradient descent. This is a common parameter of most optimisers, and can be viewed as the *step size* of the gradient descent. High learning rates result in quicker descent while lower learning rates may lead to the discovery of minima that might otherwise have been skipped. This learning rate is usually kept quite low (≈ 0.001) as to not overfit the model to a single data point. For this reason, it is common to present to model with the same training sample any number of times when training. The process of training on the entire data sample exactly once is called an **epoch**, and training a complex model will usually require several epochs. The work done in this thesis uses a learning rate scheduler which varies the learning rate with each epoch as shown in Figure 3.9[85, 86].

The optimal choice for optimiser and learning rate scheduler has been investigated in previous studies, and during the work on this thesis, no reason have been found to question their results. Loss functions are task specific, and will be addressed separately for each task in Chapters 4 to 7.

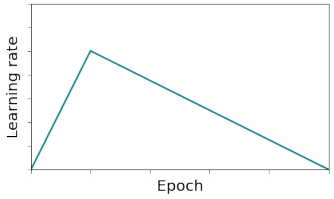


Figure 3.9: The general behavior of the learning rate scheduler. The range of epochs and learning rate can be adjusted as needed.

3.3.3 Training, Validation and Testing

In theory, one can train a neural network forever and with a large enough network eventually be able to perfectly regress or classify each example in the dataset and produce a loss of zero. This is however not

desirable, since the network has learned learn patterns specific to the training dataset which may not be generalisable to the task the network is meant to perform. When presented with new a new sample of data, this **overtrained** network will underperform on the new data, looking for features that do not exist. A principal concern in Machine Learning therefore, is to find the sweet spot where the model has learned the general structure of the data but not the individual traits of each example[10]. To obtain this, the model is presented with a **validation sample** after each epoch for which the loss is calculated. When the loss of the validation sample stops decreasing with every epoch, the model is assumed to have converged to a loss minimum. It is common to implement **early stopping** which terminates the training process after a set number of epochs with no improvement in validation loss. To obtain a measure of the performance of the model after training, it is generally presented with a third unknown dataset called the **testing sample** for which the loss is calculated and used to benchmark the model[88].

[10] This is a simplification of a complex mathematical subject, the debate of which is ongoing[87]. In this work, validation and early stopping is implemented as it has been found to work well.

Figure 3.10: Example of the behavior of the training and validation loss with each epoch of training. Early stopping kicks in after epoch 25 following five epochs without improvement in the validation loss.

3.3.4 Batching

The most common way to train on a large dataset, is to present the model with a batch of training samples at the same time, and calculate the average loss of the batch and use this for backpropagation. This has two major advantages: Firstly, it further prevents overfitting to a single outlier in the dataset, and makes the gradient descent smooth and gradual. Secondly, one of the major strengths of machine learning is that most of the calculations are matrix operations that can carried out in parallel on GPUs, feeding the GPU multiple training examples simultaneously results in a significant increase in training speed.

3.4 GraphNeT

As of April 2021, the collected GNN reconstruction efforts of the IceCube research groups in the Nielsh Bohr Institute (NBI) the Technische Universität München (TUM) have been congregated in the Graph neural networks for Neutrino event Reconstruction group (GraphNeT). GraphNeT both is a toolkit designed for IceCube data but with the intent of being generally applicable to data from other neutrino detectors and the group responsible for its development and deployment within IceCube. The work in this thesis uses the GraphNeT framework for graph creation and model training, and subsequently, some time has gone into aiding development of the framework and implementation of features necessary for the following investigations. Full documentation of GraphNeT can be found on the GraphNeT github page[89].

3.4.1 DynEdge

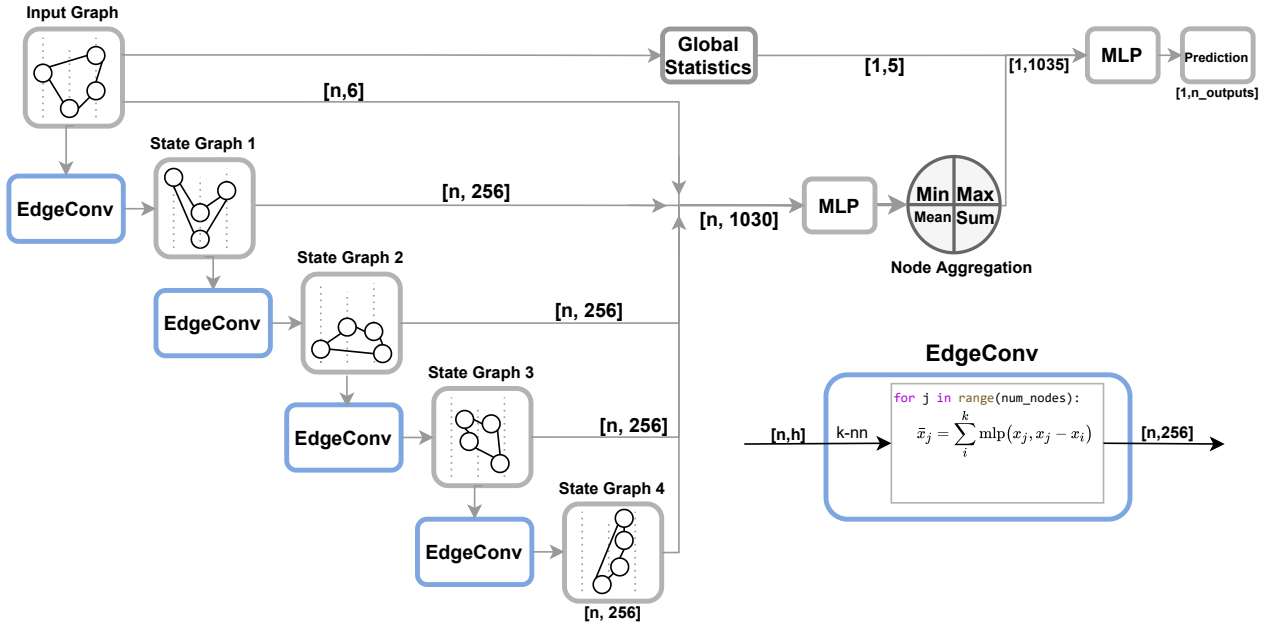


Figure 3.11: Schematic overview of DynEdge with data flowing from top left to top right. Left side shows the message passing and reconnection of nodes, lower right shows the EdgeConv operator, and upper right shows the concatenation and aggregation step.
Figure from soon-to-be published work by the GraphNeT group.

The primary ML model used in within the GraphNeT group is called DynEdge, and its architecture is shown in the schematic in Figure 3.11. DynEdge was developed during a previous masters project in this group and uses the EdgeConv message passing scheme described in Section 3.2.2, but they key feature that distinguishes DynEdge from other GNN architectures is that between every convulution layer, a new state of the graph is generated. After each convolution, new edges

calculated using a KNN[90] algorithm replace the existing ones, and the following message passing is based on these new edges. The KNN algorithm takes the first 3 features of the graph as input, which during the first convolution represent the physical position (x,y,z) of the DOM hit, but which represent features in an increasingly complex latent space with each convolution.

The features of each graph state are saved and eventually concatenated together to form the basis for generation of the output. Since the model accepts 6 input features, and each graph state after the initial (of which there are 4) are chosen to have 256 features, the intermediate result of feeding a single event into the model is a matrix with dimensions $N \times 1030$ (as $1030 = 6 + 4 \cdot 256$), where N is the number of pulses (DOM hits) in the given event. The number N raises a computational issue, since this number varies with each event, and turning a matrix of variable dimensions into a single number for the output is a non-trivial task. In DynEdge, this is solved by aggregating the pulses using minimum and maximum values, mean and sum, illustrated by the circular object in Figure 3.11, which forces the first dimension to always have length 4. Before and after aggregation, the data is fed through simple MLPs, the last of which can accept some global features calculated from the initial graph state as additional features, and casts the data to the desired output shape.

The work done over the course of this project has included a few attempts at improving on or producing an alternative to DynEdge, but not much seems to be gained from changes to the architecture. The general consensus in the GraphNeT group is that time is better spent contributing to the framework around DynEdge as well as studies of the performance of loss functions and data processing. As such, the DynEdge architecture is employed for the following studies.

3.5 Preprocessing

3.5.1 Data Selection

When performing a task such as binary classification, where one of the outcomes is significantly more prevalent than the other – such as identifying rare signals in an abundance of noise, training the model on a dataset that represents the true distribution is often not the optimal approach. The probability of being rewarded with a low loss score by classifying each event as noise will be very high, and the model will typically learn to classify all events as noise while still obtaining a satisfactory average loss. The common solution to this problem is to select the training sample such that each outcome is equally prevalent.

In some cases the option of selecting a training sample with equal

distributions of outcomes may not be available. In the study performed in Chapter 6, models are trained on a pulse level with varying amounts of noise in each event, which means that one cannot control noise distribution by re-weighting events. One could modify each pulse either before or during cleaning, but since the pulses are both the input data and the classification targets, this is undesirable. Another choice is to use loss function that approximates minimising a parameter that is robust to changes in the distributions, such as the F1-score or area under PR Curve. Several attempts at creating differentiable approximations to these quantities have been made already[91, 92], but since the implementations are written in TensorFlow and would have to be adapted to PyTorch and since the performance in Chapter 6 is already satisfying, this has not been investigated further.

3.5.2 Feature Transformation

Due to the nature of loss functions and backpropagation, the training of neural networks behaves the best when the input features share the same order of magnitude, and even better when that order of magnitude is around 1. A common choice when dealing with data that approximates a uniform distribution[11], is to scale the data to be between 0 and 1. When the data is normally distributed or has a more complex shape, this might not be the optimal choice, as outliers may force the majority of the data to lie within a limited range. In this case, more advanced scaling methods can be used to obtain a suitable distribution of the data[94, 95]. The following describes two transformation methods included in the scikit-learn[96] data science package for Python:

[11] For example an image recognition problem where each colour intensity is equally likely[93].

Robust Scaler is designed to be robust to outliers. Where the simpler Standard Scaler subtracts the mean and divides by the standard deviation, this transformation scales the data by

$$\hat{x}_i = \frac{x_i - \text{median}(x)}{\text{IQR}(x)} \tag{3.14}$$

where $IQR(x)$ is the Interquantile Range[12] of x. Since the transformation is based on percentiles, outliers have no effect regardless of their difference from the bulk of the data.

[12] Defined as the difference between the 75th and 25th percentiles of x. Other percentiles may be used if desired.

Quantile Transformer is a non-linear method which transforms the features the follow a normal or uniform distribution. It allows for more direct comparison of data that span different scales by spreading out more frequent values, but distorts the linear correlation between values on the same scale.

Both of these methods have been applied in the early stages of this work. Both of them however, require fitting of the transformation to a representative subset of the data before they can be applied. Over

the course of this work it was agreed within the GraphNeT group, that the transformations could be substituted for simpler predefined transformations, without a decrease in performance. These transformations are determined with the knowledge of the approximate shape and range of the data, and since the data from the current IceCube and DeepCore arrays differ from the (simulated) Upgrade data, the appropriate transformations are different as well. The transformations used on Upgrade data are shown in Table 3.2 along with an example range from a sample of the OscNext simulation, and the original IceCube and DeepCore transformations can be found on the GraphNeT github page[89].

Table 3.2: Transformations of event features from Upgrade data used for GNN reconstruction, with an example range from a sample of the OscNext simulation.

Feature Variable	Range	Mean	Transformation ($\hat{x}_i$)
dom_x	$-579.0 \leq x \leq 576.3$	36.8	$= x_i/500$
dom_y	$-521.1 \leq x \leq 509.5$	-45.3	$= x_i/500$
dom_z	$-647.1 \leq x \leq 573.2$	-249.2	$= x_i/500$
dom_time	$5.71 \cdot 10^3 \leq x \leq 4.48 \cdot 10^4$	$1.20 \cdot 10^4$	$= -x_i/(2 \cdot 10^4)$
dom_charge	$8.71 \cdot 10^{-3} \leq x \leq 6.24 \cdot 10^3$	1.00	$= \log_{10}(x_i)/2$
pmt_area	$8.17 \cdot 10^{-3} \leq x \leq 4.44 \cdot 10^{-2}$	$1.87 \cdot 10^{-2}$	$= x_i/0.05$
Upgrade only features			
string	$1 \leq x \leq 93$		$= (x_i - 50)/50$
pmt_number	$0 \leq x \leq 23$		$= x_i/20$
dom_number	$1 \leq x \leq 113$		$= (x_i - 60)/60$
pmt_x	$-0.96 \leq x \leq 0.91$	$-1.5 \cdot 10^{-3}$	$= x_i$
pmt_y	$-1.96 \leq x \leq 0.96$	$-1.5 \cdot 10^{-3}$	$= x_i$
pmt_z	$-1.0 \leq x \leq 1.0$	$-6.7 \cdot 10^{-1}$	$= x_i$
pmt_type	$20 \leq x \leq 130$		$= x_i/130$

While these transformations have been sufficient for the input variables used in this work, they have not always performed well when applied to the target variables. In Chapters 4 to 7, transformation of target variables will be addressed when relevant.

3.6 Performance Measures

For classification tasks, the performance metrics used in this work are generally simple and will be addressed when relevant. For regression tasks, the evaluation of performance on diverse data with complex distributions is non-trivial. The general evaluation scheme is common to all regression tasks, with minor variations for each regressed variable.

For a reconstructed variable, the **residual distribution**, R, is calculated. The residual is generally the difference between true and reconstructed values, but the calculation varies for a few of the target variables. For zenith angle and interaction time, the residuals are simply $R_{zenith} = \text{zenith}_{\text{reco}} - \text{zenith}_{\text{true}}$ and $R_t = t_{\text{reco}} - t_{\text{true}}$ respectively. For energy (which is approximately exponentially distributed) the residu-

als are calculated as fractions of the true energy: $R_E = \frac{E_{\text{reco}} - E_{\text{true}}}{E_{\text{true}}}$ and shown in percentages. For azimuth angle, which is circular such that a prediction of 359 degrees is actually close a truth value of 1 degree, the residual is calculated as the shortest angular distance, keeping the correct sign of rotation, which mathematically become is calculated as

$$R_{\text{azimuth}} = (\text{azimuth}_{\text{reco}} - \text{azimuth}_{\text{true}}) \mod 360 \tag{3.15}$$

$$\tilde{R}_{\text{azimuth}} = \begin{cases} R_{\text{azimuth}} & \text{if } R_{\text{azimuth}} \leq 180 \\ R_{\text{azimuth}} - 360 & \text{if } R_{\text{azimuth}} > 180 \end{cases} \tag{3.16}$$

where the angles are in degrees and $\tilde{R}_{\text{azimuth}}$ is the final residual score for the azimuth angle. Figure 3.12 shows an example of the distribution of R, indicating the mean of the distribution R, the 16th and 84th percentiles, and the Interquantile Range $IQR(R)$.

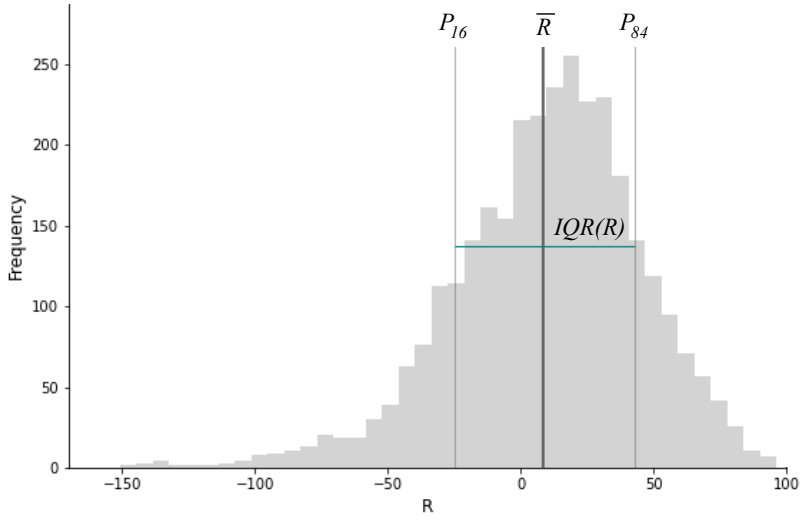


Figure 3.12: Example of residual distribution R, showing the mean of the distribution R, the 16th and 84th percentiles, and the Interquantile Range $IQR(R)$.

As a summary metric for the residual distribution, the **width** of the distribution, W, is calculated as

$$W(R) = \frac{IQR(R)}{1.349} \tag{3.17}$$

where the constant 1.349 ensures that the width corresponds to one standard deviation, assuming that residual distribution is gaussian[13]. Since quantiles are used, W is robust to outliers and can be used to describe the performance of the model with a single number where lower is better and 0 describes perfect reconstruction.

To obtain an uncertainty on W, **order statistics** are employed. The derivation can be found in Section B.1 and yields the estimation of the

[13] As in Section 3.5.2, the Interquantile Range, $IQR(R)$, is the difference between the 75th and 25th percentiles of R.

standard error on the width W

$$\sigma_W = \frac{1}{1.349}\sqrt{\frac{0.25 \cdot (1 - 0.25)}{n}\left(\frac{1}{f(R_{0.25})^2} + \frac{1}{f(R_{0.75})^2}\right)} \tag{3.18}$$

where n is the sample size, f is a pdf (usually a gaussian) that matches the distribution of W and $R_{0.25}$ and $R_{0.75}$ are the 25h and 75th percentiles of R, respectively[97].

4 Interaction Time Reconstruction

Contents

4.1 Task

The following describes a task that was undertaken about halfway through the work on this Master's thesis. By that point, it was already established that the GraphNeT framework and DynEdge architecture performed well on most of the target variables shown in Table 2.2. However, no model seemed to produce satisfying results on the **interaction time** variable when compared to the Retro reconstruction. Figure 4.1 shows the distributions of the predicted (x-axis) and true (y-axis) interaction time as a 2D histogram on the right and a scatter plot on the left for the retro reconstruction. A black line across the diagonal indicates perfect reconstruction. Figure 4.2 shows the same plot for an initial attempt at reconstruction with DynEdge.

Figure 4.3 shows a 1D histogram of the distributions for each model on top of one another with the truth in the background. At first glance the reconstructions look very similar in the scatter plot and both follow the diagonal line reasonably well, and for the 1D distributions, the GNN reconstruction seems to follow the truth more closely than the Retro reconstruction. But Figure 4.3 is slightly misleading, since firstly, the y-axis is on a base 10 log scale because of the irregular distribution of the time variable, which makes it hard to interpret the differences, and secondly, the plot only shows the distribution of truth and reconstruction independently, and says nothing about the accuracy of individual events. A clearer image of the performance can be gained by looking to the 2D histograms from which it is apparent that the distribution is wider for the GNN reconstruction and as such diverges

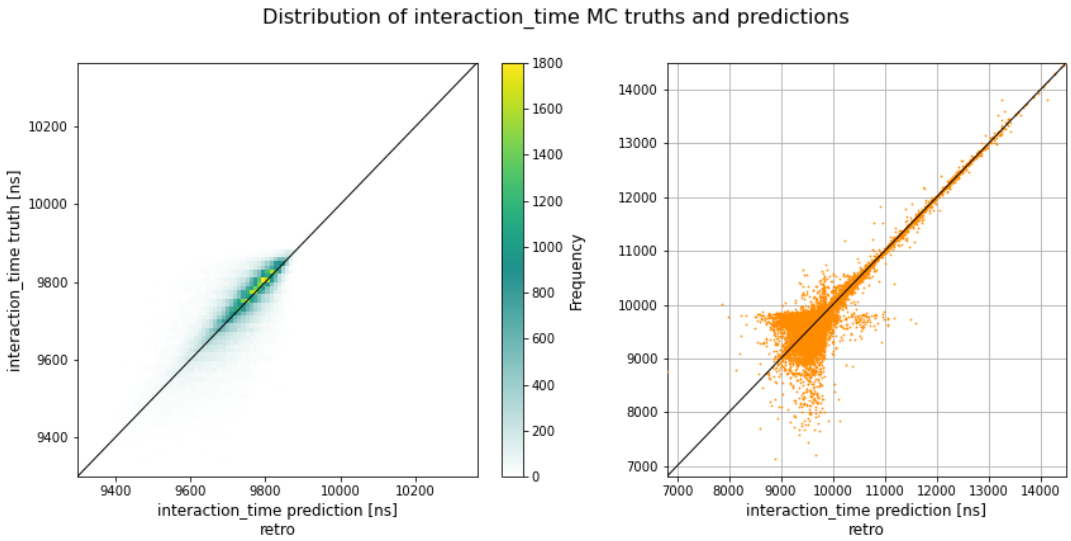


Figure 4.1: Distribution of the true and reconstructed interaction time, reconstruction by RetroReco as a 2D histogram on the left and a scatter plot on the right. The diagonal black line indicates perfect reconstruction.

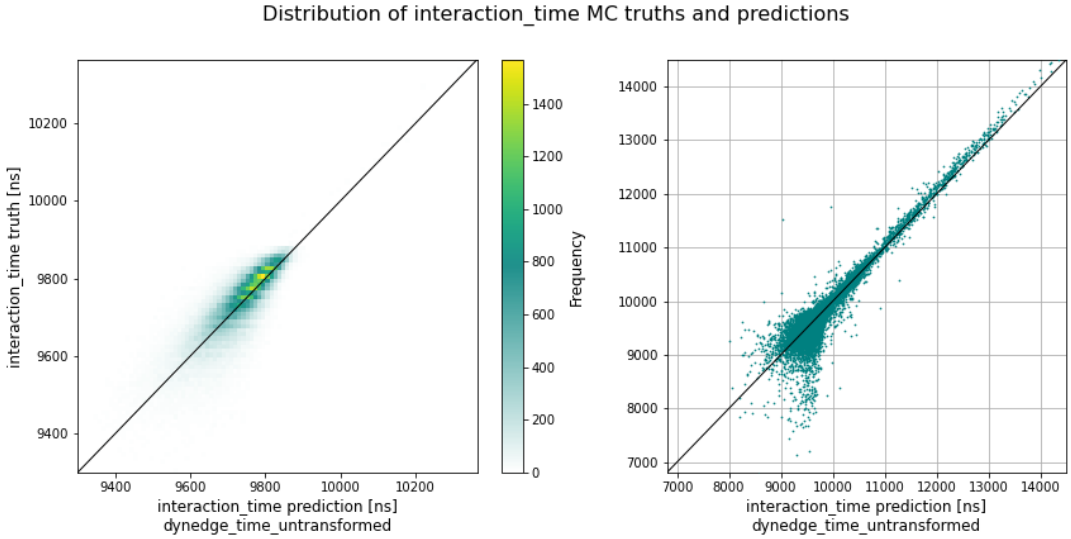


Figure 4.2: Distribution of the true and reconstructed interaction time, reconstruction by DynEdge with no transformation as a 2D histogram on the left and a scatter plot on the right. The diagonal black line indicates perfect reconstruction.

more from the optimal reconstruction.

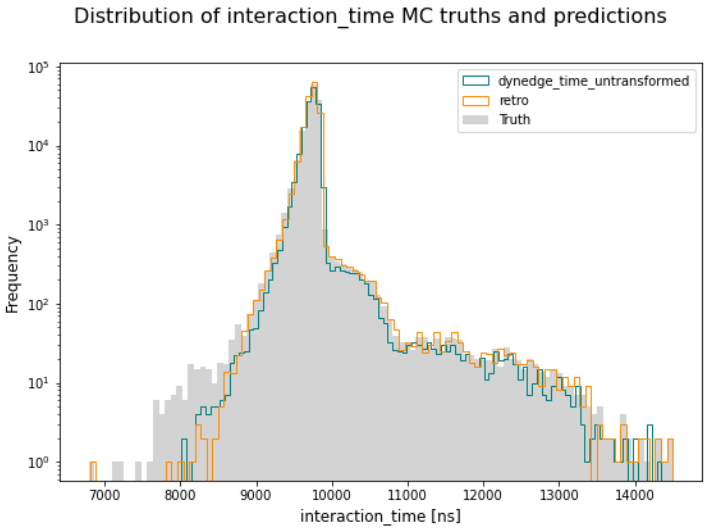


Figure 4.3: Distribution of the true and reconstructed interaction time, reconstruction by RetroReco and DynEdge with no transformation as a 1D histogram.

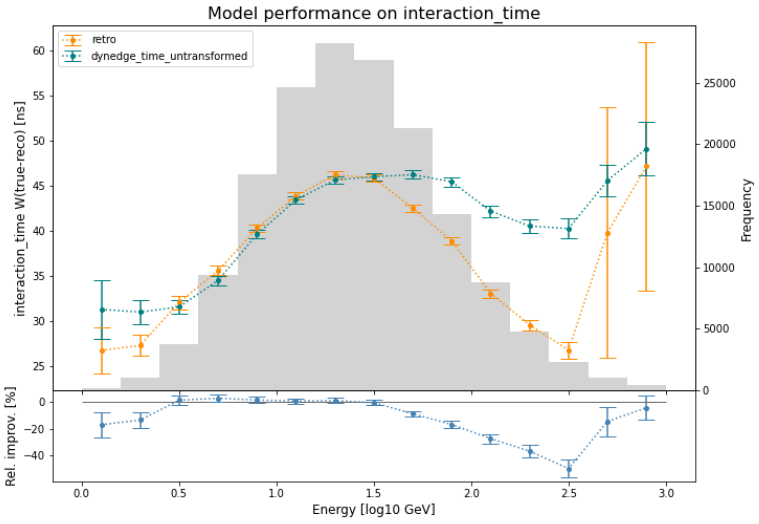


Figure 4.4: Width W of the error distribution of reconstruction interaction time by by RetroReco and DynEdge with no transformation separated by energy. gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from RetroReco to DynEdge.

This difference becomes more pronounced when looking at the error distribution. Figure 4.4 shows the width of the error distribution as described in Section 3.6 divided into energy bins to show the performance across different energy ranges. A histogram of the events in each energy bin is shown as gray bars in the background to indicate the amount of the available to the model in each energy range. At the bottom of the figure is shown the relative improvement in performance (or decrease in this case) between the two reconstructions. For

a few of the energy ranges (around 0.5 - 1.5 log10 GeV) the GNN is on par with Retro, but for all others it is significantly outperformed.

The poor performance of the GNN was believed to be caused by the irregular distribution of the interaction time variable, so I set out to apply various transformations to the target variable with to hope of improving the reconstruction[1].

[1] For the training of the models in this section, a pure sample of simulated muon neutrino events from the OscNext Level 7 data was used, and each model was trained using the LogCosh loss function.

4.2 Results

Two different transformations were applied to the interaction time variables, the **RobustScaler** and the **QuantileTransformer**, both described in Section 3.5.2. For each transformation, the model was retrained on the data with the scaled target, and the performance of each model is described in the following.

4.2.1 RobustScaler

Figure 4.5 shows the distribution of the true and reconstruction interaction time in the same way as Figure 4.2 but this time for a model trained using the RobustScaler. Figure 4.6 shows the 1D distribution. I many regards, the distributions look very similar to the distributions of the unscaled reconstructions.

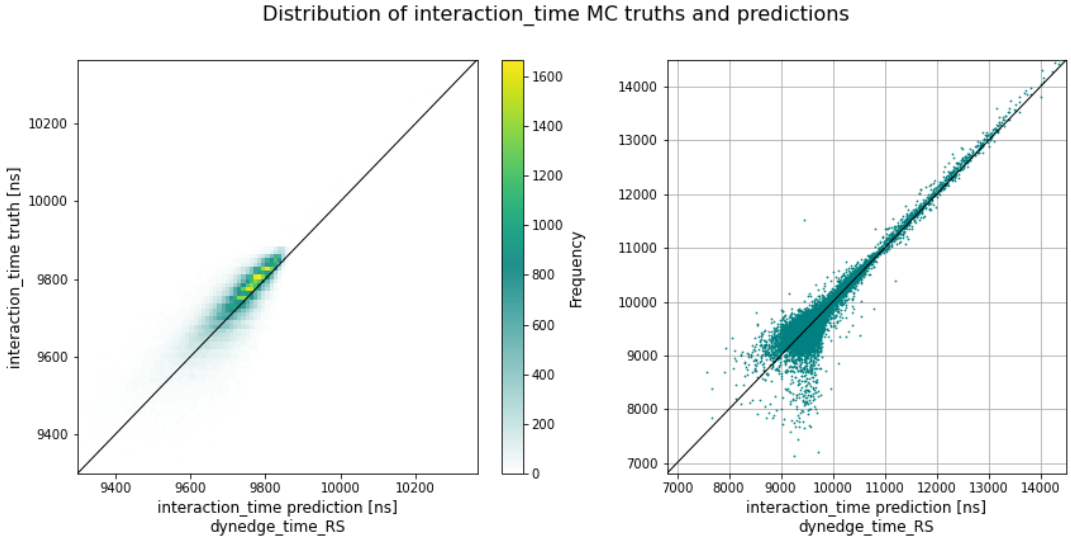


Figure 4.5: Distribution of the true and reconstructed interaction time, reconstruction by DynEdge with RobustScaler transformation as a 2D histogram on the left and a scatter plot on the right. The diagonal black line indicates perfect reconstruction.

From the 2D distribution, it does seems that the distribution is slight offset from the black tine towards the top left corner of the plot. When looking at the width of the errors distribution in Figure 4.7, the GNN seemed to have gained some ground on the Retro reconstruction, especially in the low energy regime (in which we are the most interested)

and in the bins with more data (from which we expect better performance).

Evidently, the model seems to gain some improvement in performance, but seems to be sacrificing some events in order to obtain optimal average performance.

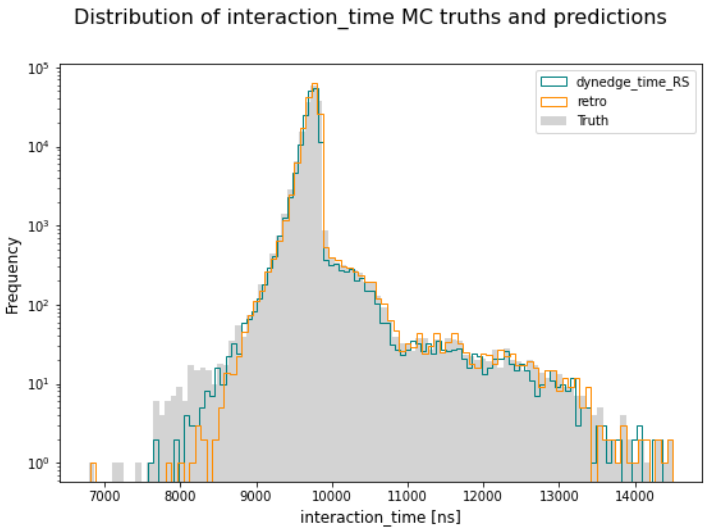


Figure 4.6: Distribution of the true and reconstructed interaction time, reconstruction by RetroReco and DynEdge with RobustScaler transformation as a 1D histogram.

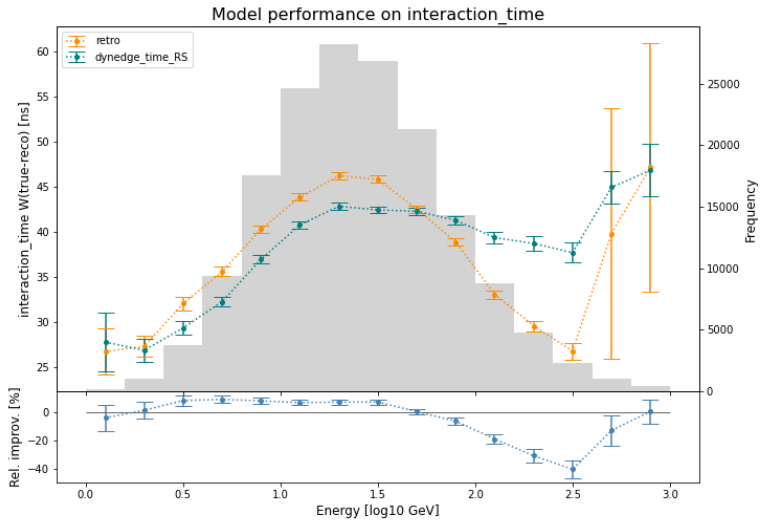


Figure 4.7: Width W of the error distribution of reconstruction interaction time by by RetroReco and DynEdge with RobustScaler transformation separated by energy. gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from RetroReco to DynEdge.

4.2.2 QuantileTransformer

As mentioned in Section 3.5.2, the QuantileTransformer is a non-linear transformation which scales various ranges of the distribution differ-

ently, such that the transformed distribution is approximates a normal distribution. To illustrate this, Figure 4.8 shows the unscaled interaction time distribution as well as the distributions after applying the RobustScaler and QuantileTransformer, respectively.

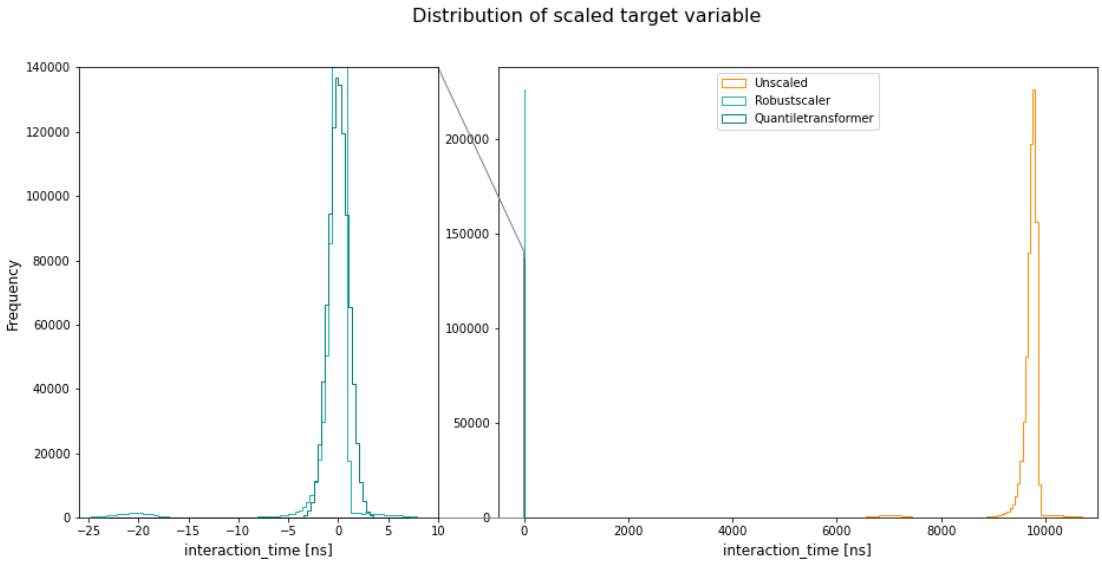


Figure 4.8: The distribution of the interaction variable before transformation and after transformation with RobustScaler and QuantileTransformer respectively. The left side of the figure shows an enlarged version of the range from -25 to 10.

On the left of the figure, an enlarged version of the range from -25 to 10 shows how the RobustScaler – being a linear transformation – precisely follows the shape of the original distribution, while the QuantileTransformer produces a bell shaped distribution. In both plots, a small but significant amount of events form an irregular *island* of outliers near 7000 for the unscaled distribution and -20 for the RobustScaler. These outliers may be hard for the LogCosh loss function to handle, and are not present in the QuantileTransformer distribution.

When looking at the distribution of reconstructions using the QuantileTransformer, it initially looks even more irregular than before the transformation. The scatter plot in Figure 4.9 displays an irregular shape that is quite different from the unscaled and Retro distributions, and in the 1D histogram in Figure 4.10, the reconstruction looks significantly worse than the previous results. But as mentioned previously, the 1D histogram does not show the full truth, and when looking on the left of Figure 4.10, the distribution actually lies closer the optimal black line than in the previous cases.

When looking at the width of the error distribution in Figure 4.11, this result is confirmed. The error distribution is narrower for the GNN reconstruction than for the Retro reconstruction in each energy bin from 0 to around 1.7 log10 GeV and several bins show between 10 and 20 % relative improvement.

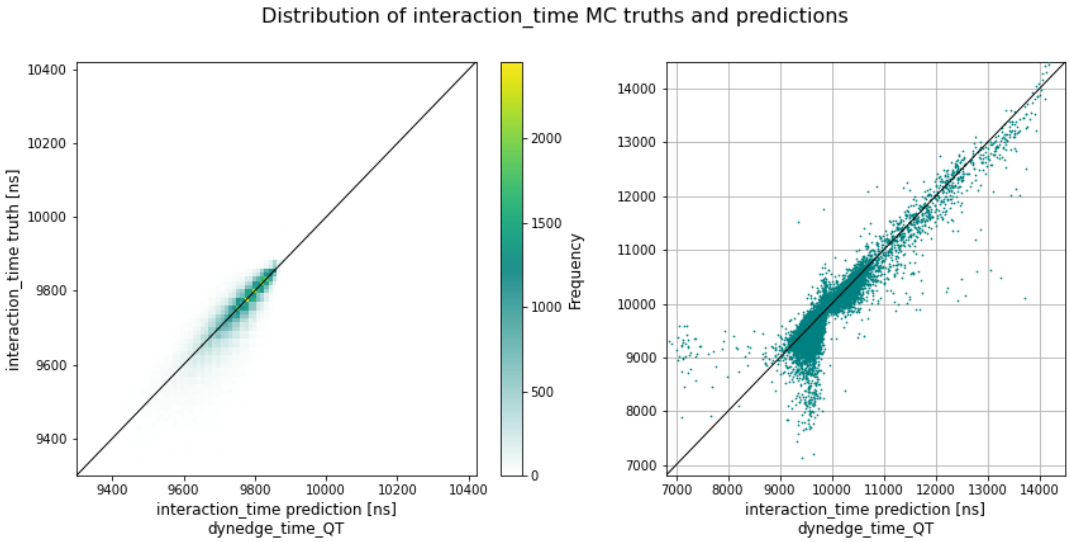
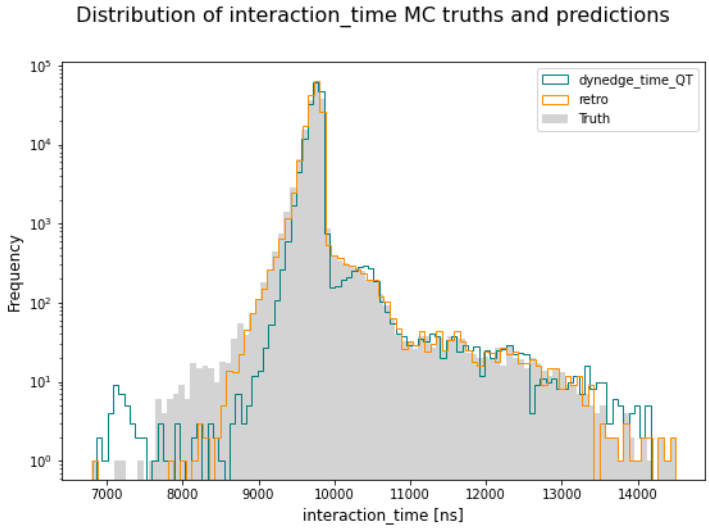


Figure 4.9: Distribution of the true and reconstructed interaction time, reconstruction by DynEdge with QuantileTransformer transformation as a 2D histogram on the left and a scatter plot on the right. The diagonal black line indicates perfect reconstruction.

Figure 4.10: Distribution of the true and reconstructed interaction time, reconstruction by RetroReco and DynEdge with QuantileTransformer transformation as a 1D histogram.



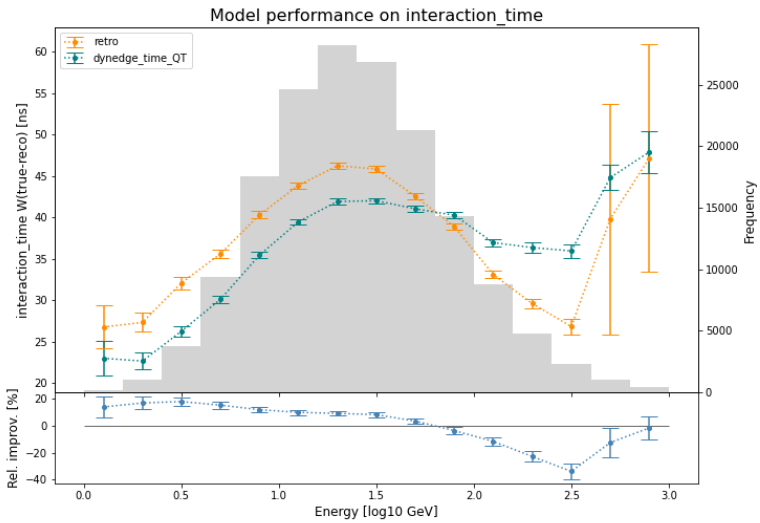


Figure 4.11: Width W of the error distribution of reconstruction interaction time by by RetroReco and DynEdge with QuantileTransformer transformation separated by energy. gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from RetroReco to DynEdge.

With these results I feel confident in concluding that a non-linear scaler is the best choice for the interaction time variable, and may be in other cases as well. This can cause problems in the context of machine learning, since it is not possible to take the derivative of a non-linear transformation, which means that backwards propagation of the loss function is not possible. And indeed this has lead to me implementing this transformation in a more permanent way in the GraphNeT framework, an effort which will not be documented here as it has been deemed to code-specific and would take too long to write up.

5 Event Level Classification

Contents

Note: The following work was made in collaboration with my fellow student Leon Bozianu. I was in charge of training and benchmarking the classifiers for noise/particles events and for track/cascade events. My fellow student was in charge of the classifiers for neutrino/muon events and for stopped/through-going muon events, which is not described here. I then used the first 3 classifiers to make the full cleaning pipeline described in Section 5.2.3, while my fellow student worked on muon events. The following plots and results are a product of my contributions to the project, with the exception of the borrowed neutrino/muon classifier.

5.1 Task

This chapter describes an attempt at showing how the current OscNext event cleaning pipeline described in Section 2.6.1 could be replaced by GNN event classification models. At the inception of this project, it was already established that the GNNs are able to classify IceCube events with very high accuracy, but it was unclear how implementing these classifiers would impact the OscNext event selection and what signal rate and purity would result from it. This work uses three GNNs trained to separate noise from particles, muons from neutrinos and track-like events from cascade-like events respectively, such that events that receive model scores above a chosen threshold for all classifiers make it to the final sample. The following sections describe the performance of the classifiers for noise/particles and track/cascade-

like events and the implementation of the full cleaning pipeline[1].

5.2 Results

5.2.1 Noise/Particle Classification

The noise/particle classifier aims to separate events caused by noise from the radioactive decay in the detectors from events involving neutrinos or atmospheric muons. Figure 5.1 shows the distribution of model scores for the classifier, and shows a very clear separation between noise and particles. For a more detailed picture, the same distribution is shown with a base 10 log scale on the y-axis in Figure A.4 in the Appendix.

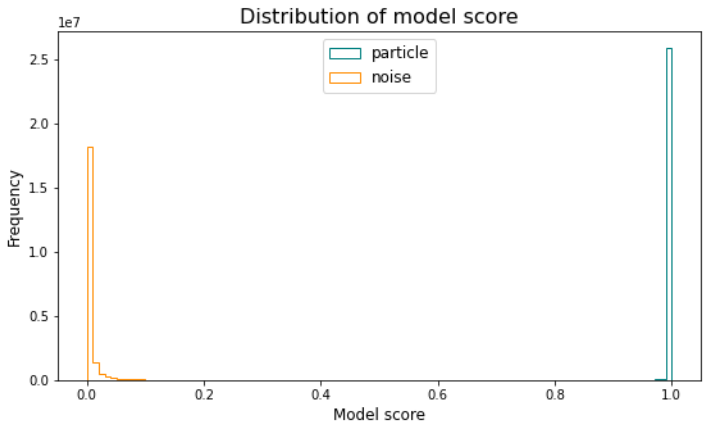


Figure 5.1: Distribution of model scores for noise/particle event classification for the GNN.

Figure 5.2 shows the ROC curve for the classifier, with the plot on the rights side using a base 10 log scale on the y-axis to show the finer details of the separation and confirms that the model does a good job at separating noise from particles. Selected thresholds are marked, for noise reduction factors of 10^4, 10^5 and 10^6, showing that the model allows for more than 90% of the particle events to be preserved when allowing only 10^{-6} of the noise events to make it into the final sample, and even more particle at still reasonable false positive rates.

Some important points to mention about these figures (that are also valid for the figures in the subsequent section) is that firstly, since the OscNext analysis usually does not select events in this way, no alternative model or method is shown alongside the GNN for comparison, making the results somewhat invalid without context. This will be remedied in Section 5.2.3. Secondly, since the desired outcome of the classifier is to have a pure sample with most of the physics remaining, one could have shown i precision-recall curve alongside these figures. But such a plot will vary if the composition of the initial sample varies, and as this will be relevant in the following sections, the ROC has been

[1] For the training of the noise models in this sections, a sample of simulated events consisting of 50% noise and 12.5% each of muons, electron neutrinos, muon neutrinos and tau neutrinos from the appropriate OscNext level was used. And for the training of the noise models in this sections, a sample of simulated events consisting of 50% muon neutrino CC and $33\frac{1}{3}$% each of muon neutrino NC, electron neutrinos CC + NC and tau neutrinos CC + NC was used. Each model was trained using the Binary Cross Entropy loss function.

chosen as the initial performance metric.

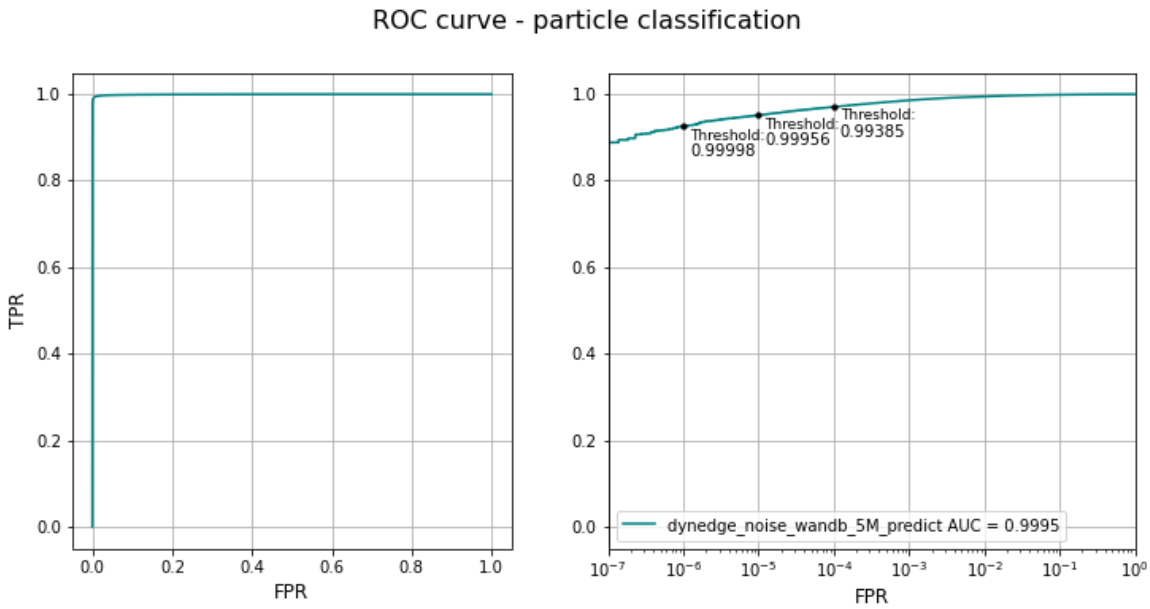


Figure 5.2: ROC curve for the noise/particle event classification model, with base 10 logarithmic x-axis on the right. Thresholds indicated for FPR = 10^{-3}, 10^{-4} and 10^{-5}.

5.2.2 Track/Cascade Classification

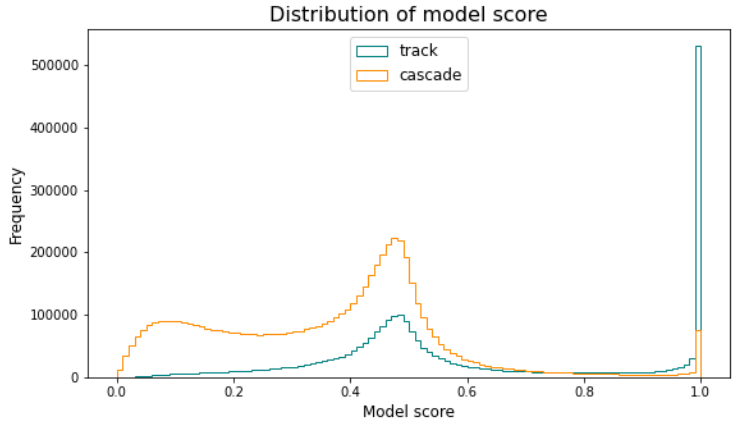


Figure 5.3: Distribution of model scores for track-like/cascade-like event classification for the GNN.

The track/cascade classifier aims to separate neutrino events into track-like events and cascade-like events. Figure 5.3 shows the distribution of model scores for the classifier, and illustrates how track/cascade classification is a significantly harder task for the model than noise/particle classification. A significant amount of both event types receive model scores around 0.5, which can be interpreted as the model being unsure about their type. However, the bins near each of the figure, have are significantly dominated by events of either the track or cascade type, meaning that it will be possible to obtain a reasonably pure sample at

the cost of some of the harder-to-classify events. For a more detailed picture, the same distribution is shown with a base 10 log scale on the y-axis in Figure A.5 in the Appendix.

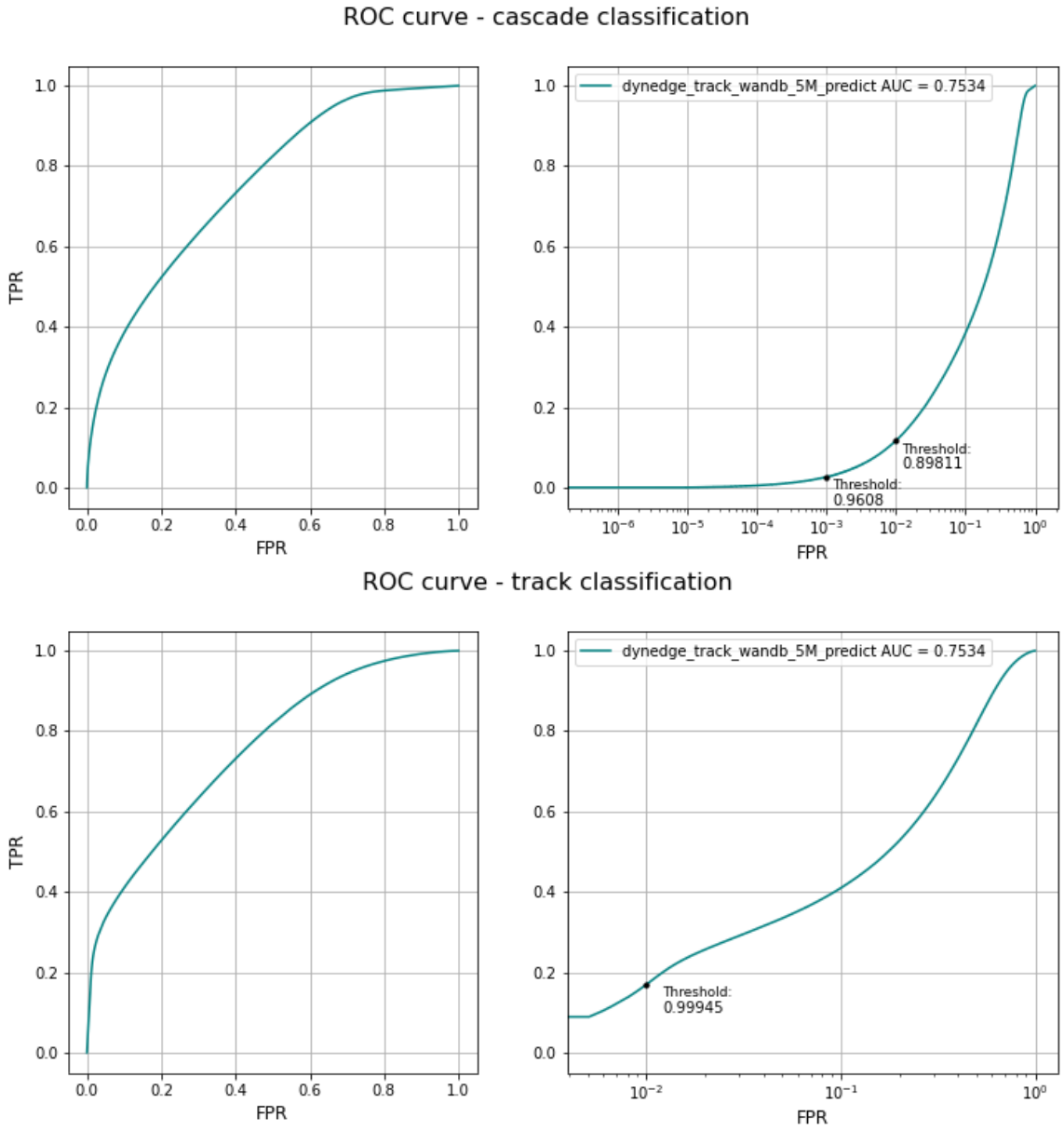


Figure 5.4: ROC curve for the track/cascade event classification model when selecting for track-like events (**top**) and cascade-like events (**bottom**), with base 10 logarithmic x-axes on the right. Thresholds indicated for FPR = 10^{-2} and 10^{-3} (**top**) and 10^{-2} (**bottom**).

Figure 5.4 shows ROC curves for the classifier. The top of the figure shows the ROC curve when desiring a pure sample of track-like events, and the bottom of the figure shows it from the perspective of desiring a sample of cascade-like events. In both cases, it is indeed possible to get rid of most events of the undesired type while keeping a significant portion of the desired events. The indicated thresholds show that one

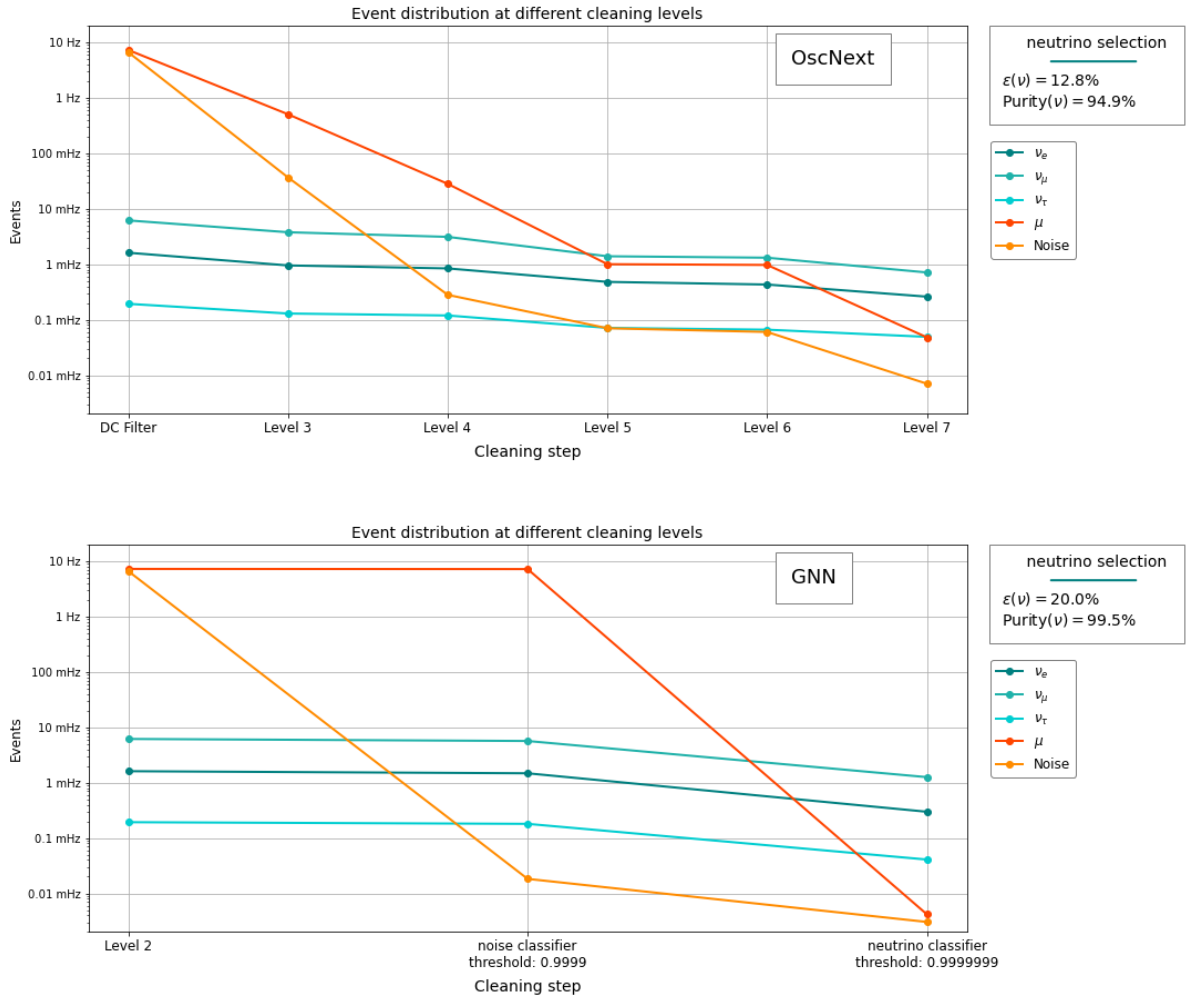


Figure 5.5: Distribution of events at different steps of the IceCube OscNext cleaning process (**top**) and the GNN cleaning process (**bottom**). To the right of each plot is indicated the resulting efficiency ($n_{\text{neutrinos,end}}/n_{\text{neutrinos,start}}$) and purity ($n_{\text{neutrinos,end}}/n_{\text{all,end}}$).

can keep ∼18% of the track events or ∼16% of the track events and remove 99% of the other type. With the high event rates of IceCube, losing ∼84% of the data is no catastrophe, as long as the sample is reasonably pure.

5.2.3 Full Cleaning Pipeline

This sections covers the discoveries found, choices made, and results obtained when attempting to replace the OscNext event cleaning pipeline with GNN classifiers. Figure 5.5 shows the current cleaning levels at the top and the initial attempt at the bottom, and is meant to serve as a baseline or proof-of-concept[2].

[2] The top plot is functionally identical to Figure 2.7, differing only in style.

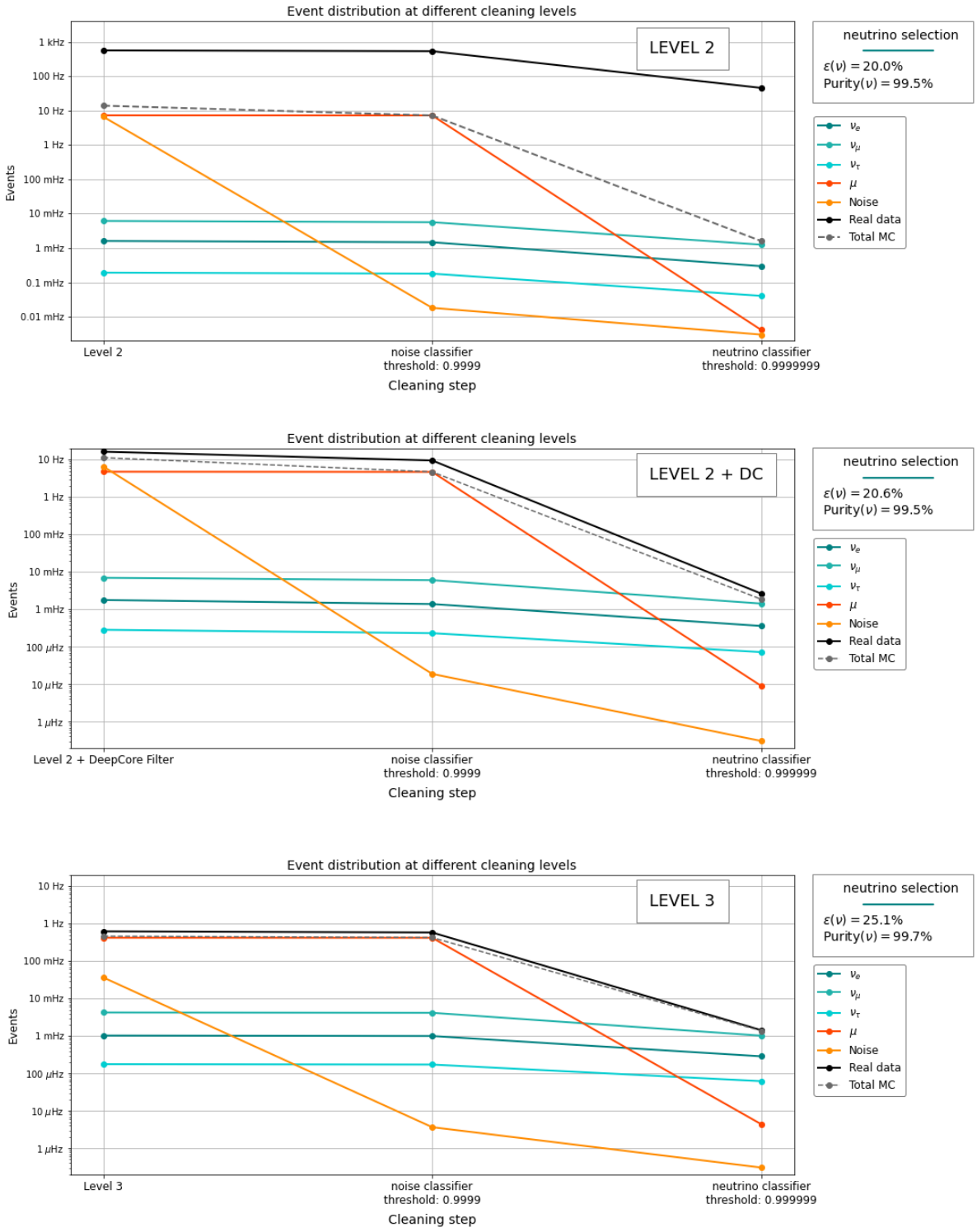


Figure 5.6: Distribution of events at different steps of the GNN cleaning process using data from OscNext Level 2 (**top**), Level 2 with DeepCore filter (**center**) and Level 3 (**bottom**). To the right of each plot is indicated the resulting efficiency ($n_{\text{neutrinos,end}}/n_{\text{neutrinos,start}}$) and purity ($n_{\text{neutrinos,end}}/n_{\text{all,end}}$).

From left to right is shown the different cleaning steps, (with the chosen model score threshold in the case of the GNNs), and on the y-axis is shown the amount of events that passes to a certain level, with the event types shown in different colours. On the right is shown the efficiency and the purity at the final cleaning level. The rates are scaled to match the rates of the data stream in IceCube at Level 2. Comparing to the ROC curves of the previous chapter, these numbers significantly worse, but this is due to the high rates of noise and muons in IceCube which means that removing $\sim$ 99.9999% of noise and muons only results in $\sim$ 99.5% purity. Still however, the GNNs preserve more neutrinos and remove more of the other events than the current method.

The next leg of this investigation is to compare the found rates to the rates of real data, an endeavour that yields some interesting results. Figure 5.6 shows the same type of plots as Figure 5.5, but only for the GNNs and with 3 different starting data. Like Figure 5.5, the top plot uses data from OscNext Level 2, while the bottom plot uses OscNext Level 3 data. The middle plot also uses Level 2 data, but with the DeepCore filter described in Section 2.6.1. Additionally, the rates of data that passes through the two cleaning models is shown in black, and the total rate of MC data is shown in dashed gray. The reason for this choice in data is that the cuts made at Level 3 are specifically aimed at agreement between real data and MC simulation. The Level 2 DeepCore is shown both because it can be viewed as the middle ground between Levels 2 and 3, and because it is actually where the OscNext pipeline illustrated in Figure 2.7 starts. And indeed, the plots show increasingly better agreement between real data and simulation from top to bottom, with the real data (solid black) following the simulation (dashed gray) closer with each figure. This is admittedly a simple way to evaluate data/MC agreement, and more rigorous testing (like regression of key variables) can be performed to quantify the agreement, but the results shown here are enough to support the decision use Level 3 data in the final part of this section. The models applied at Level 3 also exhibit the best performance, with an efficiency of 25.1% yielding a purity of 99.7%.

In this final part of this section, the track/cascade classifier described in Section 5.2.2 is added to the GNN cleaning pipeline. Figure 5.7 shows similar plots to Figures 5.5 and 5.6, including the real data and total simulation, but this time with the additional track/cascade classifier applied at the end. As both track and cascade-like events can be of use in IceCube and the value lies in separating the two, the top plot of Figure 5.7 shows the results when selecting for track-like events, and the bottom plot of Figure 5.7 shows the results when selecting for cascade-like events. As would be expected from the results

in Section 5.2.2, this makes for a harder task than the noise/particle and muon/neutrino separation. For track-like events, this is remedied somewhat by these events being the most common type, allowing for a purity of 99.7% at the cost of keeping only 7.3% of the events, which as mentioned earlier is not necessarily as bad as it sounds. The downside of the high rate of track-like events is that this also makes obtained a pure sample of cascade-like events quite hard. Here it is shown that when keeping 1.5% of events, one can get a purity of 64.9%. Higher purity will come at the cost of even lower event rates.

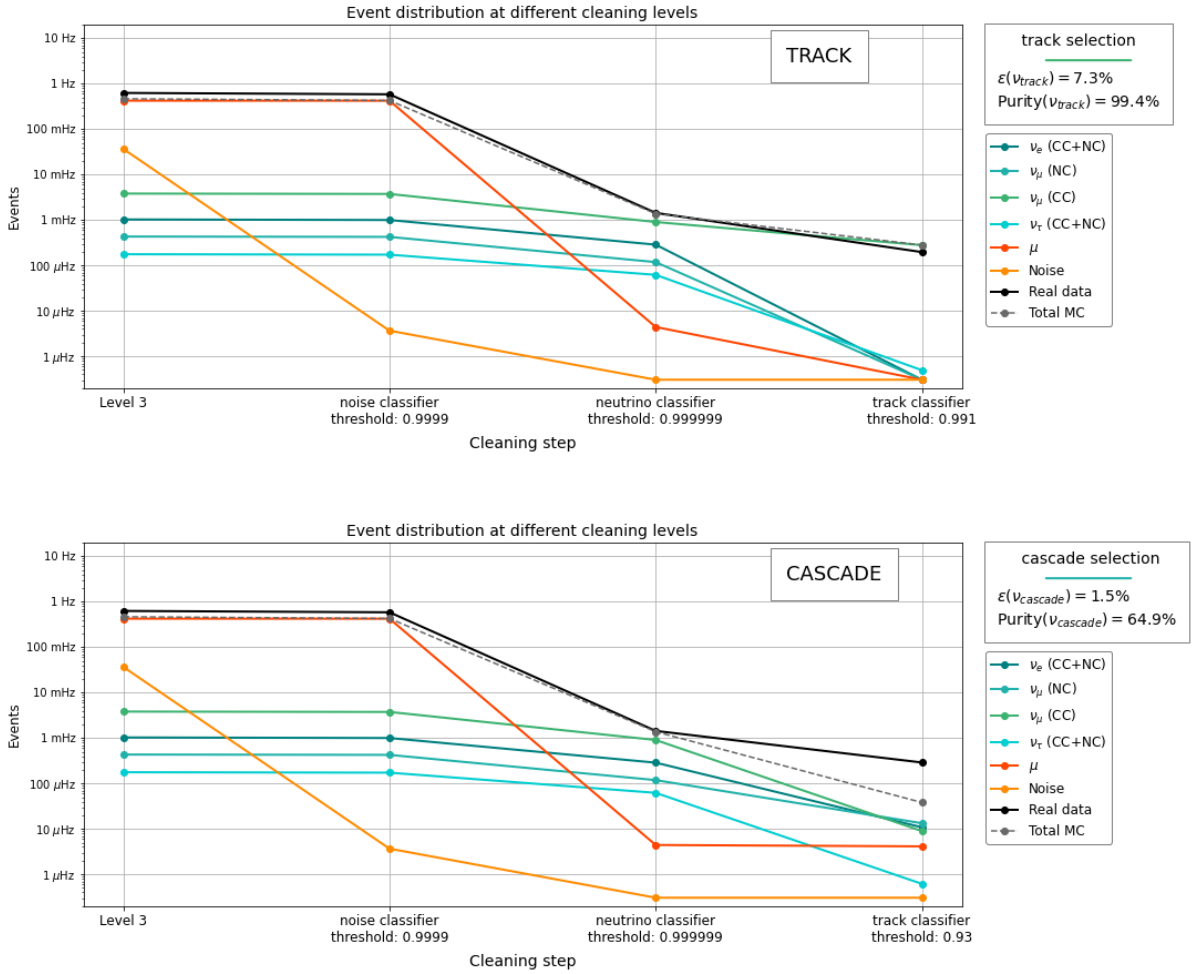


Figure 5.7: Distribution of events at different steps of the GNN cleaning process using data from OscNext Level 3 including the track/cascade classifier selecting for track-like events (**top**) and cascade-like events (**top**). To the right of each plot is indicated the resulting efficiency ($n_{\text{neutrinos,end}}/n_{\text{neutrinos,start}}$) and purity ($n_{\text{neutrinos,end}}/n_{\text{all,end}}$).

It is also worth noting, that for cascade-like events, the agreement between data and simulation is not very convincing at the final cleaning level. With knowledge of the underlying data however, this is not as alarming as it looks. As mentioned earlier, the real data is a full day

of IceCube events of which 1.42 million pass the deepcore filter, and only 54170 events make it to Level 3. Using the expected rates from the OscNext analysis[3], the average number of track and cascade-like events in one day of events would be 326 and 139 respectively. And as the efficiency of the pipeline including the cascade selection is 1.5%, only 2 muon neutrino CC events will on average make it to the final sample. With amount of statistics, the real data at the track/cascade level is very susceptible to fluctuations in the neutrino event rate, and does not have any practical value.

[3] See [25].

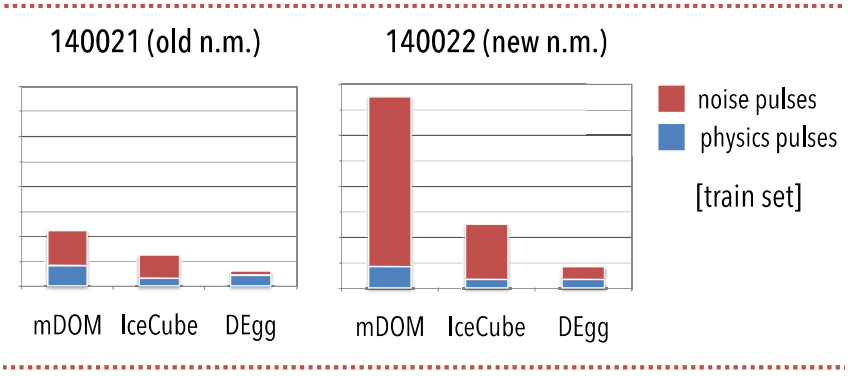
Regardless it is clear from these plots that GNNs have the potential to replace the current OscNext data cleaning levels. In addition to increased efficiency and purity, using neural networks will, as mentioned earlier, provide an increase in processing speed. Implementing GNN cleaning at Level 3 instead of Level 2, should not impact the performance speed significantly, as the cuts used are not the computationally demanding parts of the OscNext cleaning process.

6 Upgrade Pulse Level Cleaning

Contents

Note: The following work was made in collaboration with my fellow student Morten Holm. I was in charge of implementing the functionality in the GraphNeT repository that made it possible to do node-level classification, and produced the first naive attempt at pulse cleaning. My fellow student then took over further improvement and development of the pulse cleaning while I used the initial cleaned events to make an attempt at event reconstruction as described in Section 7. Unless otherwise stated, the following plots and results are a product of my contributions to the project.

6.1 Task



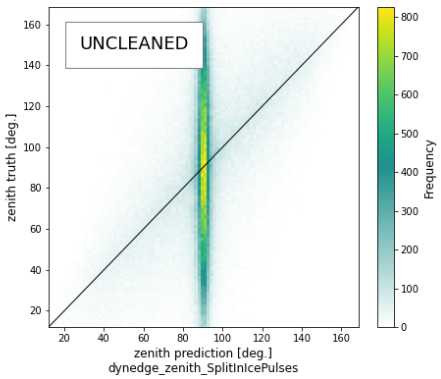


Figure 6.1: **Left:** Noise and physics rates in the old and new noise model for each DOM type in the IceCube Upgrade. *Figure from* [98]
Right: Attempt at reconstructing the zenith angle of noisy simulated Upgrade events.

IceCube Upgrade, the upcoming expansion of the IceCube detector array described in Section 2.3.2 will be born with a significant birth defect. Due to a change in the production methods at Hamamatsu Photonics – The company that manufactures the IceCube PMTs – the

PMTs are contaminated with radioactive isotopes during production, the decays of which result in significantly higher noise rates than in the original IceCube array[99].

On the left of Figure 6.1 is shown the difference in noise rates between the old noise model and the new noise model (simulations without and with the high noise rates of the new PMTs, respectively) on a dataset level. These new noise pulses completely dominate each event, and have a significant negative impact on the reconstruction performance. On the right side of Figure 6.1, an attempt at reconstructing the zenith angle from these new noisy pulses is shown. Evidently, the model predicts all events to have a zenith angle around 90 degrees[1].

Several attempts have already been made at a more rigorous cleaning of these noise pulses, using either changes to the current reconstruction algorithms[100] or a rudimentary GNN[98]. The following describes my efforts to improve pasts results by using the more sophisticated DynEdge architecture to clean the Upgrade pulses. This has been done on a simulation of only muon neutrinos using the new noise model[2].

[1] This is the most common zenith angle, and indicates that the model gains no information from the pulse information and learns to predict the most likely angle for all events to get optimal performance.

[2] For the training of the model in this section, a pure sample of simulated muon neutrino events from Step 4 of the Upgrade event selection chain (which corresponds to Level 2 in the OscNext analysis chain described in Section 2.6.1)[101] was used, and the model was trained using the Binary Cross Entropy loss function.

6.2 Results

Figure 6.2 shows the the distributions of model decision scores for noise and physics for both the GraphNeT pulse cleaning (this attempt) and **GraphSAGE** pulse cleaning which is the simple GNN used in [98].

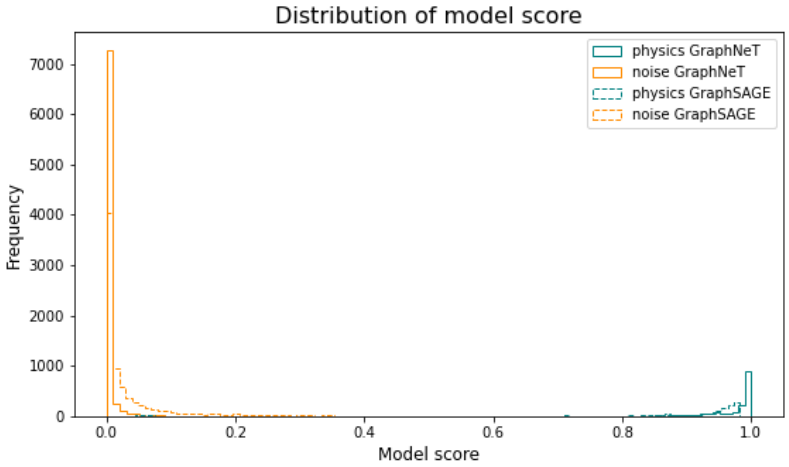


Figure 6.2: Distribution of model scores for noise and physics pulses for the GraphNeT and GraphSAGE pulse cleaning models.

Both cleaning models are very confident in their predictions, but the GraphNeT model is slightly more precise for both noise and physics as evident by areas bounded by the dashed lines to the right of 0.0 and the left of 0.1. For further illustration of the differences, Figure A.6 in the Appendix shows the same figure but with a base 10 log scale on the y-axis.

Figure 6.3 shows the ROC curves of the GraphNet and GraphSAGE noise classifiers. The ROC curves on the left shows a significant improvement from the GraphSAGE to the GraphNeT model, which have an AUC of 0.9755 and 0.9974, respectively.

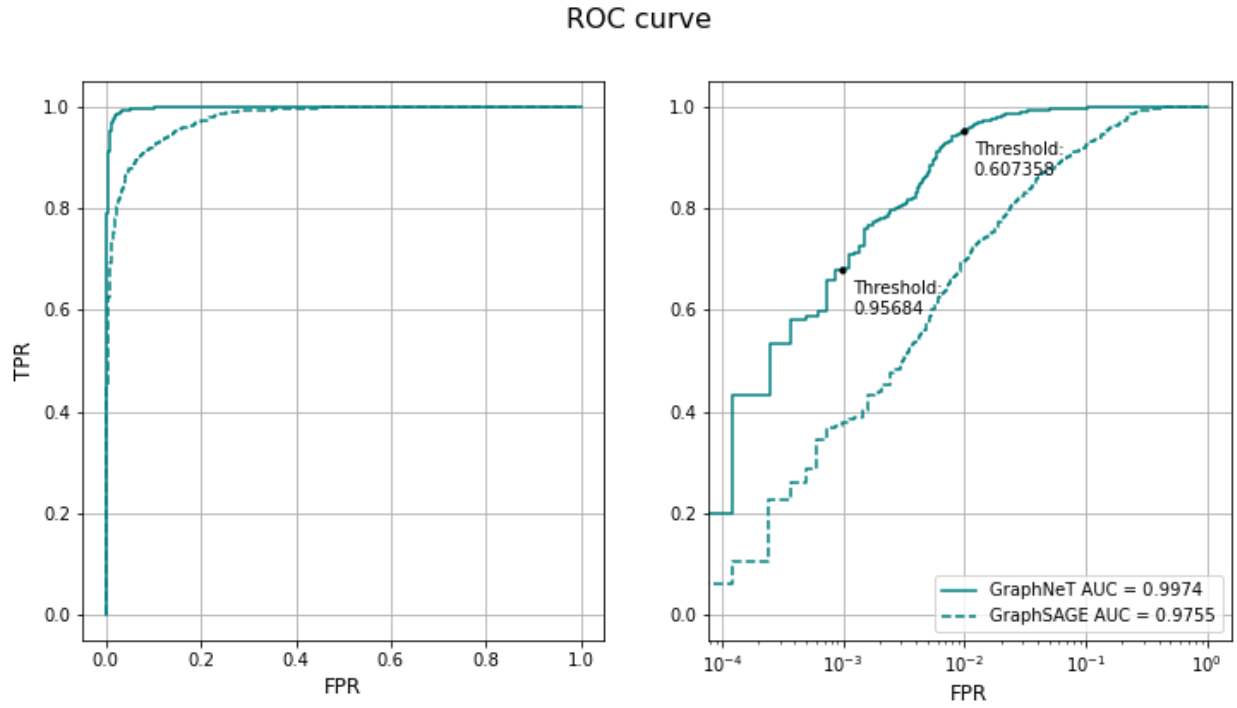


Figure 6.3: ROC curve for the GraphNeT and GraphSAGE pulse cleaning models, with base 10 logarithmic x-axis on the right. Thresholds indicated for FPR = 10^{-2} and FPR = 10^{-3}.

On the right is shown the same ROC curves but with a base 10 log scale on the x-axis which illustrates how 99% of the noise can be removed (FPR = 10^{-2}) while keeping ∼95% of the physics pulses, or 99.9% of the noise can be removed (FPR = 10^{-3}) while keeping ∼68% of the physics pulses.

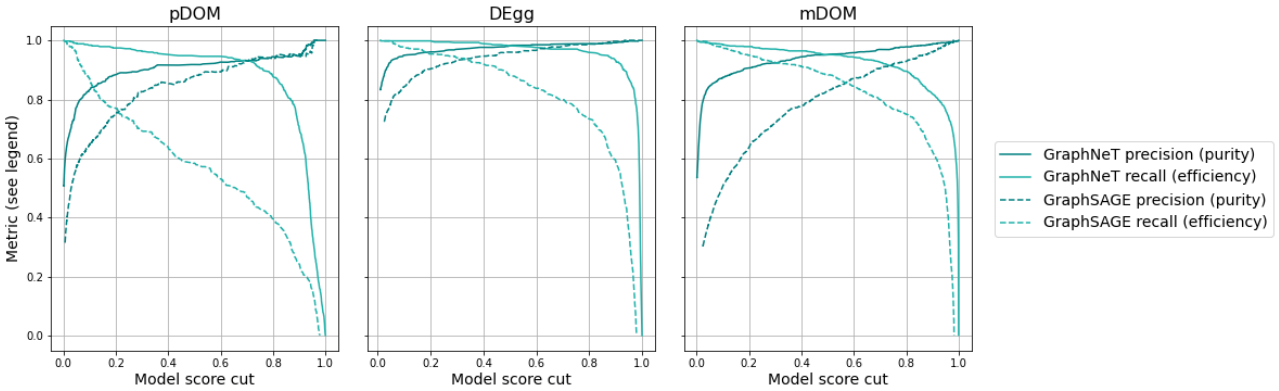
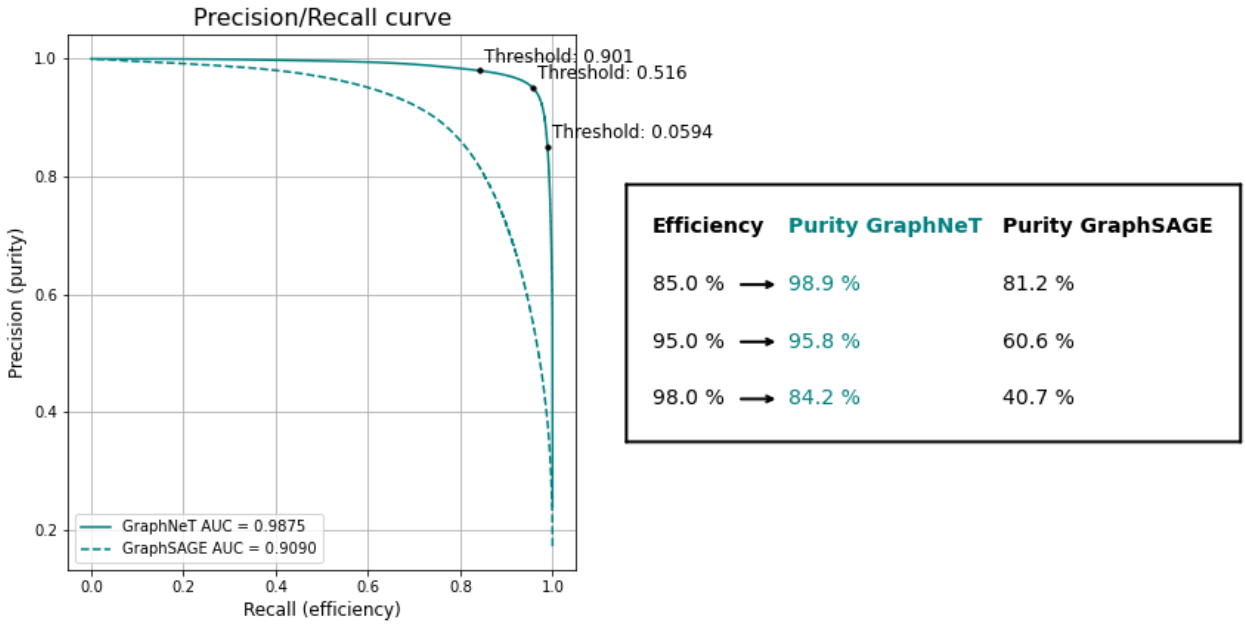


Figure 6.4: Precision and recall scores of the GraphNeT and GraphSAGE pulse cleaning models as a function of the model score cut for each DOM type in the IceCube Upgrade.

Scoring the model performance according to the prior distributions

(which is what the FPR does), however, may not be suitable, when the amount of noise pulses is so much greater than the amount of physics pulses. Here the precision-recall curve is used instead as a better metric that gives an idea of the amount of noise in the final sample.

Figure 6.4 shows the precision and recall score of both the GraphNet and GraphSAGE models for each of the different DOM types. While both models struggle with classifying both the mDOMs and the original IceCube pDOMs, the GraphNet model seems to reach a satisfying equilibrium around 0.6 for all DOM types.



Efficiency	Purity GraphNeT	Purity GraphSAGE
85.0 % →	98.9 %	81.2 %
95.0 % →	95.8 %	60.6 %
98.0 % →	84.2 %	40.7 %

Figure 6.5: Precision-recall curves for the GraphNeT and GraphSAGE pulse cleaning models. The table on the right indicates the purity (precision) scores resulting from different choices in efficiency (recall).

Figure 6.5 shows the total precision-recall curves for both models, and illustrates how different target efficiencies (recalls) lead to higher purities (precisions) with GraphNeT than GraphSAGE. For the following work, the data has been cleaned using a model score cut of 0.439 which yields a sample with 95.8% physics while keeping 95.0% of the original physics pulses.

To investigate the composition of this final sample and the effect of the pulse cleaning on each event, the distribution of the number of physics and noise pulses in each event in the original sample (called **SplitInIcePulses**) and the sample created in this work (called **GraphNeTPulses**) is shown on the left in Figure 6.6. The figure shows that the distribution of physics remains intact from uncleaned to cleaned, save for a selection of events with a lower number of pulses (below 10) seemingly disappearing into the bin closest to zero. Meanwhile, the noise pulses go from a wide bell curve distribution centered around

~100 in the uncleaned sample to a narrower bell curve distribution centered around zero in the cleaned sample[3].

[3] Notice that the plot cuts at $y = 1.5 \cdot 10^6$ to better show the less populated bins.

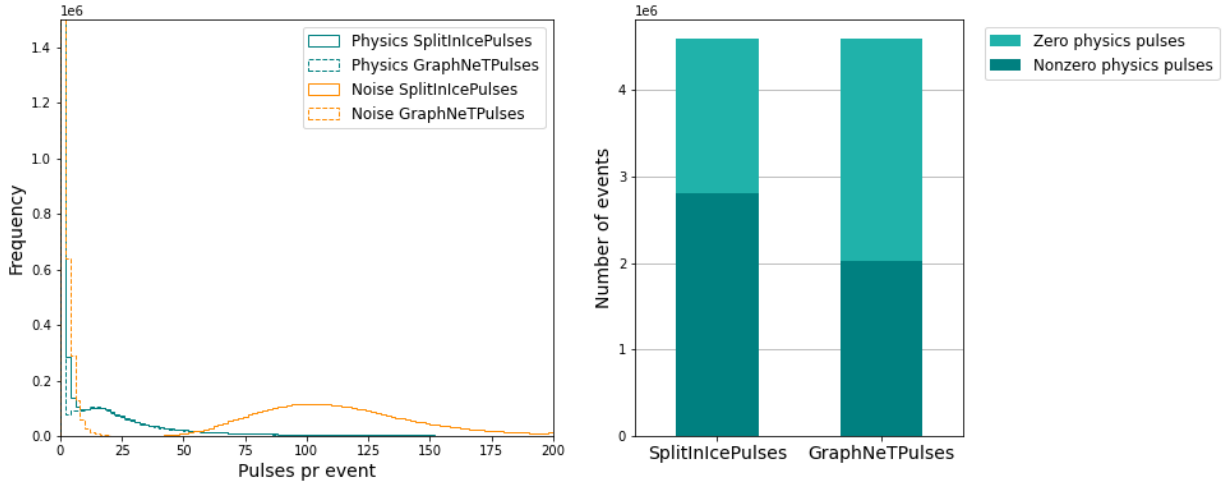


Figure 6.6: **Left:** Distribution of the number of physics and noise pulses in each event the original sample and the cleaned sample. **Right:** Ratio of events with zero and more than zero physics pulses in each event the original sample and the cleaned sample.

Both noise and physics have most of the events in the bin closest to zero, indicating that for a lot of events, all pulses may be removed in the cleaning. Thus, on the right of Figure 6.6 is shown the ratio of events with zero physics pulses in the cleaned and uncleaned sample, showing that many of the events in the leftmost bin to indeed contain zero physics pulses.

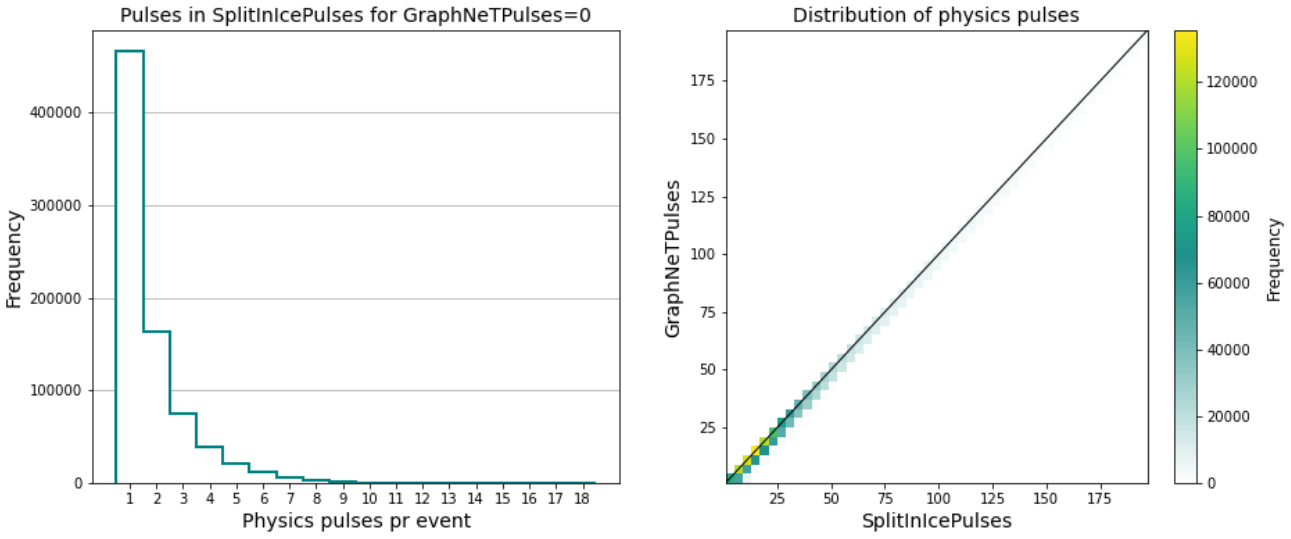


Figure 6.7: **Left:** Distribution of physics pulses in each event in the original sample the has zero physics events in the cleaned sample. **Right:** Distribution of physics pulses in each event in the original and cleaned samples.

To make certain that the events that end up with zero physics pulses

are only events that have a few physics pulses to begin with, the plot on the left of Figure 6.7 is produced. It shows the distribution of physics pulses in the uncleaned sample for all events that that have zero physics pulses in the cleaned sample. Fortunately, the vast majority of these events have less than 5 physics pulses in the uncleaned sample.

In addition, the plot on the right of Figure 6.7 shows a more generalised study of the behavior of physics pulses in the sample. With number of physics pulses in the uncleaned sample on the x-axis and number of physics pulses in the cleaned sample on the y-axis, it shows how the majority of events retain a number of physics pulses close the original amount, and that no bins see a significantly greater change than others. The black diagonal line indicates no change in physics distribution.

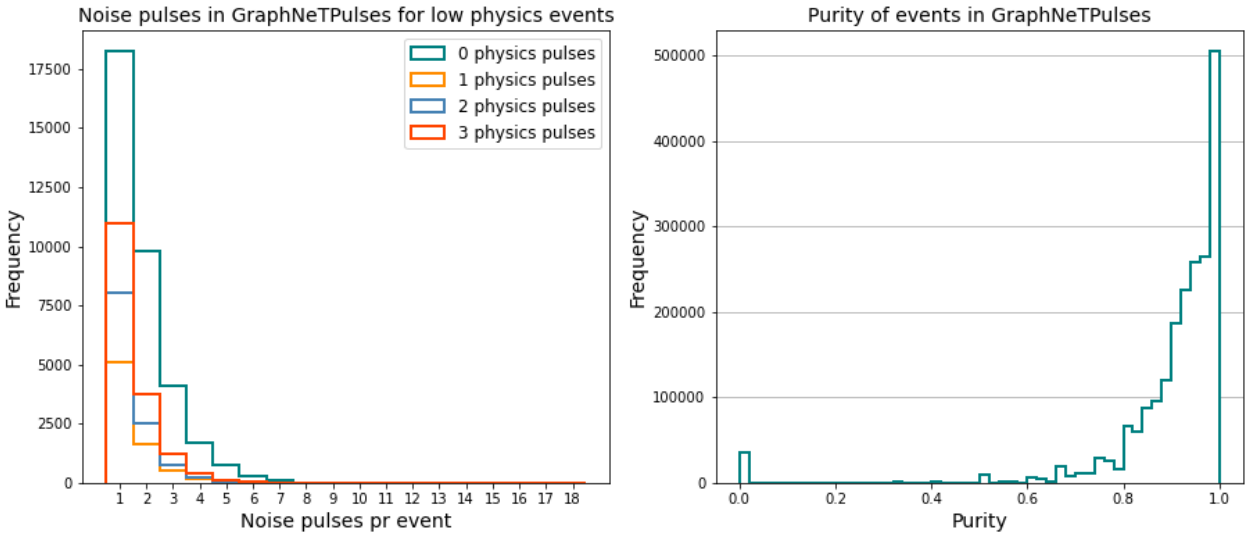


Figure 6.8: **Left:** Distribution of noise pulses in events in the cleaned sample with 0, 1, 2, or 3 physics pulses. **Right:** Distribution of purity of events in the cleaned sample.

Finally, to make sure that the number of events in the cleaned sample with few physics pulses but more noise pulses is low, the plots in Figure 6.8 are produced. The plot on the left shows the distribution of the amount of noise pulses in events in the cleaned sample with 0, 1, 2, or 3 physics pulses. Aside from the interesting observation the events with 0 and 3 physics pulses are more frequent than 1 and 2, the distributions show that the majority of the low physics events have very little noise as well. The plot on the right shows the distribution of purity of events in the cleaned sample, which is also reassuringly centered around 1, with very few events below 0.5. The relatively large bin at 0.0 can be interpreted as all events with zero physics pulses, which from the left of the figure is known to consist mostly of events

with one or a few noise pulses, which would likely be discarded when applying a simple event cleaning step.

To summarise, the GraphNeT pulse cleaning method performs better than the GraphSAGE model, and the resulting events display promising distributions of noise and physics. The relatively large amount of zero-physics pulses in Figure 6.6 should not be a concern as the majority contain very few pulses in total. The real test of the method is the reconstruction performance, which will be addressed in the upcoming section.

7 Upgrade Reconstruction

Contents

7.1 Task

Having obtained satisfying results for the pulse cleaning model, the real test of the cleaning method was to see if it actually impacts the reconstruction performance. In the context of Machine Learning, this is actually a quite interesting question. In theory, a sufficiently sophisticated model could be able to learn which pulses to trust and which to disregard. In practice however, in most cases the models benefit from some amount of human guidance or decision-making, in this case in the form of the choice to train a model to clean out the noise first, and then training a separate model for regression[1].

[1] This should provide some comfort to those who fear becoming subjects of AI overlords.

Three separate models were trained to regress three of the most commonly reconstructed variables used for benchmarking within the OscNext group: the energy, and the zenith and azimuth angles. The models were trained for 50 epochs with and converged between 20 and 30 epochs. The energy, the LogCosh loss of the log10 of the target and predictions was used, and for the zenith and azimuth, the 2D Von Mises-Fischer loss with error estimation was used.

7.2 Results

The models described in the following are trained and tested on three different datasets all containing muon neutrino simulations for the Upgrade detector:

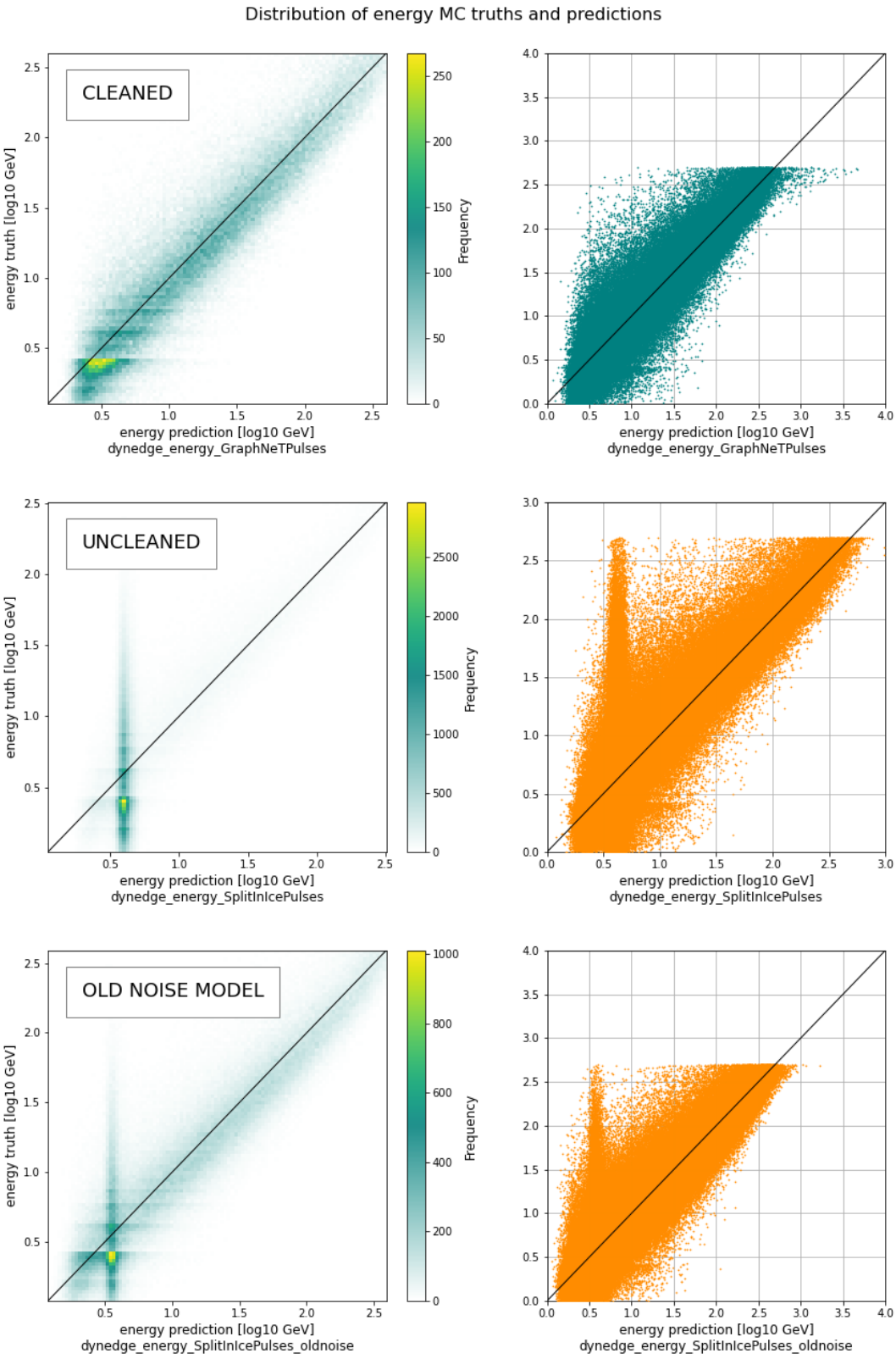


Figure 7.1: Distributions of the reconstructed energy by the models trained on the GraphNeTPulses (top), SplitInIcePulses (center) and SplitInIcePulses_oldnoise (bottom) datasets as 2D histograms (left) and scatter plots (right).

GraphNeTPulses: Simulations made using the **new noise model** (simulated with the more noisy PMTs) **cleaned** using the pulse GNN pulse cleaning described in Chapter 6,

SplitInIcePulses: The same simulations with **no pulse cleaning** to investigate the immediate effect of pulse cleaning on reconstruction performance

SplitInIcePulses_oldnoise: Corresponding simulations made using the **old noise model** to test if pulse cleaning makes reconstruction match the expected performance of Upgrade reconstruction.

All datasets are initially drawn from Step 4 of the Upgrade event selection chain, which corresponds to Level 2 in the OscNext analysis chain described in Section 2.6.1[101][2].

[2] For the training of energy models, the LogCosh Loss function was used, and for the azimuth and zenith angles, the von Mises-Fisher 2D loss function was used.

7.2.1 Energy

Figure 7.1 shows the reconstructed energy for GraphNeTPulses (top), SplitInIcePulses (center) and SplitInIcePulses_oldnoise (bottom) as a 2D histogram on the left and a scatter plot on the right with reconstructed energy on the x-axis and true energy on the y-axis. While each model predicts at least a fraction of the events within an acceptable range of the correct values (the ones that follow the diagonal black line in the scatter plots), both the SplitInIcePulses and SplitInIcePulses_oldnoise models predict the majority of the events to have the same energy ($\sim 0.6 \log_{10}$ GeV), with the former showing this tendency the clearest. This value can be interpreted as the rough mean of the energy distribution, for which the model will receive the smallest punishing during training if it is unable to make conclusions based on the input data.

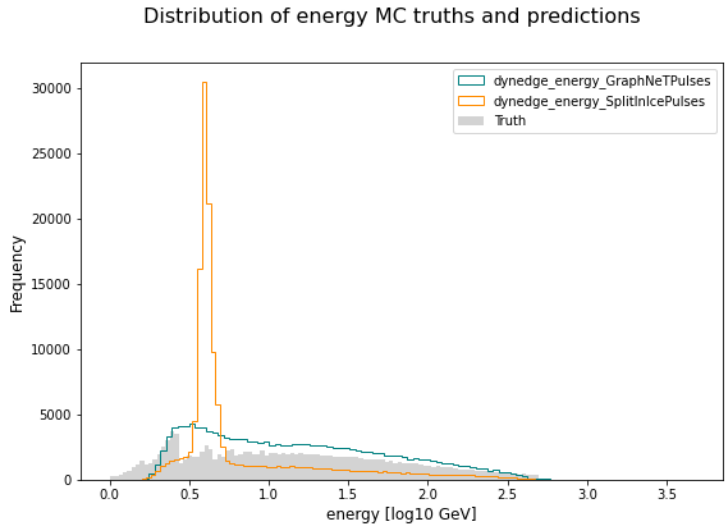


Figure 7.2: Distributions of the reconstructed energy by models trained on the GraphNeTPulses and SplitInIcePulses datasets with the true distribution in gray in the background.

Additionally, the 2D histogram shows a clear band structure in the true energy values. These values are caused by a bug in the simulation of the Upgrade events, which at the time of writing has recently been found and removed. During the final stages of this project, new Upgrade simulations without the bug were in progress, but they did not finish in time to make it into this thesis. The irregularities in the simulations will undeniably impact the finer points of training and comparison, but as it is not completely skewed, the conclusions made in the following are still valid.

In Figures 7.2 and 7.3, the GraphNeTPulses and SplitInIcePulses reconstructions are compared. Figure 7.2 shows a regular histogram of the reconstructed energy distributions with the truth in the background in gray, and confirms that the uncleaned pulse reconstructions has a spike around the mean of the distribution while the cleaned pulse reconstruction follow the true distribution more closely. The figure also clearly shows bands in true energy from the bugged simulation.

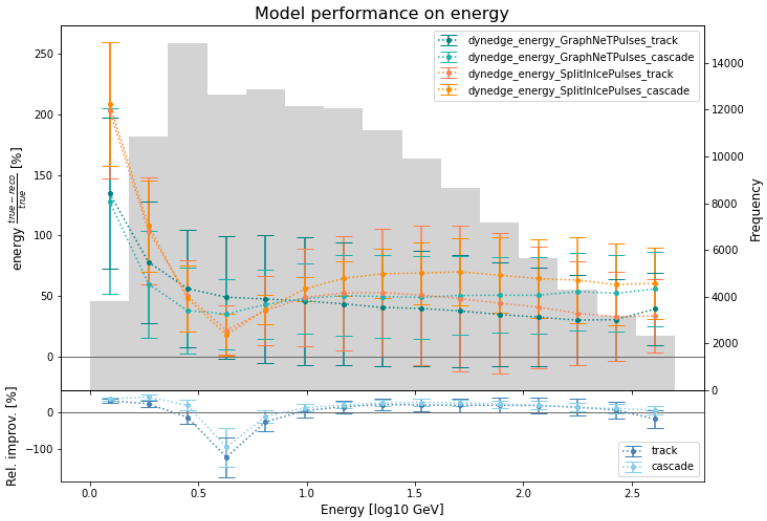


Figure 7.3: Mean of the error distribution of reconstructed energy by models trained on the GraphNeTPulses and SplitInIcePulses datasets separated by energy and split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from SplitInIcePulses to GraphNeTPulses.

Figure 7.3 shows the mean of the residual distribution ($\frac{\text{true}-\text{reco}}{\text{true}}$) of the energy in percent for different energy ranges, and separated in track- and cascade-like events. This makes for a quite complex plot, so a version without the track/cascade separation is shown in Figure A.7 in the Appendix. On the bottom of the plot is shown the relative increase in performance, indicating a significant improvement in all energy ranges except close to the $\sim 0.6 \log_{10}$ GeV range where the naive SplitInIcePulses model has an advantage (and a few bins in the high energy range which are not important to the OscNext analysis). Since the uncleaned model predicts roughly the same energy value for the majority of the events, the width of the error distribution is

not a good performance metric for this model and is not plotted in a separate figure.

Figures 7.4 to 7.6 compare reconstruction performance between the cleaned new noise model dataset and the uncleaned old noise model. Figure 7.4 shows the same energy peak for SplitInIcePulses_oldnoise as observed for SplitInIcePulses, but otherwise follows the true energy distribution.

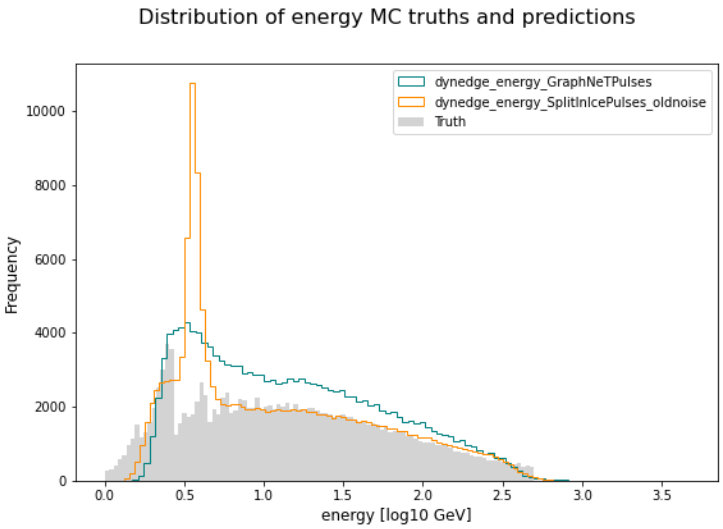


Figure 7.4: Distributions of the reconstructed energy by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets with the true distribution in gray in the background.

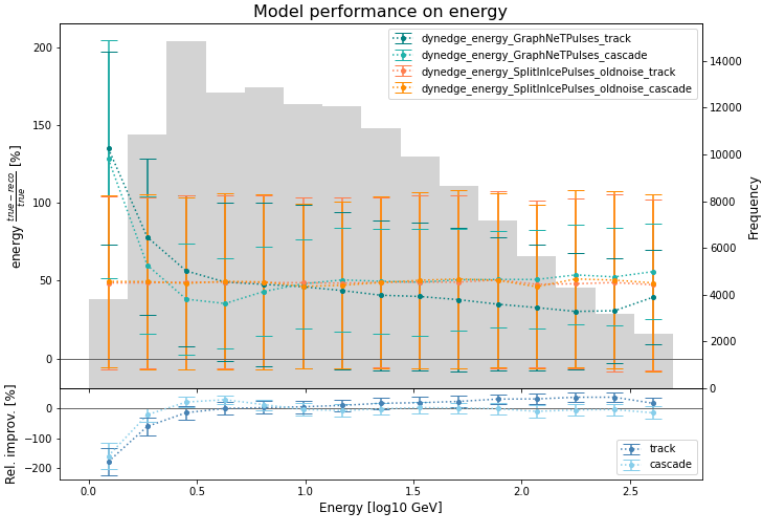


Figure 7.5: Mean of the error distribution of reconstructed energy by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets separated by energy and split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. The bottom of the figure shows relative improvement from SplitInIcePulses_oldnoise to GraphNeTPulses.

Figure 7.5 shows the residual means, and in this case, the peak is not significant enough the produce an advantage in performance for the old noise model predictions.[3] Instead, the old noise model performs better in the low energy range, where – as can be observed by inspection Figure 7.1 more closely – the cleaned pulse model has a

[3] Again, a simpler figure without track/cascade separation is shown in Figure A.8 in the Appendix.

tendency to predict to large values of energy. This is interesting, and may be explained by the fact that the pulse cleaning algorithm can have a more difficult time classifying low energy events which have fewer pulses, meaning that the low energy pulses will be less pure and lose more physics than the rest. This precipitates further investigations of the pulse cleaning with focus on the low energy regime, which is addressed in Section 7.3.

As these two models both perform reasonably well, the width of the residuals are plotted in Figure 7.12. The figures shows the same pattern as Figure 7.5, with the cleaned new noise model performing better in all energy ranges except the very low energy region.

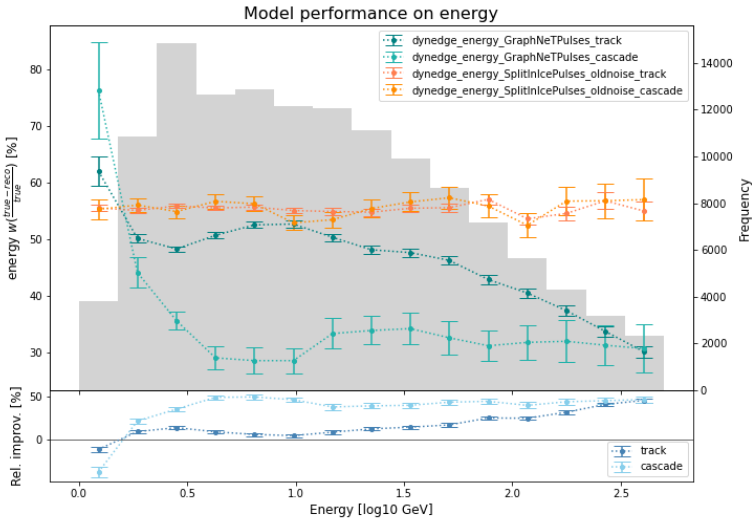


Figure 7.6: Width W of the error distribution of reconstructed energy by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets separated by energy and split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from SplitInIcePulses_oldnoise to GraphNeTPulses.

7.2.2 Azimuth

Figure 7.1 shows the reconstructed azimuth angle for GraphNeTPulses (top), SplitInIcePulses (center) and SplitInIcePulses_oldnoise (bottom) as a 2D histogram on the left and a scatter plot on the right with reconstructed energy on the x-axis and true energy on the y-axis. From the histograms, it is observed that each of the models are predicting the azimuth angle within an reasonable range of the truth for the majority of the events. The model trained on GraphNeTPulses seems to have more clearly defined white space between the green diagonal line and the corners, a observation that is reinforced when looking at the scatter plots, but direct comparisons will be needed to know how much this performance gain actually amounts to, and are carried out the following plots.

The comparison between azimuth reconstruction of the GraphNeTPulses and SplitInIcePulses models are illustrated in Figure 7.8 which shows a histogram of the reconstructed values and Figure 7.9 which

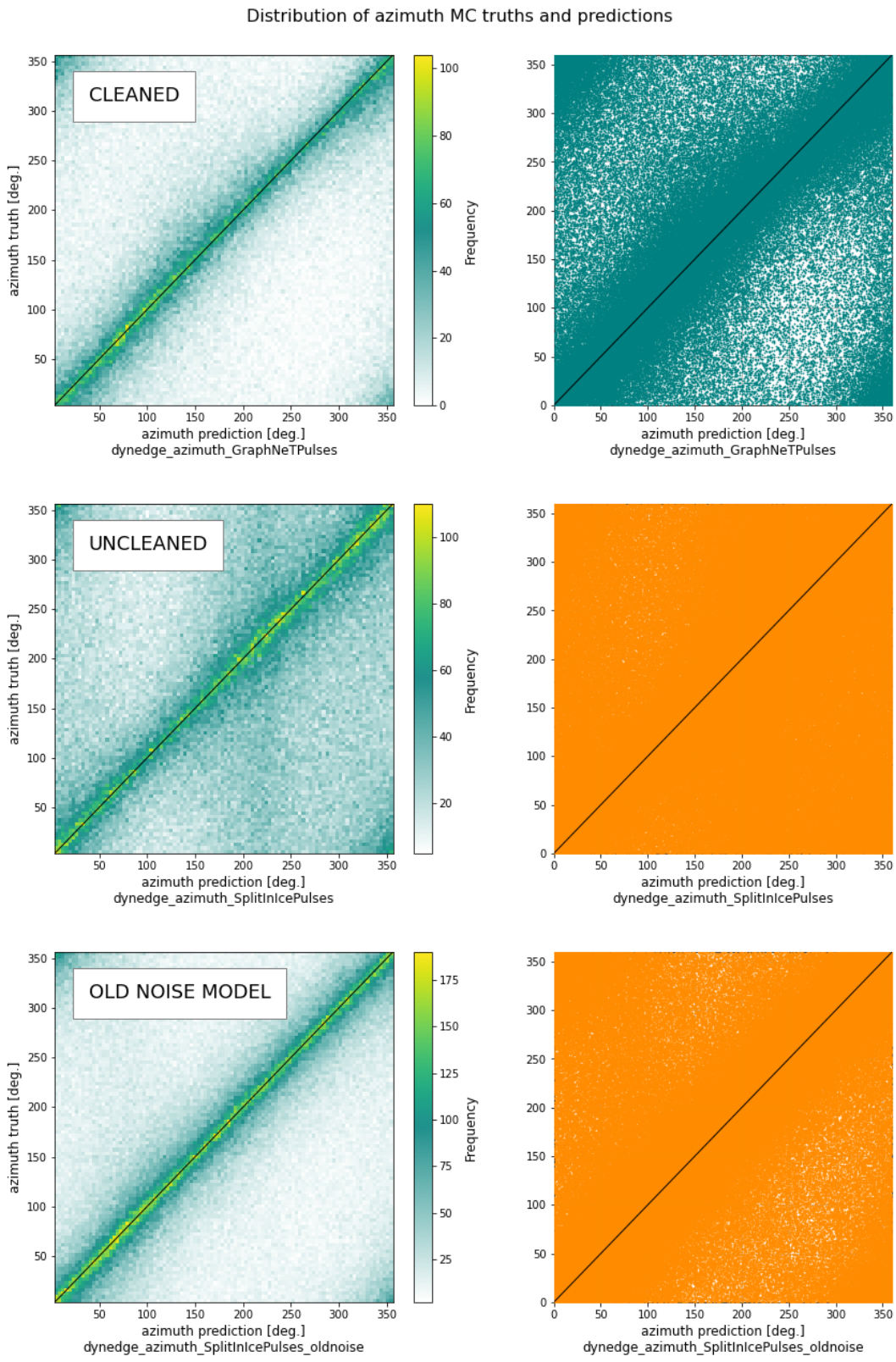


Figure 7.7: Distributions of the reconstructed azimuth angle by the models trained on the GraphNeTPulses (top), SplitInIcePulses (center) and SplitInIcePulses_oldnoise (bottom) datasets as 2D histograms (left) and scatter plots (right).

shows the means of residuals separated by energy bins[4]. In Figure 7.8, the two distributions actually looks quite similar in performance, with a characteristic and opposite sine wave shape, while the true values follow an approximately uniform distribution, showing that the GraphNeTPulses model predicts too many events in the range around 90 deg and too few in the 270 deg range, with the SplitInIcePulses doing the opposite. On the right side, near 300 deg, the uncleaned model actually looks to outperform the cleaned model.

[4] Again, a simpler figure without track/cascade separation is shown in Figure A.9 in the Appendix.

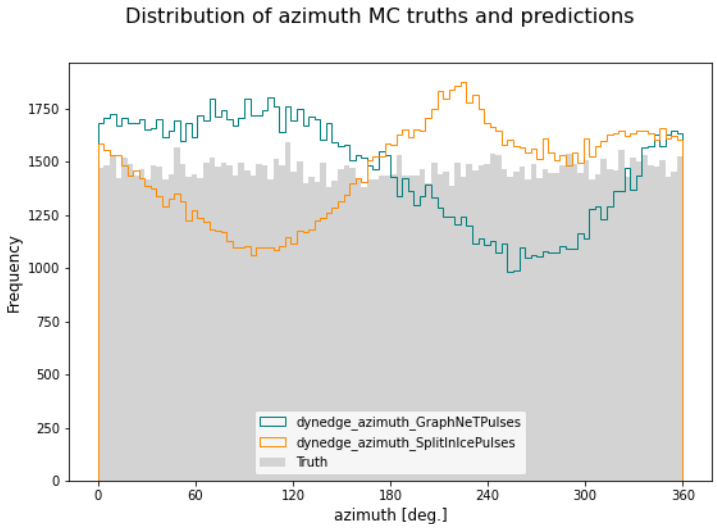


Figure 7.8: Distributions of the reconstructed azimuth angle by models trained on the GraphNeTPulses and SplitInIcePulses datasets with the true distribution in gray in the background.

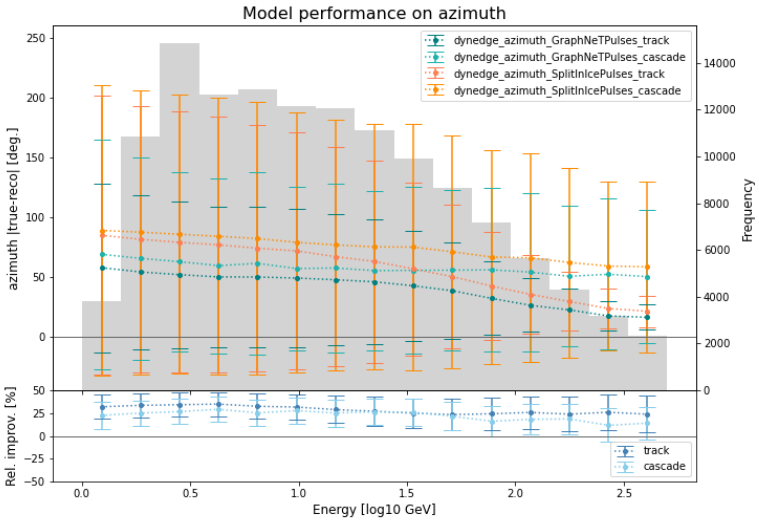


Figure 7.9: Mean of the error distribution of reconstructed azimuth angle by models trained on the GraphNeTPulses and SplitInIcePulses datasets separated by energy and split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from SplitInIcePulses to GraphNeTPulses.

But similar to the previous sections, the histogram can be misleading, and indeed Figure 7.9 shows an improvement in performance in all energy bins for both track- and cascade-like events. An interesting observation is that the reconstruction of cleaned cascade-like events

– while outperforming uncleaned cascade events – is never improved beyond the performance of uncleaned track-like events. As mentioned earlier, the models are expected to reconstruct track-like events better than cascade-like events, especially regarding the angles, but this discrepancy may indicate more serious shortcomings of the pulse cleaning model.

In Figures 7.10 to 7.12, the performances of the GraphNeTPulses and SplitInIcePulses_oldnoise models are compared. The histogram of reconstructed angles in Figure 7.10 shows that the models now approximately agree on which range to predict for too many events and which range to predict for too few, with the uncleaned old noise model seemingly lying closer to the correct range.

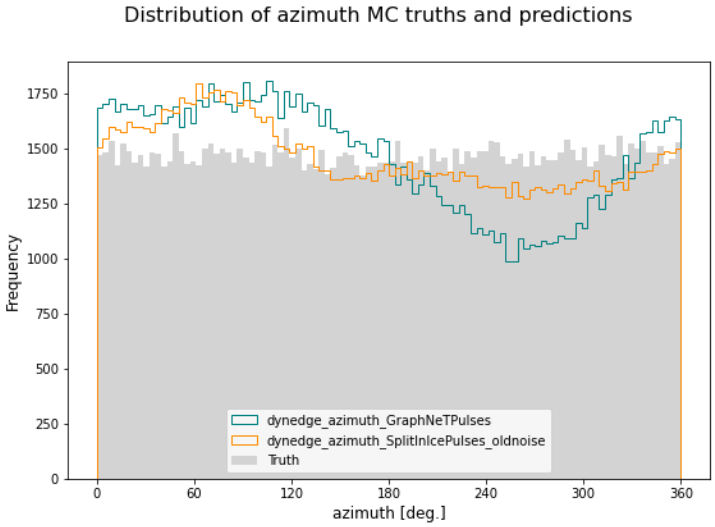


Figure 7.10: Distributions of the reconstructed azimuth angle by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets with the true distribution in gray in the background.

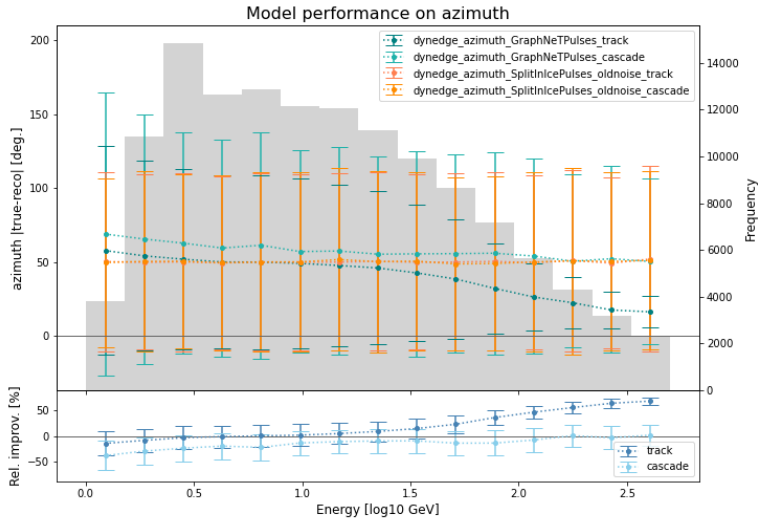


Figure 7.11: Mean of the error distribution of reconstructed azimuth angle by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets separated by energy and split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from SplitInIcePulses_oldnoise to GraphNeTPulses.

When looking at the means of the residuals in Figure 7.11, the performance of the old noise model is constant over the energy bins while the performance of the cleaned model improves with higher energy[5]. Just like for energy reconstruction, this may be an indication that lower energy makes is harder to distinguish physics pulses from noise, resulting in noisier datasets which in turn makes reconstruction harder. Another interesting observation is that the cleaned cascade reconstruction is actually worse than the old noise cascade reconstruction at all but a few energy ranges, a find that is confirmed when looking at the width of the residuals in Figure 7.12. A possible explanation for this behavior, is that it can be harder for the pulse cleaning model to separate noise from physics for the diffuse cascade-like events than for the more clearly asymmetric track-like events.

[5] Again, a simpler figure without track/cascade separation is shown in Figure A.10 in the Appendix.

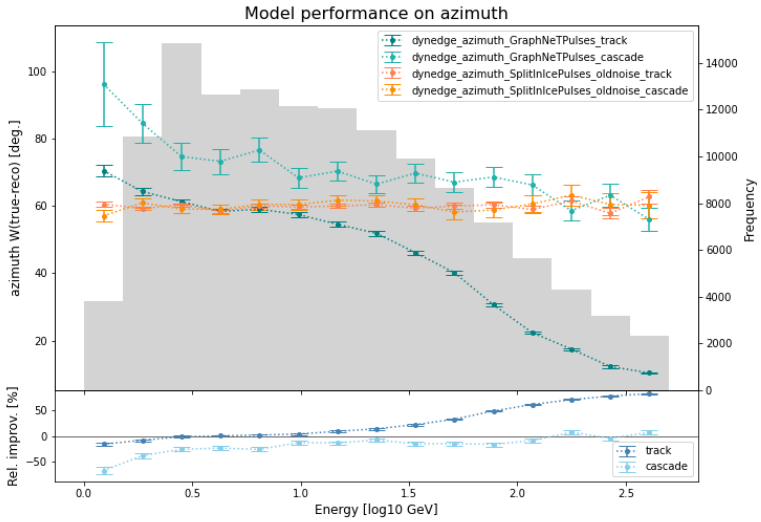


Figure 7.12: Width W of the error distribution of reconstructed azimuth angle by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets separated by energy and split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from SplitInIcePulses_oldnoise to GraphNeTPulses.

7.2.3 Zenith

The reconstruction of the final variable, the zenith angle, is described in this section. At this point two questions are of particular interest: Will the cleaned pulses continue to improve reconstruction performance in the high-energy range while lacking in the low-energy range? And will the models continue to struggle with regression the cascade-like events using the cleaned dataset?

Figure 7.13 shows the reconstructed zenith angle for GraphNeTPulses (top), SplitInIcePulses (center) and SplitInIcePulses_oldnoise (bottom) as a 2D histogram on the left and a scatter plot on the right with reconstructed energy on the x-axis and true energy on the y-axis. For the zenith angle, the cleaned dataset appears to produce a clear improvement in reconstruction. The models trained on both the SplitInIcePulses and the SplitInIcePulses_oldnoise datasets assign the

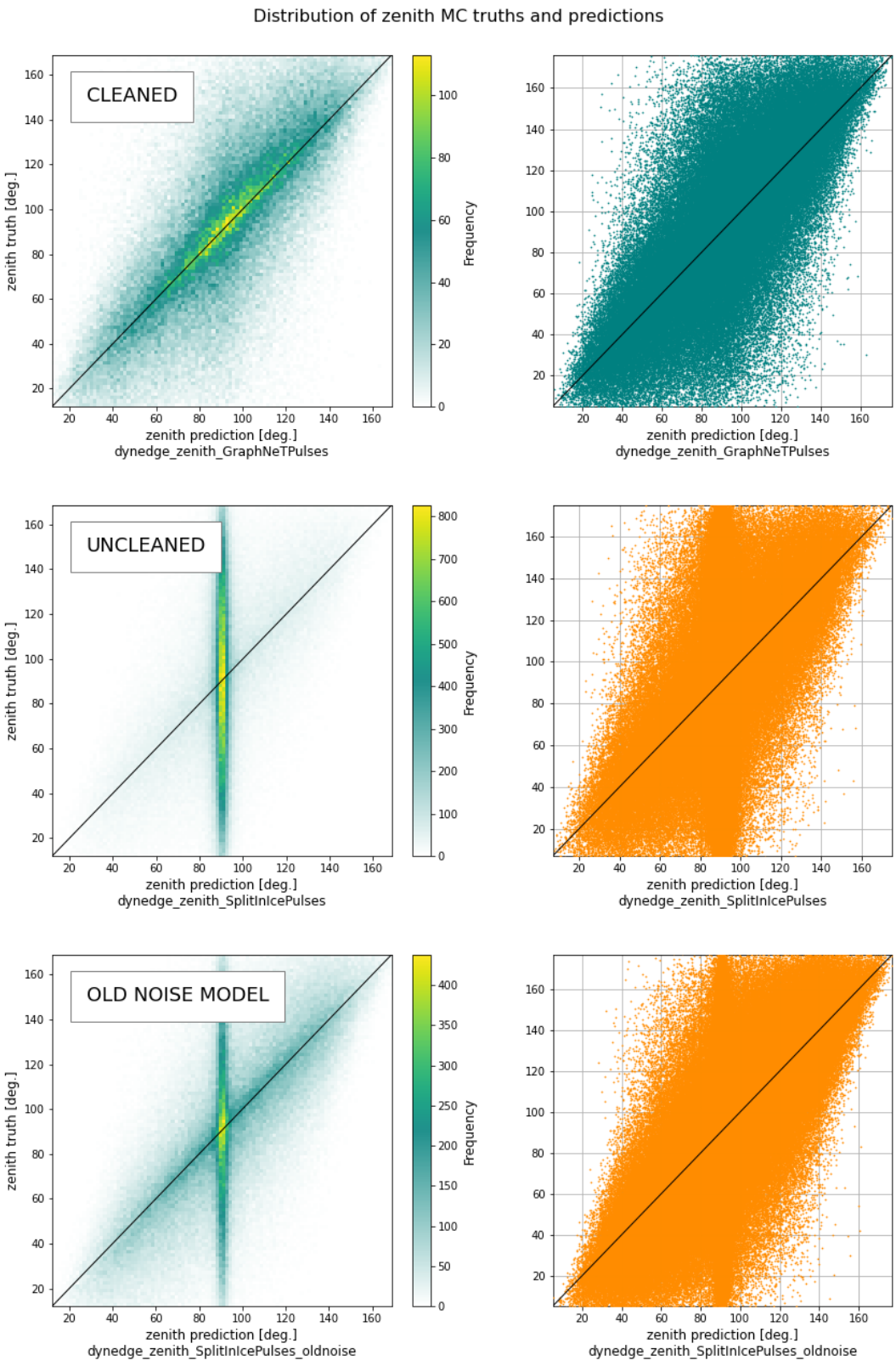


Figure 7.13: Distributions of the reconstructed zenuith angle by the models trained on the GraphNeTPulses (top), SplitInIcePulses (center) and SplitInIcePulses_oldnoise (bottom) datasets as 2D histograms (left) and scatter plots (right).

same zenith angle the majority of events. As the zenith angle is not uniformly distributed like the azimuth[6], this can again be interpreted as the models being rewarded for predicting the mean of the distribution.

[6] The reason for this is purely geometric: If one views IceCube as a point detector and imagines a sphere centered on the detector and divides the sphere into infinitesimal horizontal bands defined by infinitesimal zenith angles, one can easily infer that angles closer to horizontal cover a larger portion of the sphere than angles closer to vertical, due to the greater circumference of the associated bands.

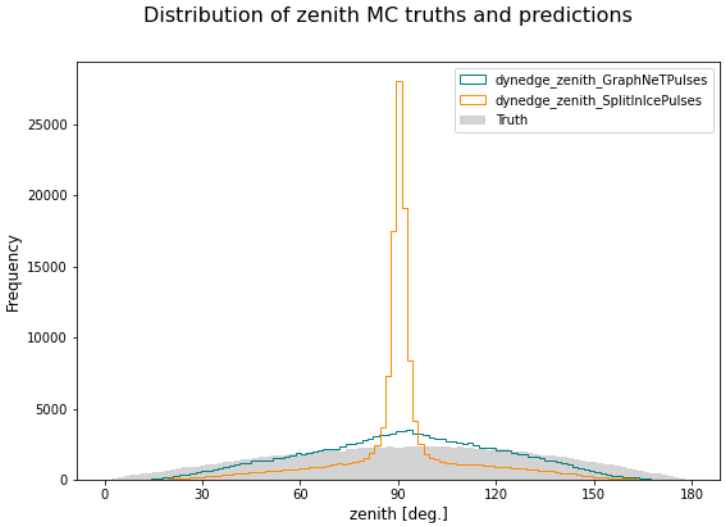


Figure 7.14: Distributions of the reconstructed zenith angle by models trained on the GraphNeTPulses and SplitInIcePulses datasets with the true distribution in gray in the background.

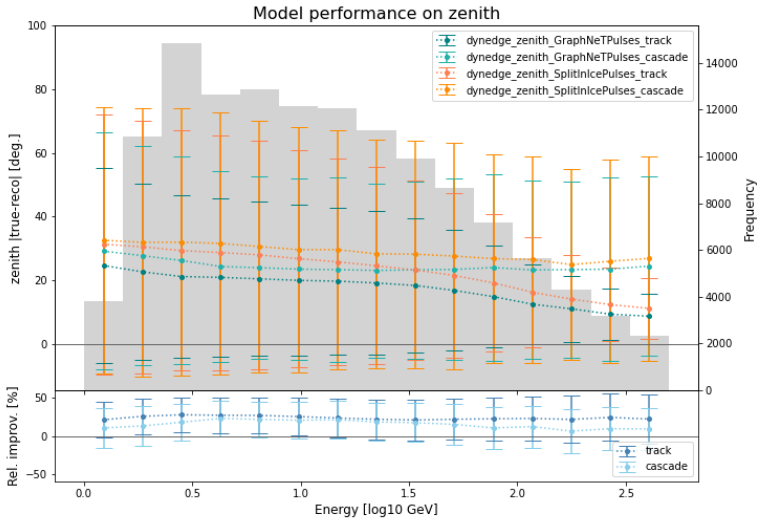


Figure 7.15: Mean of the error distribution of reconstructed zenith angle by models trained on the GraphNeTPulses and SplitInIcePulses datasets separated by energy and split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from SplitInIcePulses to GraphNeTPulses.

The comparison between the GraphNeTPulses and SplitInIcePulses models shown in Figures 7.14 and 7.15 exhibits the same patterns as observed for the other variables. In all energy ranges in Figure 7.15, the cleaned model outperforms the uncleaned model for both track-like and cascade-like events, even though the cascade reconstruction does not improve with higher energy the same way the track reconstruction does[7].

When comparing the cleaned GraphNeTPulses to the uncleaned

[7] Again, a simpler figure without track/cascade separation is shown in Figure A.11 in the Appendix.

SplitInIcePulses_oldnoise reconstruction, the previously observed patterns repeat as well. The cascade reconstruction residual mean of the cleaned model in Figure 7.17 flattens out after approximately 0.6 log10 GeV and never reaches the performance of the uncleaned old noise model[8]. And for the reconstructions at low energy the uncleaned old noise model is also better than the cleaned new noise model.

[8] Again, a simpler figure without track/cascade separation is shown in Figure A.12 in the Appendix.

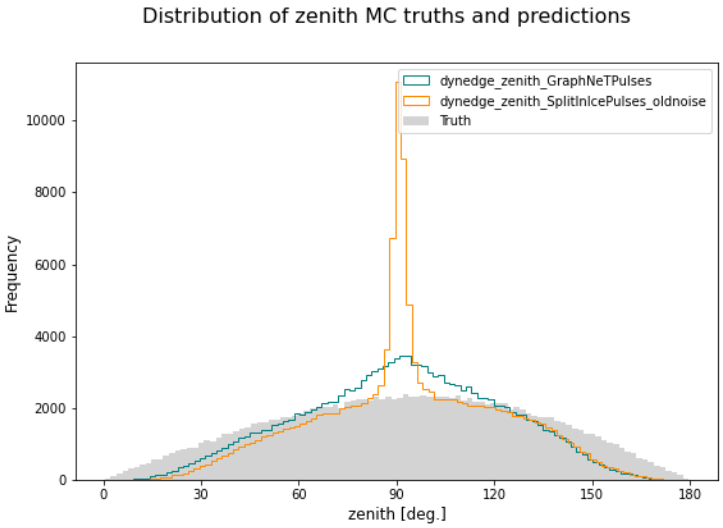


Figure 7.16: Distributions of the reconstructed zenith angle by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets with the true distribution in gray in the background.

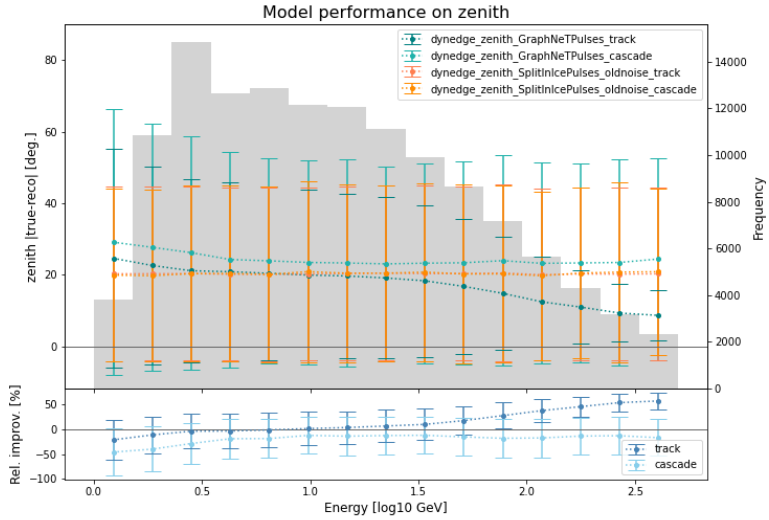


Figure 7.17: Mean of the error distribution of reconstructed zenith angle by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets separated by energy and split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. The bottom of the figure shows relative improvement from SplitInIcePulses_oldnoise to GraphNeTPulses.

When looking at the means of the residual distributions in Figure 7.18, the same patterns show even more clearly, indicated by a steep decrease in width from the cleaned model within the fist few bins, which continues for the track-like event regression. From this it is apparent that the performance of the cleaned new noise model matches and often surpasses the performance of the old noise model,

there are some areas where performance can be improved, mainly the low energy range and the cascade-like events. An immediate question that arises is whether this drop in reconstruction performance is also present in the pulse classification model in the same critical areas.

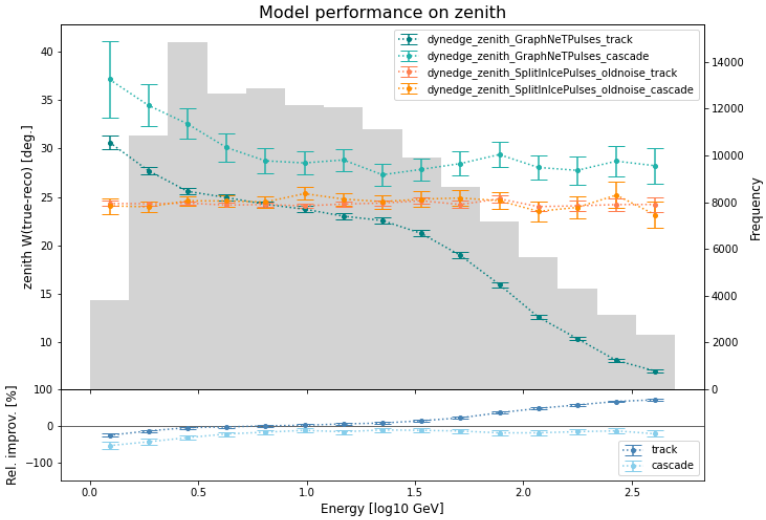


Figure 7.18: Width W of the error distribution of reconstructed zenith angle by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets separated by energy and split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. The bottom of the figure shows relative improvement from SplitInIcePulses_oldnoise to GraphNeTPulses.

7.3 Revisiting Pulse Cleaning

This section addresses the apparent shortcomings of the event reconstruction using the cleaned pulses pertaining to the regime of low energy. Performance metrics and pulse distributions are shown with a focus on variations with energy, and the low energy regime in particular.

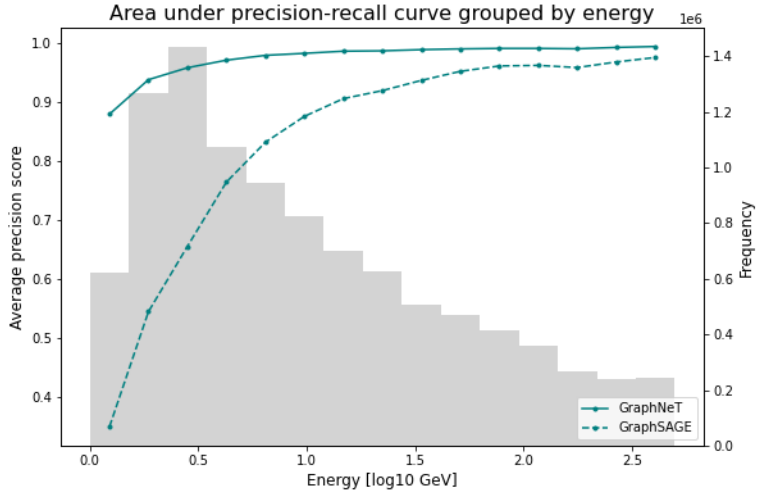


Figure 7.19: Area under precision-recall curve for events separated by energy for the GraphNeT and GraphSAGE models. The distribution of events are shown in the background as gray bars.

Figure 7.19 shows the performance of the GraphNeT and GraphSAGE models quantified by the area under the precision-recall curve, separated into bins according to the energy of the events associated with each pulse. As expected, the models exhibits significantly worse performance for both model in the low energy regime. A similar figure for the ROC AUC is shown in the Appendix, exhibiting similar behavior.

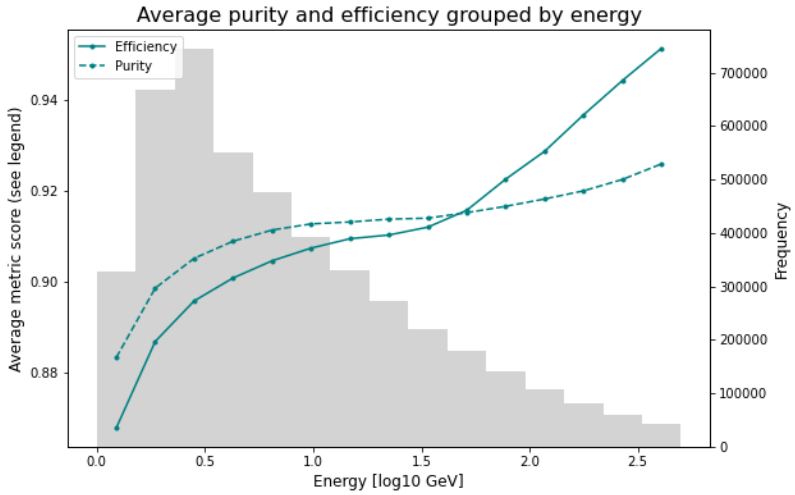


Figure 7.20: Average purity (precision) and efficiency (recall) when applying the model score threshold of 0.439 for the GraphNeT model on events separated by energy. The distribution of events are shown in the background as gray bars.

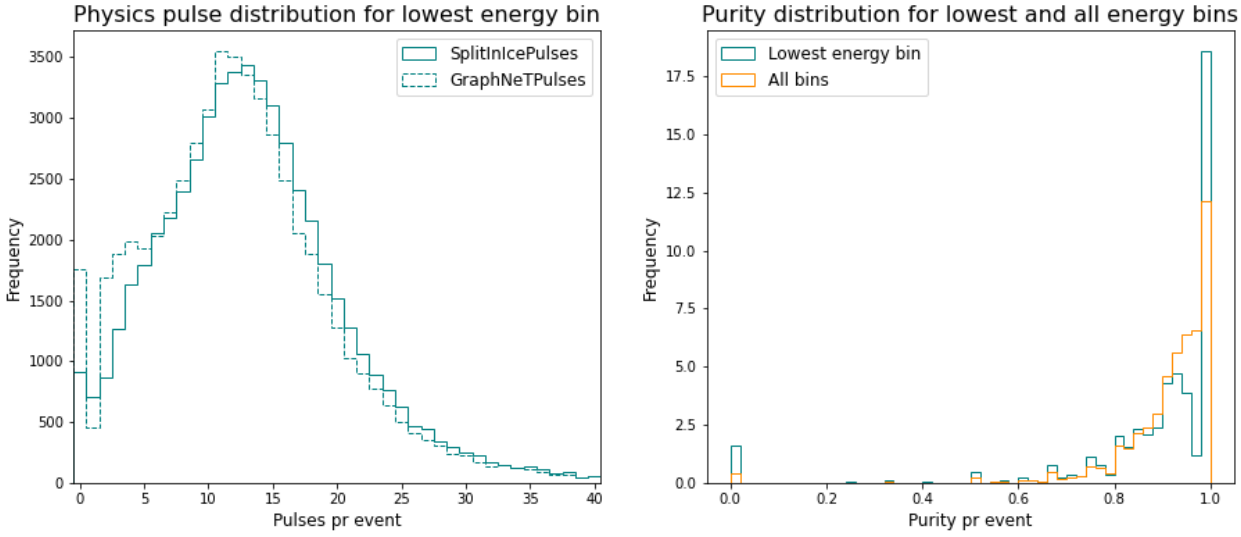


Figure 7.21: Area under precision-recall curve for events separated by energy for the GraphNeT and GraphSAGE models. The distribution of events are shown in the background as gray bars.

Figure 7.20 shows the average precision and efficiency of the pulses in each energy bin, when cleaning the pulses using the GraphNeT model with a threshold of 0.439 (the same cut used for the previous results). It shows the same behavior as in Figure 7.19, but in a more

readily interpretable manner. Using this cut, the events in the lowest energy bin retain on average $\sim$ 87% of the physics pulses and obtain a final purity of a little over 88%, while most events of higher energies retain above 90% of physics pulses and are 90% pure or more. While these differences are relatively small, they may have a significant impact on reconstruction.

Figure 7.21 shows the impact of the inferior cleaning on the events in the lowest energy bin. The on the left of the figure is shown the distribution of physics pulses pr. event (similar to Figure 6.6) of the cleaned (GraphSagePulses) and uncleaned (SplitInIcePulses) dataset. Many of the events end up with zero physics pulses (pulses with a total of zero pulses in the final dataset are not included in this plot), and the number of events with fewer than seven pulses grow significantly as well. On the right of the Figure is shown the distribution of the purity of each event in the lowest energy bin, compared to the distribution of all bins (the counts are normalised for comparison). At first glace it looks like the lowest energy bin is performing well, with the vast majority of events having 100% purity. But this bin includes all events o noise and can have any number of physics pulses, and can therefore be ignored for this comparison (still, they are left in for clarity). If the final bin is discarded, the plot resembles two normal distributions, the low energy distribution centered around 0.9 and the total distribution centered closer two 1.0. This tells the same story as the other plots, and it is worse noting that the lowest energy bin ends up with a significantly higher number of events with purity close to zero (which likely means zero physics and any amount of noise).

It is worth noting that this does not necessarily signify bad performance for the pulse cleaning model, but rather suggestions on how to improve it. Many ideas spring to mind from the knowledge gained from these plots, which are unfortunately outside the scope of this thesis. Perhaps a secondary model could be trained to focus on on the low energy regime, and the most confident model would decide the fate of each pulse. Or maybe one could get away with relaxing the cut with a lower threshold value (like the most efficient threshold in Figure 6.5) to retain more physics. Or perhaps some of the events should be discarded – as they would if this was to be implemented early in the OscNext pipeline – either by straight cuts at a given number of pulses or by a more sophisticated model like a GNN. Additionally, with more time, one could make the same investigations regarding the inferior reconstruction performance for cascade-like events also observed in the previous sections. And lastly, since the Upgrade arrays will have different types of DOMs than the original arrays, with multiple PMTs on each DOM, it might be beneficial to come up with a graph representation of Upgrade that includes this hierarchy between PMTs.

8 Conclusions and Outlook

Contents

8.1 Interaction Time Reconstruction

For the reconstruction of interaction_time variable, which has a very irregular distribution, the GNN model has been found to perform better when the target is scaled using the QuantileTransformer function than when using the RobustScaler or using no scaling of the target. For the QuantileTransformer, the width of the error distribution is lower than that of the RetroReco reconstruction for all energy bins except the high energy range (in which we are not interested and have less training data). This makes it comparable in performance to the GNN reconstruction of other variables.

Not much is left to desire from the interaction_time reconstruction, except of course even better performance when IceCube Upgrade is fully available. It is however worth keeping these results in mind when regressing other variables with irregular distribution. Therefore, I am satisfied to have spent the time implementing non-linear target transformation in the GraphNeT framework for future studies.

8.2 Event Level Classification

Generally, it is concluded that the GNN event selection surpasses that of the current OscNext cleaning pipeline in terms of final purity and efficiency. The model does however appear to benefit from being implemented on simulated data from Level 3, where the purity and efficiency is 99.7% and 25.1%, respectively, compared to 99.5% and 20.6% for Level 2 + DeepCore, 99.5% and 20.0% for Level 2, and 94.9% and

12.8% for the OscNext cleaning alone. This pairs well with the observation that using Level 3 data (as expected) yields better agreement between simulations and real data.

The real data is also where the greatest improvement or extensions of this study can be made. With a month or even a year of real data, the final sample should contain enough neutrinos to perform more rigorous tests like feature summaries of measures and reconstructed variables to gain more information about the agreement between simulated and real data. And again, this would be great study to perform again once IceCube Upgrade is installed and a adequate pulse cleaning algorithm has been implemented.

8.3 Upgrade Pulse Level Cleaning

The GNN also shows very promising results for the task of pulse level cleaning. The GraphNeT model does significantly better than the GraphSAGE model: When choosing an efficiency of 95%, the resulting purities are 95.8% for the GraphNeT model and 60.6% for the GraphSAGE model. These is a significant amount of events that have all or most of both their physics and noise pulses removed. This is not expected to be a concern in the a true cleaning pipeline where events with too few pulses would be removed.

With more time, I would like to further investigate the results obtained after performing the reconstruction of the cleaned pulses concerning the pulse cleaning performance on low energy events. Additionally, investigating and benchmarking pulse cleaning separate for track-like and cascade-like events seems to have a lot of potential. Since neither the energy nor the interaction type is known as the pulse-cleaning stage, the implementation of this is non-trivial. Training a separate classifier on low energy events and allowing the most certain classifier to determine the outcome could be an interesting solution, but with more investigation one might find other more sophisticated methods for this task.

8.4 Upgrade Reconstruction

There is no doubt that the pulse cleaning has a positive effect on reconstruction performance. For energy, the cleaned pulses allow the reconstruct to match and often even outperform that of the old noise model – effectively nullifying the increased noise rates of the Upgrade PMTs – in all energy ranges except for the very lowest range, where the pulse cleaning is expected (and later found) to perform poorly. For azimuth and zenith, the model matches or outperforms the old noise model in the same energy ranges for track-like events, but displays

worse performance for track-like events. This precipitates the investigation of pulse cleaning performance mentioned above.

In addition to improvements of the pulse cleaning, choices in the *event* selection could also be of value for the reconstruction performance. With more time, I would like to try and imitate the SRT-cleaning process or other low level cleaning algorithm, to sort out noisy events and events unfit for reconstruction after pulse cleaning. This would emulate match how a the cleaned pulses would be treated in a true reconstruction pipeline. Additionally, I would like to compare these results to IceCube + DeepCore reconstructions, as the purpose of Upgrade is to allow for better reconstruction that the current detector array.

Finally, I think it would be worth-wile to come up with a new graph representation of IceCube including IceCube upgrade which takes into account the multiple PMTs pr DOM to create a hierachy between DOMs. This would require deeper forays into graph theory than have been made in this work and would likely be both fun and instructive.

A Supporting Figures

A.1 Particle Physics

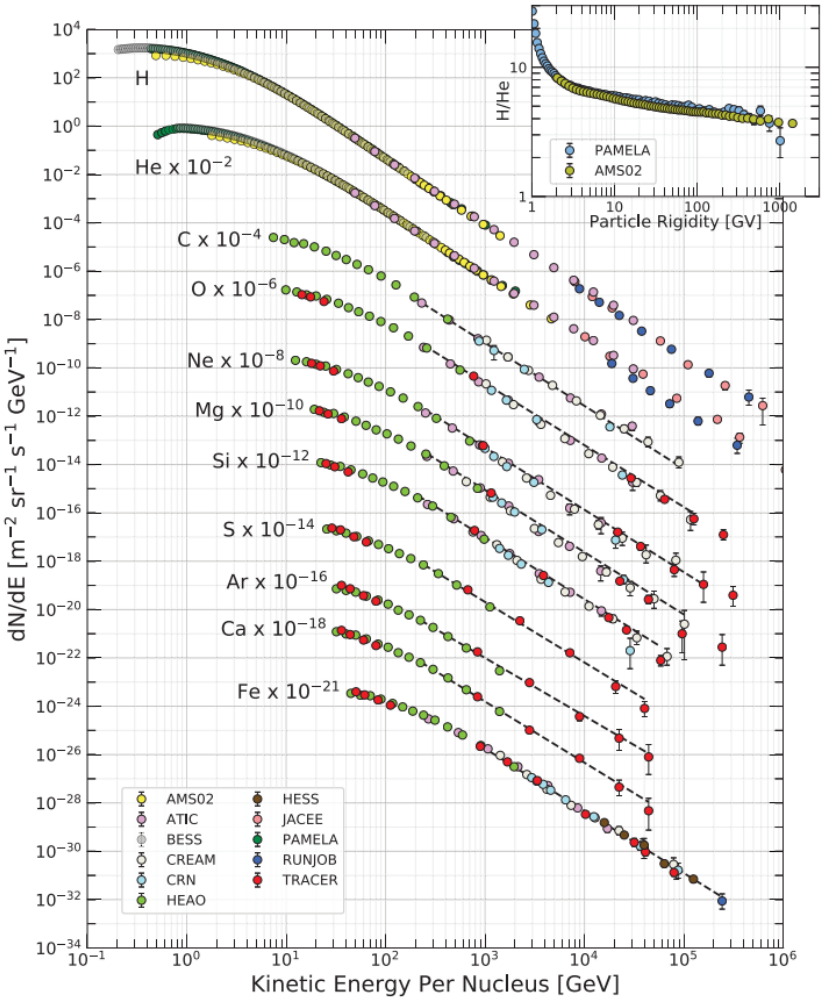


Figure A.1: The composition and energy distribution of cosmic ray primary nuclei. Top right shows hydrogen and helium cores a fraction of particle rigidity, which is related to particle momentum. *Image from* [30]

A.2 IceCube

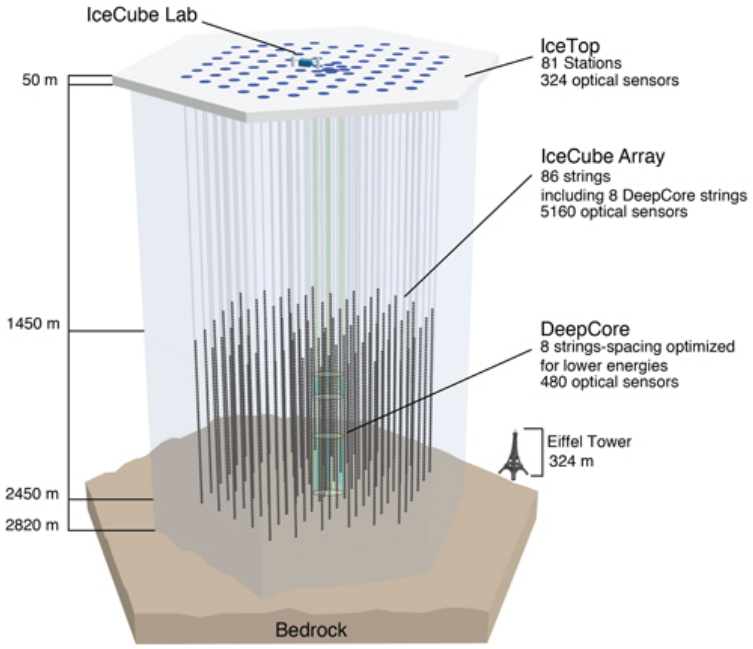


Figure A.2: Illustration of the IceCube detector array.
Image from [102]

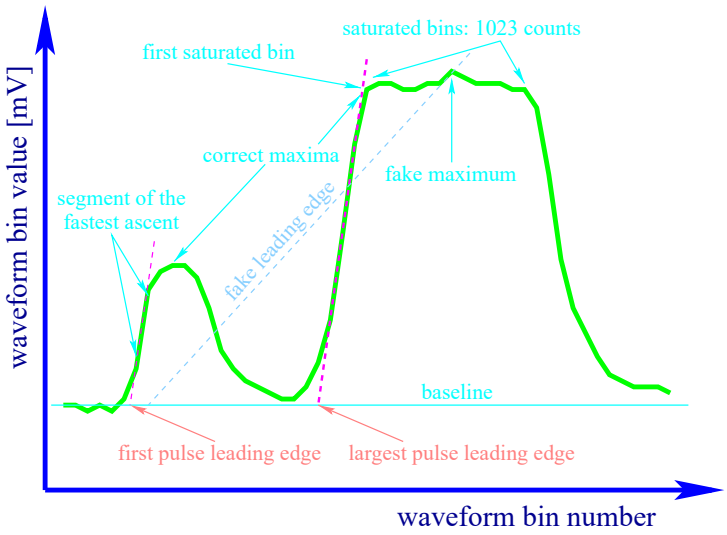


Figure A.3: Schematic of the calculation of DOM pulse features.
Image from the IceCube FeatureExtractor documentation [51]

A.3 Event Level Classification

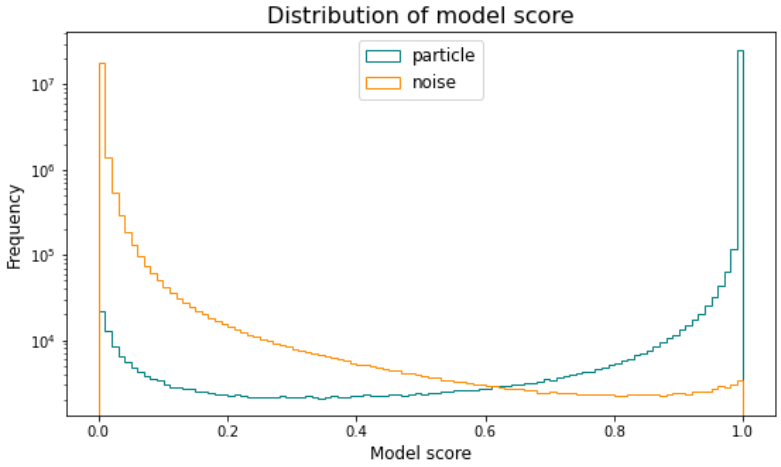


Figure A.4: Distribution of model scores for noise/particle event classification for the GNN with base 10 logarithmic y-axis.

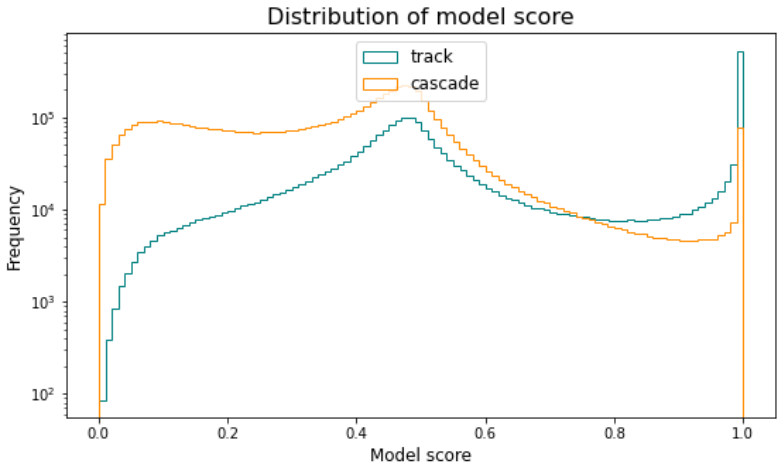


Figure A.5: Distribution of model scores for track/cascade event classification for the GNN with base 10 logarithmic y-axis.

A.4 Upgrade Pulse Level Noise Cleaning

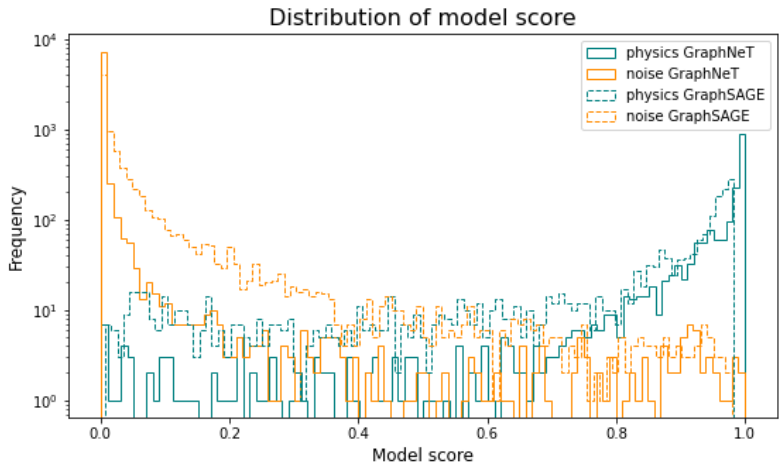


Figure A.6: Distribution of model scores for noise and physics pulses for the GraphNeT and GraphSAGE pulse cleaning models with base 10 logarithmic y-axis.

A.5 Upgrade Reconstruction

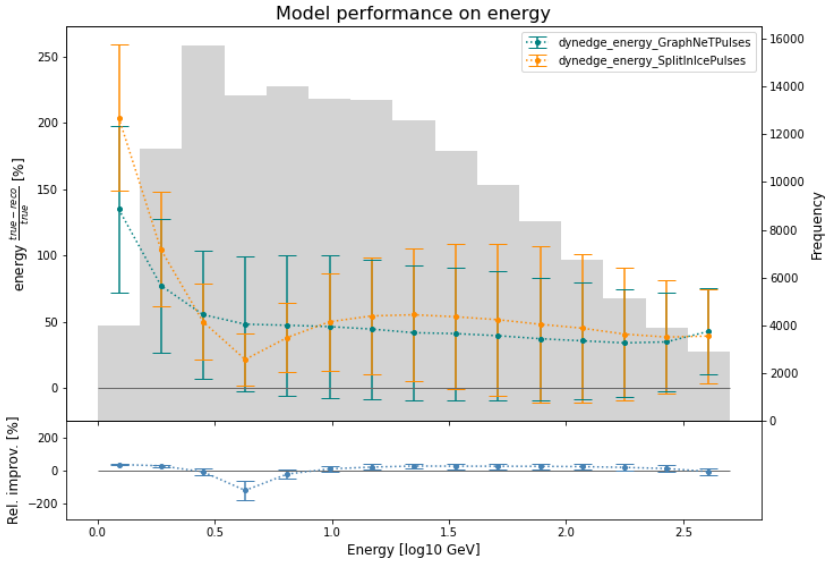


Figure A.7: Mean of the error distribution of reconstructed energy angle by models trained on the GraphNeTPulses and SplitInIcePulses datasets separated by energy with no split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from SplitInIcePulses to GraphNeTPulses.

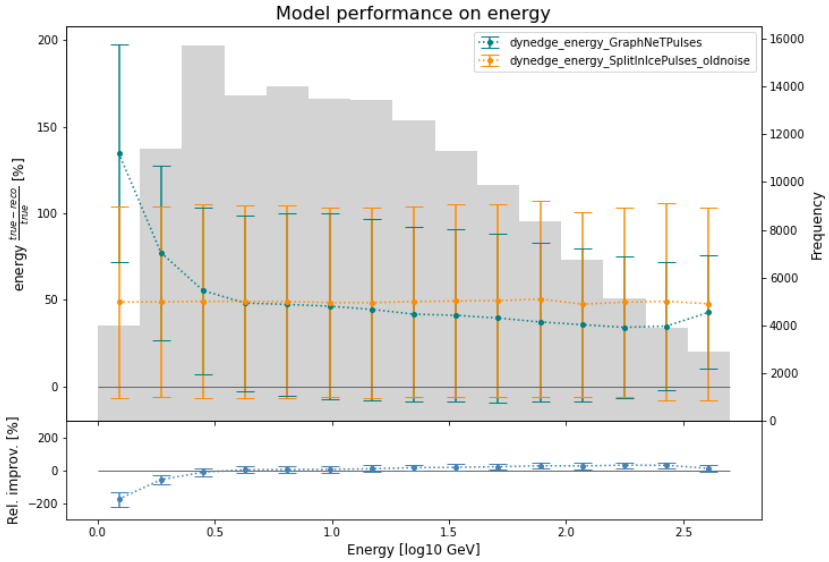


Figure A.8: Mean of the error distribution of reconstructed energy angle by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets separated by energy with no split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. The bottom of the figure shows relative improvement from SplitInIcePulses_oldnoise to GraphNeTPulses.

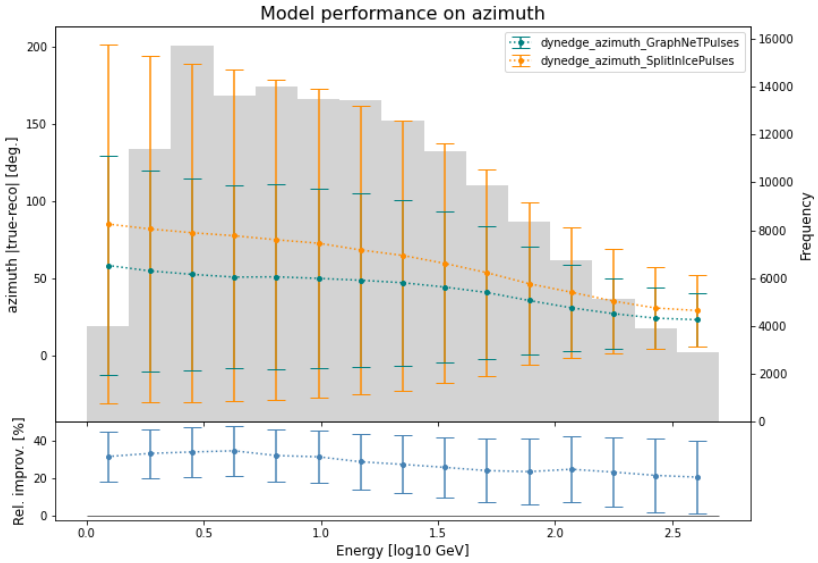


Figure A.9: Mean of the error distribution of reconstructed azimuth angle by models trained on the GraphNeTPulses and SplitInIcePulses datasets separated by energy with no split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from SplitInIcePulses to GraphNeTPulses.

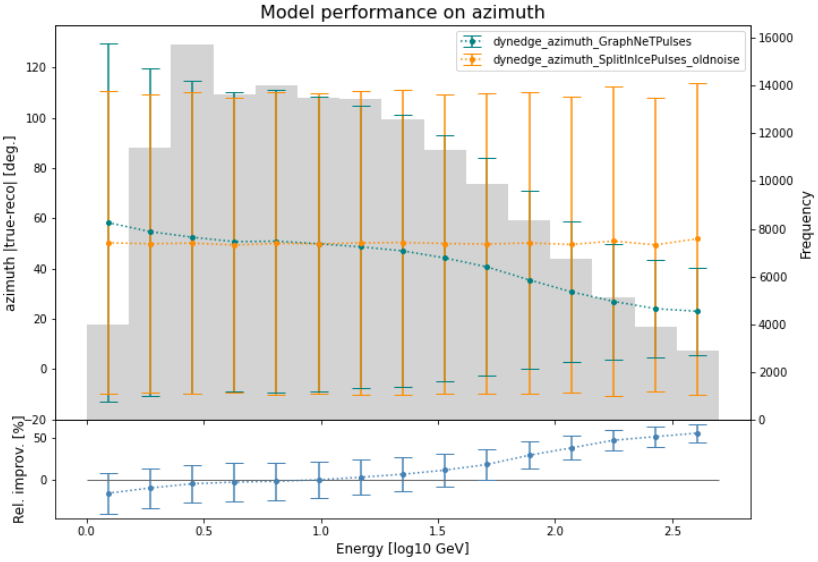


Figure A.10: Mean of the error distribution of reconstructed azimuth angle by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets separated by energy with no split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. The bottom of the figure shows relative improvement from SplitInIcePulses_oldnoise to GraphNeTPulses.

Figure A.11: Mean of the error distribution of reconstructed zenith angle by models trained on the GraphNeTPulses and SplitInIcePulses datasets separated by energy with no split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. Bottom of the figure shows relative improvement from SplitInIcePulses to GraphNeTPulses.

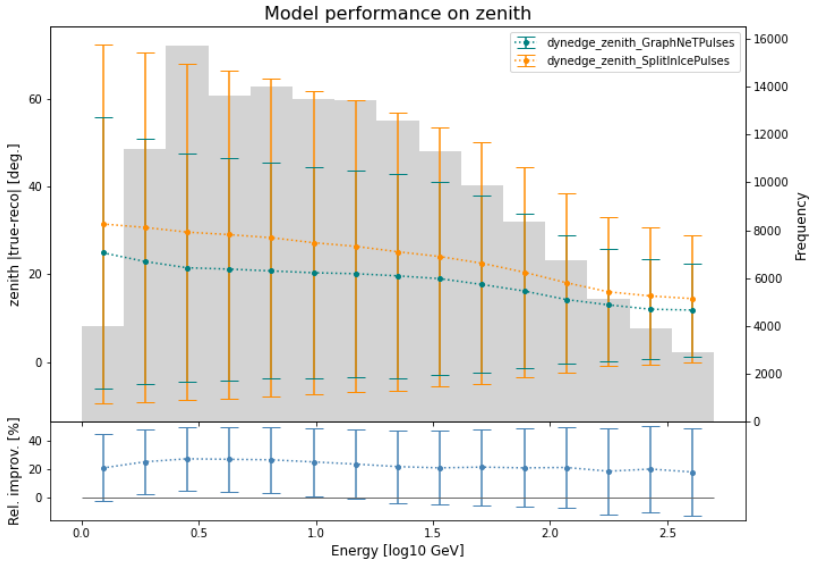
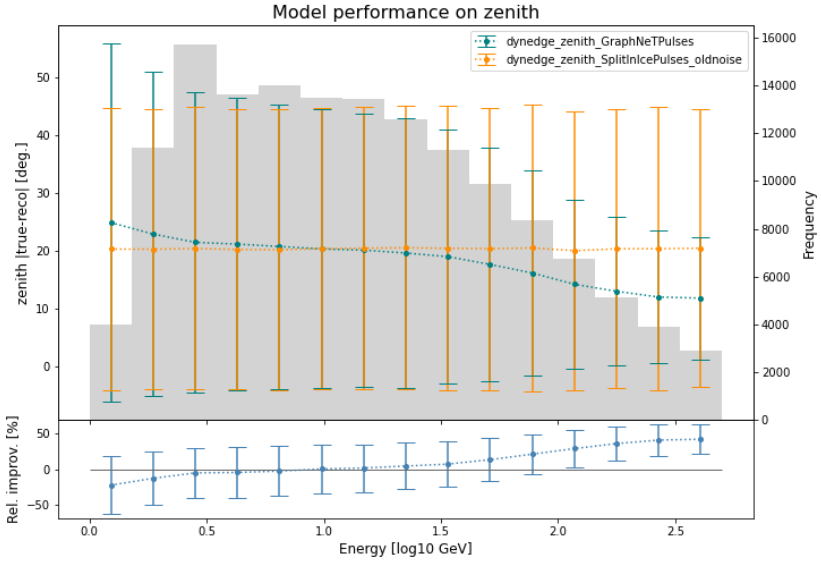


Figure A.12: Mean of the error distribution of reconstructed zenith angle by models trained on the GraphNeTPulses and SplitInIcePulses_oldnoise datasets separated by energy with no split by track and cascade-like events. Gray histogram in the background indicates the amount of events in each energy bin. The bottom of the figure shows relative improvement from SplitInIcePulses_oldnoise to GraphNeTPulses.



B Derivations

B.1 Uncertainty σ_W of the residual distribution W

This derivation is based on the introduction to order statistics in [97] and uses the theorems on normal distributions of order statistics from [103].

B.1.1 On order statistics

Let $X_1, ..., X_n$ be n **independent and identically distributed continuous random variables** with the density f and distribution function F. Then the **order statistic** $X_{(j)}$ is then the jth smallest value of $X_1, ..., X_n$, such that $X_{(1)}$ is the smallest value and $X_{(n)}$ is the largest.

The **joint density function** of the order statistics, meaning the density function of $X_{(1)}, ..., X_{(n)}$ equalling the values $x1 \leq ... \leq x_n$ is given by

$$f_{X_{(1)},...,X_{(n)}}(x_1, ..., x_2) = n! f(x_1)...f(x_n) \tag{B.1}$$

This is taken from Equation 6.1 in [97], and can easily be explained by noting that in order for the order statistics $X_{(1)}, ..., X_{(n)}$ to equal the values $(x_1, ..., x_2)$, is it sufficient for the values $X_1, ..., X_n$ to equal any **permutation** of $(x_1, ..., x_2)$. There are $n!$ of these and their probability (density) equals $f(x_1)...f(x_n)$, and this leads to Equation B.1.

In order for the jth order statistic $X_{(j)}$ to equal a number x, the follow conditions must apply

There are j-1 values smaller than x. Probability density:	$F(x)^{j-1}$
There are n-j values greater than x. Probability density:	$[1 - F(x)]^{n-j}$
There is 1 value equal to x. Probability density:	$f(x)$

Combining the above statements yields the probability density:

$$F(x)^{j-1}[1 - F(x)]^{n-j} f(x) \tag{B.2}$$

but since this holds for any set of $j - 1$ being smaller than any set of

$n - j$, among the $n!$ permutations, there are

$$\binom{n}{j-1, n-j, 1} = \frac{n!}{(n-j)!(j-1)!} \tag{B.3}$$

different partitions that fulfill the criteria, resulting in the density function

$$f_{X_{(j)}}(x) = \frac{n!}{(n-j)!(j-1)!} F(x)^{j-1}[1 - F(x)]^{n-j} f(x) \tag{B.4}$$

which approaches a normal distribution as $n \to \infty$.

B.1.2 Calculating σ_W

The definitions and calculations above are useful since W is calculated from percentiles, and the kth percentile is exactly the order statistic $X_{(np)}$, where $p = k/100$.

From the theorem of Pearson and Smirnoff in [103], the order statistic $X_{(np)}$ is distributed according to the normal distribution with means μ_p given by

$$\int_{-\infty}^{\mu_p} f(x)dx = p \tag{B.5}$$

and variances:

$$\sigma_p^2 = \frac{p(1-p)}{nf(\mu_p)^2} \tag{B.6}$$

Isolating μ_p in Equation B.5 yields

$$\mu_p = F^{-1}(p) \tag{B.7}$$

and using the law of error propagation

$$\sigma_f^2 = \sum_i \frac{\delta f^2}{\delta x_i} \sigma_{x_i}^2 \tag{B.8}$$

while ignoring correlations gives the estimated error on W:

$$\sigma_W = \frac{1}{1.349} \sqrt{\frac{0.25 \cdot (1 - 0.25)}{n} \left(\frac{1}{f(R_{0.25})^2} + \frac{1}{f(R_{0.75})^2} \right)} \tag{B.9}$$

List of Figures

List of Tables

Bibliography

[1] Wikipedia contributors. *Standard Model — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-March-2022]. 2022. URL: https://en.wikipedia.org/wiki/Standard_Model.

[2] Bogdan Povh et al. *Particles and Nuclei*. Jan. 2015. ISBN: 978-3-662-46320-8. DOI: 10.1007/978-3-662-46321-5.

[3] Wikipedia contributors. *File:Elementary particle interactions in the Standard Model.png — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-March-2022]. 2022. URL: https://en.m.wikipedia.org/wiki/File:Elementary_particle_interactions_in_the_Standard_Model.png.

[4] S. Bilenky. "Neutrino oscillations: From a historical perspective to the present status". In: *Nuclear Physics B* 908 (July 2016), pp. 2–13. ISSN: 0550-3213. DOI: 10.1016/j.nuclphysb.2016.01.025. URL: http://dx.doi.org/10.1016/j.nuclphysb.2016.01.025.

[5] Carsten Jensen. *Controversy and Consensus : Nuclear Beta Decay 1911-1934*. Basel Boston, Mass: Birkhäuser Verlag, 2000. ISBN: 978-3-0348-9569-9.

[6] Laurie M. Brown. "The Idea of the Neutrino". In: *Physics Today* 31.9 (1978), pp. 23–28.

[7] "The Reines-Cowan Experiments"." In: *Los Alamos Science* 25.25 (1997).

[8] 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.

[9] Ivan V. Anicin. "The Neutrino - Its Past, Present and Future". In: (2005). DOI: 10.48550/ARXIV.PHYSICS/0503172. URL: https://arxiv.org/abs/physics/0503172.

[10] John N. Bahcall and Raymond Davis. "Solar Neutrinos: A Scientific Puzzle". In: *Science* 191.4224 (1976), pp. 264–267. ISSN: 0036-8075. DOI: 10.1126/science.191.4224.264. eprint: https://science.sciencemag.org/content/191/4224/264.full.pdf. URL: https://science.sciencemag.org/content/191/4224/264.

[11] B. Pontecorvo. "Mesonium and anti-mesonium". In: *Sov. Phys. JETP* 6 (1957), p. 429.

[12] K. Abe et al. "Constraint on the matter–antimatter symmetry-violating phase in neutrino oscillations". In: *Nature* 580.7803 (Apr. 2020), pp. 339–344. DOI: 10.1038/s41586-020-2177-0. URL: https://doi.org/10.1038%5C%2Fs41586-020-2177-0.

[13] E. Kh. Akhmedov and A. Yu. Smirnov. "Paradoxes of neutrino oscillations". In: *Physics of Atomic Nuclei* 72.8 (Aug. 2009), pp. 1363–1381. DOI: 10.1134/s1063778809080122. URL: https://doi.org/10.1134/s1063778809080122.

[14] J W F Valle. "Neutrino physics overview". In: *Journal of Physics: Conference Series* 53 (Nov. 2006), pp. 473–505. DOI: 10.1088/1742-6596/53/1/031.

[15] 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.

[16] Carlo Giunti and Marco Laveder. *Neutrino Mixing*. 2003. DOI: 10.48550/ARXIV.HEP-PH/0310238. URL: https://arxiv.org/abs/hep-ph/0310238.

[17] Etienne Bourbeau. *Neutrino oscillation — Niels Bohr Institute*. [Online; accessed 30-March-2022]. 2021. URL: https://nbi.ku.dk/english/research/experimental-particle-physics/icecube/neutrino-oscillation/.

[18] J.-L. Tastet, O. Ruchayskiy, and I. Timiryasov. "Reinterpreting the ATLAS bounds on heavy neutral leptons in a realistic neutrino oscillation model". In: *Journal of High Energy Physics* 2021.12 (Dec. 2021). DOI: 10.1007/jhep12(2021)182. URL: https://doi.org/10.1007%5C%2Fjhep12%282021%29182.

[19] Anna Katharina Steuer. "Searching for Heavy Neutral Leptons at CERN". PhD thesis. University of Copenhagen, Feb. 2021.

[20] Shatendra Sharma. *Atomic And Nuclear Physics*. S.l: Pearson Education India, 2008. ISBN: 978-81-317-1924-4.

[21] A. Aab et al. "Observation of a large-scale anisotropy in the arrival directions of cosmic rays above 8 × 10¹⁸ eV". In: *Science* 357.6357 (2017), pp. 1266–1270. DOI: 10.1126/science.aan4338. eprint: https://www.science.org/doi/pdf/10.1126/science.aan4338. URL: https://www.science.org/doi/abs/10.1126/science.aan4338.

[22] P A Zyla et al. "Review of Particle Physics". In: *Progress of Theoretical and Experimental Physics* 2020.8 (Aug. 2020). DOI: 10.1093/ptep/ptaa104. URL: https://doi.org/10.1093/ptep/ptaa104.

[23] Takaaki Kajita. "Atmospheric Neutrinos". In: *Advances in High Energy Physics* 2012 (2012), pp. 1–24. DOI: 10.1155/2012/504715. URL: https://doi.org/10.1155/2012/504715.

[24] R. Fedynitch A.and Engel. "MCEq - efficient computation of particle particle cascades in the atmosphere". In: *Unpublished* (2018).

[25] Etienne Bourbeau. "Measurement of Tau Neutrino Appearance in 8 Years of IceCube Data". PhD thesis. University of Copenhagen, 2021.

[26] Mark Thomson. *Modern Particle Physics*. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139525367.

[27] P. Berghaus. "Muons in IceCube". In: *Nuclear Physics B - Proceedings Supplements* 196 (Dec. 2009), pp. 261–266. DOI: 10.1016/j.nuclphysbps.2009.09.050. URL: https://doi.org/10.1016%5C%2Fj.nuclphysbps.2009.09.050.

[28] J. Beringer et al. *PDGLive Particle Summary 'Leptons (e, mu, tau, ... neutrinos ...)* 2012. URL: https://pdg.lbl.gov/2012/tables/rpp2012-sum-leptons.pdf.

[29] Carlotta Giusti, Andrea Meucci, and Franco Davide Pacati. "Charged-Current and Neutral-Current Neutrino-Nucleus Scattering in a Relativistic Approach". In: (2009). DOI: 10.48550/ARXIV.0906.2873. URL: https://arxiv.org/abs/0906.2873.

[30] M. Tanabashi et al. "Review of Particle Physics: Particle Data Group". English. In: *Physical Review D* 98.3 (Aug. 2018). ISSN: 2470-0010. DOI: 10.1103/PhysRevD.98.030001.

[31] John David Jackson. *Classical electrodynamics*. 3rd ed. New York, NY: Wiley, 1999. ISBN: 9780471309321. URL: http://cdsweb.cern.ch/record/490457.

[32] Hadiseh Alaeian. *An Introduction to Cherenkov Radiation*. [Online; accessed 30-March-2022]. Mar. 2015. URL: http://large.stanford.edu/courses/2014/ph241/alaeian2/.

[33] Adam Bernstein et al. *Report on the Depth Requirements for a Massive Detector at Homestake*. 2009. DOI: 10.48550/ARXIV.0907.4183. URL: https://arxiv.org/abs/0907.4183.

[34] P Sengupta. *Classical electrodynamics*. New Delhi: New Age International, 2000. ISBN: 9788122412499.

[35] 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.

[36] "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.

[37] R. Abbasi et al. "The IceCube data acquisition system: Signal capture, digitization, and timestamping". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 601.3 (Apr. 2009), pp. 294–316. ISSN: 0168-9002. DOI: 10.1016/j.nima.2009.01.001. URL: http://dx.doi.org/10.1016/j.nima.2009.01.001.

[38] IceCube Collaboration. *IceCube Overview*. [Online; accessed 21-March-2022]. 2022. URL: https://icecube.wisc.edu/about-us/overview/.

[39] M. G. Aartsen et al. "Measurement of Atmospheric Neutrino Oscillations with IceCube". In: *Phys. Rev. Lett.* 111 (8 Aug. 2013), p. 081801. DOI: 10.1103/PhysRevLett.111.081801. URL: https://link.aps.org/doi/10.1103/PhysRevLett.111.081801.

[40] M. G Aartsen et al. "Measurement of atmospheric tau neutrino appearance with IceCube DeepCore". In: *Physical Review D* 99 (Feb. 2019). DOI: 10.1103/PhysRevD.99.032007.

[41] M.G. Aartsen et al. "The IceCube Neutrino Observatory: instrumentation and online systems". In: *Journal of Instrumentation* 12.03 (Mar. 2017), P03012–P03012. DOI: 10.1088/1748-0221/12/03/p03012. URL: https://doi.org/10.1088/1748-0221/12/03/p03012.

[42] Wikipedia contributors. *File:ICECUBE dom taklampa.jpg — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-March-2022]. 2022. URL: https://commons.wikimedia.org/wiki/File:ICECUBE_dom_taklampa.jpg.

[43] DESY - Deutsches Elektronen Synchrotron. *SENSE - Experiments Portraits - IceCube*. [Online; accessed 21-March-2022]. 2022. URL: https://www.sense-pro.org/portraits/experiments/icecube.

[44] R. Abbasi et al. "The design and performance of IceCube DeepCore". In: *Astroparticle Physics* 35.10 (2012), pp. 615–624. ISSN: 0927-6505. DOI: https://doi.org/10.1016/j.astropartphys.2012.01.004. URL: https://www.sciencedirect.com/science/article/pii/S0927650512000254.

[45] Aya Ishihara. *The IceCube Upgrade – Design and Science Goals*. 2019. DOI: 10.48550/ARXIV.1908.09441. URL: https://arxiv.org/abs/1908.09441.

[46] D. Heck et al. *CORSIKA: a Monte Carlo code to simulate extensive air showers.* 1998.

[47] Anna Katharina Steuer. "Cascade type identification in IceCube and an application in a search for new physics". PhD thesis. Mainz U., 2018. DOI: 10.25358/openscience-4463.

[48] Costas Andreopoulos et al. *The GENIE Neutrino Monte Carlo Generator: Physics and User Manual*. 2015. DOI: 10.48550/ARXIV.1510.05494. URL: https://arxiv.org/abs/1510.05494.

[49] Richard D Hipp. *SQLite*. Version 3.31.1. 2020. URL: https://www.sqlite.org/index.html.

[50] Kevin Meagher. "I3File Tools and Visulaization". In: *IceCube Bootcamp* (June 2020).

[51] Dmitry Chirkin. *Feature Extraction of IceCube Waveforms*. URL: http://code.icecube.wisc.edu/svn/projects/FeatureExtractor/.

[52] F. Träger. *Springer Handbook of Lasers and Optics*. Springer Handbooks. Springer Berlin Heidelberg, 2012. ISBN: 9783642194092. URL: https://books.google.dk/books?id=Ad5G1HWtDRgC.

[53] L. Garren et al. "Monte carlo particle numbering scheme". In: *The European Physical Journal C* 15.1-4 (Mar. 2000), pp. 205–207. DOI: 10.1007/bf02683426. URL: https://doi.org/10.1007/bf02683426.

[54] R. Abbasi et al. "Low Energy Event Reconstruction in IceCube DeepCore". In: (Mar. 2022). arXiv: 2203.02303 [hep-ex].

[55] Marek Kowalski and. "Neutrino astronomy with IceCube and beyond". In: *Journal of Physics: Conference Series* 888 (Sept. 2017), p. 012007. DOI: 10.1088/1742-6596/888/1/012007. URL: https://doi.org/10.1088/1742-6596/888/1/012007.

[56] Wikipedia contributors. *Tau (particle) — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-April-2022]. 2022. URL: https://en.wikipedia.org/wiki/Tau_(particle).

[57] I. Alikhanov and E.A. Paschos. "Distributions for tau neutrino interactions observed through the decay $\tau \to \mu\nu_\tau\bar{\nu}_\mu$". In: *Physics Letters B* 765 (2017), pp. 272–275. ISSN: 0370-2693. DOI: https://doi.org/10.1016/j.physletb.2016.12.037.

[58] Julien Aublin, Giulia Illuminati, and Sergio Navas. *Searches for point-like sources of cosmic neutrinos with 11 years of ANTARES data*. 2019. DOI: 10.48550/ARXIV.1908.08248. URL: https://arxiv.org/abs/1908.08248.

[59] Arthur L. Samuel. "Some studies in machine learning using the game of Checkers". In: *IBM JOURNAL OF RESEARCH AND DEVELOPMENT* (1959), pp. 71–105.

[60] Donald Michie. "Experiments on the Mechanization of Game-Learning Part I. Characterization of the Model and its parameters". In: *The Computer Journal* 6.3 (Nov. 1963), pp. 232–236. ISSN: 0010-4620. DOI: 10.1093/comjnl/6.3.232. eprint: https://academic.oup.com/comjnl/article-pdf/6/3/232/1030939/6-3-232.pdf. URL: https://doi.org/10.1093/comjnl/6.3.232.

[61] Cornell Tech. *arXiv*. [Online; accessed 18-May-2022]. URL: https://arxiv.org/.

[62] Philippe Fournier-Viger. *Too many machine learning papers?* [Online; accessed 18-May-2022]. 2019. URL: https://data-mining.philippe-fournier-viger.com/too-many-machine-learning-papers/.

[63] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003. URL: http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[64] Laith Alzubaidi et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of Big Data* 8.1 (Mar. 2021). DOI: 10.1186/s40537-021-00444-8. URL: https://doi.org/10.1186/s40537-021-00444-8.

[65] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[66] Xiaomin Fang et al. "Geometry-enhanced molecular representation learning for property prediction". In: *Nature Machine Intelligence* 4.2 (Feb. 2022), pp. 127–134. DOI: 10.1038/s42256-021-00438-4. URL: https://doi.org/10.1038/s42256-021-00438-4.

[67] Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.01.001. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[68] Zhiyuan Liu and Jie Zhou. "Introduction to Graph Neural Networks". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14 (Mar. 2020), pp. 1–127.

[69] Yue Wang et al. "Dynamic Graph CNN for Learning on Point Clouds". In: *CoRR* abs/1801.07829 (2018). arXiv: 1801.07829. URL: http://arxiv.org/abs/1801.07829.

[70] Vincent Dumoulin and Francesco Visin. *A guide to convolution arithmetic for deep learning*. 2018. arXiv: 1603.07285 [stat.ML].

[71] IBM Cloud Education. *What are convolutional neural networks?* [Online; accessed 16-May-2022]. 2020. URL: https://www.ibm.com/cloud/learn/convolutional-neural-networks.

[72] Jason Brownlee. *How to Visualize Filters and Feature Maps in Convolutional Neural Networks*. [Online; accessed 16-May-2022]. 2019. URL: https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/.

[73] 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. DOI: 10.1088/1748-0221/16/07/p07041.

[74] J. Shore and R. Johnson. "Properties of cross-entropy minimization". In: *IEEE Transactions on Information Theory* 27.4 (1981), pp. 472–482. DOI: 10.1109/TIT.1981.1056373.

[75] Pieter-Tjerk De Boer et al. "A tutorial on the cross-entropy method". In: *Annals of operations research* 134.1 (2005), pp. 19–67.

[76] Usha Ruby and Vamsidhar Yendapalli. "Binary cross entropy with deep learning technique for image classification". In: *Int. J. Adv. Trends Comput. Sci. Eng* 9.10 (2020).

[77] D.W. Henderson and E. Moura. *Experiencing Geometry: On Plane and Sphere*. Prentice Hall, 1996. ISBN: 9780133737707. URL: https://books.google.dk/books?id=h1CtQgAACAAJ.

[78] GEERT SCHOU. "Estimation of the concentration parameter in von Mises–Fisher distributions". In: *Biometrika* 65.2 (Aug. 1978), pp. 369–377. ISSN: 0006-3444. DOI: 10.1093/biomet/65.2.369. eprint: https://academic.oup.com/biomet/article-pdf/65/2/369/650211/65-2-369.pdf. URL: https://doi.org/10.1093/biomet/65.2.369.

[79] Sachin Kumar and Yulia Tsvetkov. "Von Mises-Fisher Loss for Training Sequence to Sequence Models with Continuous Outputs". In: *CoRR* abs/1812.04616 (2018). arXiv: 1812.04616. URL: http://arxiv.org/abs/1812.04616.

[80] Ronald Aylmer Fisher. "Dispersion on a sphere". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 217.1130 (1953), pp. 295–305. DOI: 10.1098/rspa.1953.0064. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.1953.0064. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1953.0064.

[81] Geoffrey S. Watson. "Distributions on the circle and sphere". In: *Journal of Applied Probability* 19.A (1982), pp. 265–280. DOI: 10.2307/3213566.

[82] Gerhard Kurz and Uwe Hanebeck. "Stochastic Sampling of the Hyperspherical von Mises–Fisher Distribution Without Rejection Methods". In: Oct. 2015. DOI: 10.1109/SDF.2015.7347705.

[83] M. Ryabinin. *PyTorch Implementation of "Von Mises-Fisher Loss for Training Sequence to Sequence Models with Continuous Outputs"*. https://github.com/mryab/vmf_loss/blob/master/losses.py. 2019.

[84] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[85] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[86] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. DOI: 10.48550/ARXIV.1609.04747. URL: https://arxiv.org/abs/1609.04747.

[87] Mikhail Belkin. "Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation". In: *Acta Numerica* 30 (2021), pp. 203–248.

[88] J. Larsen et al. "Design and regularization of neural networks: the optimal use of a validation set". In: *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*. 1996, pp. 62–71. DOI: 10.1109/NNSP.1996.548336.

[89] The GraphNeT Group. *GraphNeT - Graph Neural Networks for Neutrino Telescope Event Reconstruction*. https://github.com/icecube/graphnet. 2021.

[90] N. S. Altman. "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression". In: *The American Statistician* 46.3 (1992), pp. 175–185. DOI: 10.1080/00031305.1992.10475879. eprint: https://www.tandfonline.com/doi/pdf/10.1080/00031305.1992.10475879.

[91] Michael Larionov. *Explicit AUC maximization*. [Online; accessed 15-May-2022]. 2019. URL: https://towardsdatascience.com/explicit-auc-maximization-70beef6db14e.

[92] The TensorFlow Global Objectives Authors. *Loss functions for learning global objectives*. https://github.com/tensorflow/models/blob/36101ab4095065a4196ff4f6437e94f0d91df4e9/research/global_objectives/loss_layers.py. 2019.

[93] Jörg Martin and Clemens Elster. "Detecting unusual input to neural networks". In: *Applied Intelligence* 51.4 (Oct. 2020), pp. 2198–2209. DOI: 10.1007/s10489-020-01925-8. URL: https://doi.org/10.1007/s10489-020-01925-8.

[94] Jennifer M Franks, Guoshuai Cai, and Michael L Whitfield. "Feature specific quantile normalization enables cross-platform classification of molecular subtypes using gene expression data". In: *Bioinformatics* 34.11 (Jan. 2018), pp. 1868–1874. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty026. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/11/1868/25121531/bty026.pdf. URL: https://doi.org/10.1093/bioinformatics/bty026.

[95] Dalya Baron. *Machine Learning in Astronomy: a practical overview*. 2019. DOI: 10.48550/ARXIV.1904.07248.

[96] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[97] Sheldon M Ross. *A first course in probability*. 9th ed. Pearson Boston, 2019.

[98] Tania Kozynets and Tom Stuttard. *GNN-assisted pulse cleaning for the IceCube-Upgrade*. 2022.

[99] T. Kozynets. *Studies of the mDOM PMT noise and event cleaning*. 2022.

[100] Wing Yan Ma. *Simulation production and monopod update*. 2018.

[101] Tom Stuttard. *Upgrade MC studies update*. 2022.

[102] D. Jason Koskinen. *A New Upgrade for the IceCube Detector — Niels Bohr Institute*. [Online; accessed 15-May-2022]. 2019. URL: https://nbi.ku.dk/english/news/news19/upgrade-of-a-research-icecube/.

[103] Frederick Mosteller. "On Some Useful "Inefficient" Statistics". In: *The Annals of Mathematical Statistics* 17.4 (1946), pp. 377–408. DOI: 10.1214/aoms/1177730881. URL: https://doi.org/10.1214/aoms/1177730881.