Likelihood-free Deep Learning Techniques

in Neutrino Astronomy

Thesis to obtain the academic degree
M.Sc. Physics
at the School of Natural Sciences of the Technical University Munich.

Supervised by	Prof. Dr. Elisa Resconi
	Experimental Physics with Cosmic Particles
Author	Moritz Brandenburg
	Prinzregentenstraße 151
	81677 Munich
Submission Date	31.03.2025

I hereby confirm that this master's thesis in physics is my own work and I have documented all sources and material used.

Garching, 31.03.2025 Moritz Brandenburg

Abstract

This thesis investigates advancements in point source analyses of neutrino data from IceCube, the world's largest neutrino observatory, focusing on deep learning techniques to approximate intractable directional likelihoods. Neural ratio estimation (NRE) and neural posterior estimation (NPE) were explored as new approaches for directly estimating likelihoods from event-level data, bypassing traditional reconstruction steps. A general architecture was developed using a pre-trained backbone model for feature extraction coupled with neural networks for likelihood or posterior estimation.

While NRE demonstrated near-perfect classification performance, the angular resolution remained limited by feature extraction capabilities despite implementing various enhancement strategies. NPE successfully approximated Bayesian posteriors but faced prohibitive inference times due to computational demands of integrating over a time-dependent vector field.

Applications to point source analysis and multi-messenger astronomy alerts produced plausible likelihood contours directly from raw data, though the background events were still dominant and effectiveness was constrained by angular resolution limitations and missing energy information. For alerts, likelihood regions inferred at machine learning speeds showed promise but require further testing and optimization for real-time deployment.

In conclusion, this thesis demonstrates the feasibility of integrating deep learning into neutrino astronomy workflows, achieving direct likelihood estimation from raw data while highlighting the need for improved feature extraction models and faster inference techniques to realize the full potential of these methods.

Contents

1. Introduction

The night sky has long inspired curiosity and driven humanity's quest to understand the universe. From mapping constellations to uncovering the nature of distant galaxies, the study of the cosmos has evolved into a sophisticated scientific endeavor. Today, this exploration extends to extreme environments like the South Pole, where the IceCube Neutrino Observatory is located. IceCube is a massive detector buried deep in the Antarctic ice, designed to study neutrinos - nearly undetectable elementary particles that carry information from some of the most energetic and distant events in the universe.

Neutrinos are unique because they carry no electric charge and only rarely interact with matter, allowing them to travel vast distances without being absorbed or deflected. This makes them valuable for studying astrophysical phenomena that are otherwise hidden from view. By detecting these particles, IceCube provides insights into processes like exploding stars, gamma-ray bursts, and black hole interactions.

This thesis addresses one of the key challenges in neutrino astronomy: identifying point sources - specific locations in the sky where neutrinos originate. Traditional approaches involve complex reconstruction steps and approximations that can be computationally demanding and limiting. This work explores how modern Machine Learning (ML) techniques can simplify and enhance these analyses, offering new tools for extracting meaningful insights directly from raw data.

The structure of this thesis is as follows: Chapter 2 sets the stage by introducing neutrino physics and detection principles. Chapter 3 gives an overview over IceCube as well as the current approach to point source analyses. Chapters 4 and 5 outline the principles of ML and the deep learning framework GraphNeT used in this thesis. Chapter 6 presents the likelihood-free methods that are being investigated, while Chapters 8 and 9 describe their implementation, training and performance. Finally, Chapter 10 gives examples of how these methods could be applied before Chapter 11 summarizes the findings and draws a conclusion.

2. Neutrino Physics

2.1. History of Neutrino Physics

The Standard Model of particle physics has achieved remarkable success in describing three of the four fundamental forces: electromagnetic, weak, and strong interactions. However, it excludes the relatively weak gravitational force. Within the Standard Model, elementary particles are categorized into distinct classes based on their properties [1]. Among these, the three neutral leptons interact solely via weak interactions. These particles were first hypothesized in 1930 by Austrian physicist Wolfgang Pauli, who sought to explain the energy spectra of electrons in beta decay. Later, Italian physicist Enrico Fermi named them neutrinos, meaning "little neutral ones" [2]. The direct detection of neutrinos was not realized until 1956 by Clyde Cowan and Frederick Reines [2].

Subsequent discoveries revealed that neutrinos associated with muons (e.g., from pion decays) differ from those linked to electrons (e.g., from beta decay). This led to the concept of lepton families, which was further expanded with the discovery of the tau lepton in the 1970s [2].

While neutrinos were initially thought to be massless, Bruno Pontecorvo, Ziro Maki, Masami Nakagawa, and Shoichi Sakata, speculated during the 1950s and 1960s about the implications of non-zero neutrino mass [3]. They proposed that neutrinos might exhibit oscillations analogous to K^0-$\overline{K^0}$ mixing if flavor and mass eigenstates were not aligned [4]. In 1998, the Super-Kamiokande experiment provided the first evidence for neutrino oscillations [5], a result subsequently confirmed by other experiments [3]. This discovery implied a mass hierarchy among neutrinos and demonstrated that the Standard Model is incomplete since it considers neutrinos massless. It also resolved the long standing Solar Neutrino Problem, a large discrepancy between measured and expected neutrino flux from the sun. Consequently, many new questions about physics beyond the Standard Model arose and are yet to be answered [3].

2.2. Neutrinos as Cosmic Multi-Messengers

Multi-messenger astronomy integrates traditional electromagnetic observations with studies of cosmic rays, neutrinos, and gravitational waves. This interdisciplinary approach provides deeper insights into astrophysical processes by leveraging the unique information carried by each messenger [6].

Cosmic rays, primarily ionized nuclei such as protons and alpha particles, span a wide energy range. However, their exact sources remain uncertain due to magnetic field deflections

that obscure their trajectories. In contrast, neutrinos are electrically neutral and interact only weakly with matter. Thus, they directly point back to their sources, which are widely believed to also be the sources of cosmic rays which in turn create astrophysical neutrinos through proton interactions [7].

A pivotal moment in multi-messenger astronomy was the detection of neutrinos from Supernova 1987A. These neutrinos arrived hours before the supernova became optically visible, confirming models of core-collapse supernovae where neutrinos escape before the explosion reaches the stellar surface [6, 8]. The detection of solar and supernova neutrinos marked the beginning of the field of neutrino astronomy [9], which has since expanded through numerous experiments targeting diverse energy ranges and origins.

2.3. Cosmic Neutrino Sources

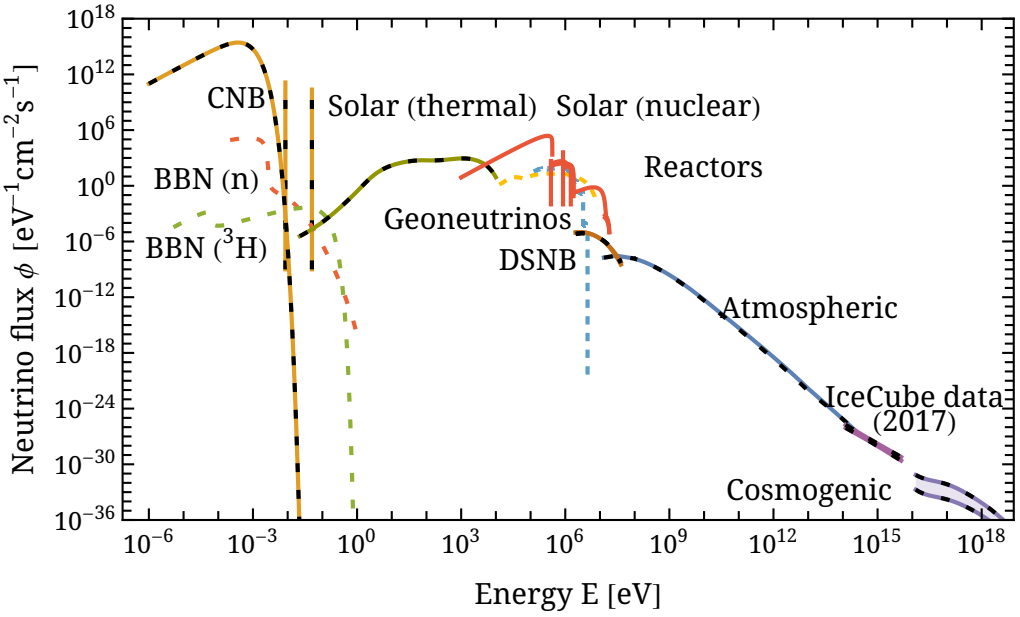

Figure 1 Grand Unified Neutrino Spectrum. Image adapted from [10].

Figure 1 illustrates several models for neutrino fluxes across different energy scales. At high energies, atmospheric neutrinos dominate. These are produced when cosmic rays interact with atmospheric nuclei at high altitudes ($\sim$15 km) [11]. Besides neutral π mesons, that will induce an electromagnetic shower, such interactions primarily generate K and charged π mesons that decay into muons and muon neutrinos (e.g. $p + X \to \pi^+ \to \mu^+ + \nu_\mu$) and form a hadronic shower. The muons subsequently decay into additional neutrinos and electrons or positrons, which usually happens in the atmosphere. However, some high-energy muons may also be able to reach the ground before decaying. As cosmic ray flux decreases rapidly at higher energies, so does atmospheric neutrino flux [11].

Above 50 TeV, astrophysical neutrinos dominate. The IceCube Neutrino Observatory has

detected a diffuse flux of these high-energy neutrinos (Figure 1) [12]. This flux represents an aggregate signal from all cosmic high-energy neutrino sources. Identifying individual sources is challenging due to limited angular resolution and low fluxes at these energies. Notably, in 2017, IceCube linked high-energy neutrinos to a gamma-ray flare from blazar TXS 0506+056, suggesting (jetted) Active Galactic Nuclei (AGNs) as potential sources of astrophysical neutrinos [13].

At ultra-high energies, cosmogenic neutrinos are expected from meson decays following interactions between ultra-high-energy cosmic rays and photons from the Cosmic Microwave Background (CMB) or extragalactic background light. In theory, this interaction sets a limit on the energy of cosmic rays, which is called Greisen Ztsepin Kuzmin (GZK) limit. The detection of neutrinos coming from these interactions remains elusive due to their low fluxes [14].

2.4. Detection Principles

Despite the vast number of extraterrestrial neutrinos reaching Earth every second, detecting these particles remains a significant challenge due to their weak interaction with matter. Neutrino detectors are designed with large target masses to increase the probability of observing interactions between neutrinos and the detector material.

For high-energy neutrinos, deep inelastic scattering of two types, neutral-current (NC) and charged-current (CC), dominates interactions with matter. NC interactions are mediated by the neutral Z^0 boson, while CC interactions involve the exchange of charged $W^{\pm}$ bosons. The primary interaction processes between high-energy neutrinos ν and nuclei X in the detector material are as follows [7]:

$$\nu_\ell + \mathrm{X} \longrightarrow \ell + \mathrm{X}\ (\mathrm{CC}),$$
$$\nu_\ell + \mathrm{X} \longrightarrow \nu_\ell + \mathrm{X}\ (\mathrm{NC}),$$

where ℓ represents a lepton (electron, muon, or tau).

For both processes, the neutrino interacts with a single quark within the nucleus X, leading to the disruption of the target nucleus and the generation of a hadronic shower. In CC interactions involving muon neutrinos ν_μ, a muon is produced that travels through the detector, creating a track-like signature. This track emits Cherenkov radiation when the muon moves faster than the speed of light in the medium. Cherenkov photons result from the coherent relaxation of atoms in the medium that were polarized by the charged particle [15]. These tracks can either originate within the detector or enter it from surrounding materials. An example track event is shown on the left in Figure 2.

In contrast, cascade events are produced by CC interactions of electron neutrinos ν_e and tau

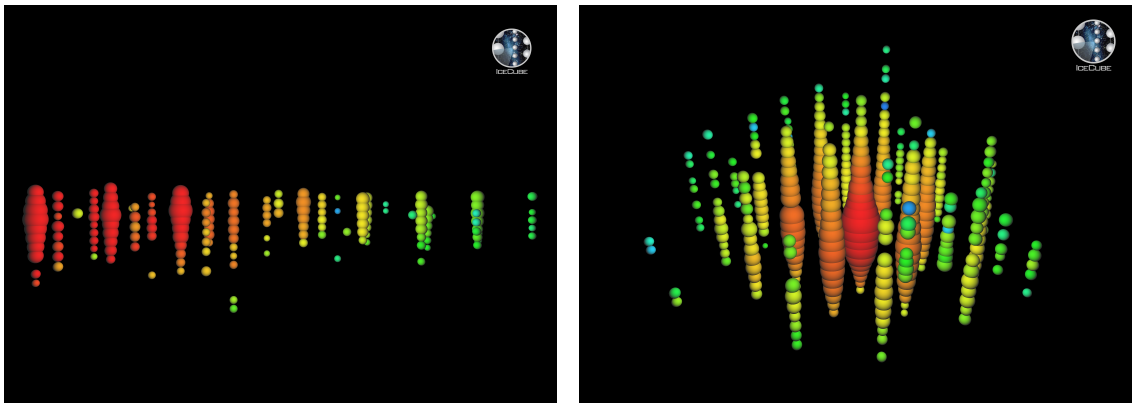

Figure 2 Event topologies recorded by the IceCube Neutrino Observatory. Spheres represent measured charges at individual Digital Optical Modules (DOMs), with size proportional to charge magnitude and color indicating detection time. **Left:** Track event from November 13, 2010, showing a muon traversing from left to right. **Right:** High-energy cascade event from January 3, 2012, produced by a neutrino with estimated energy of 1.14 PeV. Credit: IceCube Collaboration

neutrinos ν_τ, as well as all NC interactions. These events generate hadronic cascades, and for ν_e CC interactions, an additional forward electromagnetic cascade is formed. Cascades have shorter characteristic lengths compared to tracks, resulting in lower angular resolution but improved energy resolution for contained events since most of the neutrino's energy is deposited within the detector volume [7]. The topology of a cascade event can be seen on the right in Figure 2.

2.5. Large-Volume Neutrino Experiments

For atmospheric and astrophysical neutrinos, expected neutrino fluxes decrease rapidly with higher energies (see Figure 1). Consequently, large-volume neutrino observatories are necessary to achieve sufficient target mass for detection. These observatories typically utilize natural media such as water or ice to form detection volumes spanning hundreds of cubic meters to even cubic kilometers.

The design of large-volume neutrino telescopes is usually very similar: long vertical mooring lines equipped with DOMs are deployed into the medium. These optical modules detect the Cherenkov radiation emitted from neutrino interactions. To minimize background noise from atmospheric muons, detectors are positioned deep below the earth's surface.

For high-energy telescopes like IceCube, key signatures are upgoing muon tracks since Earth acts as a filter, blocking atmospheric muons while allowing astrophysical neutrinos to pass through. However, at energies above 10 TeV, Earth becomes increasingly opaque to neutrinos due to their rising interaction cross-section. As a result, optimal acceptance of upgoing tracks is limited to approximately 20–30 degrees above the telescope's horizon [16].

To complement IceCube, several additional large-volume experiments are envisioned or un-

der development, such as Baikal-GVD in Lake Baikal, Russia, KM3NeT in the Mediterranean Sea and P-ONE in the Pacific Ocean. Combined, sensitivity to astrophysical neutrinos can improve by up to two orders of magnitude depending on the declination and spectral index [16]. A combined sensitive area covering nearly the entire sky is depicted in Figure 3, which also illustrates reconstructed directions for detected neutrinos linked to blazar TXS 0506+056 and AGN NGC 1068.

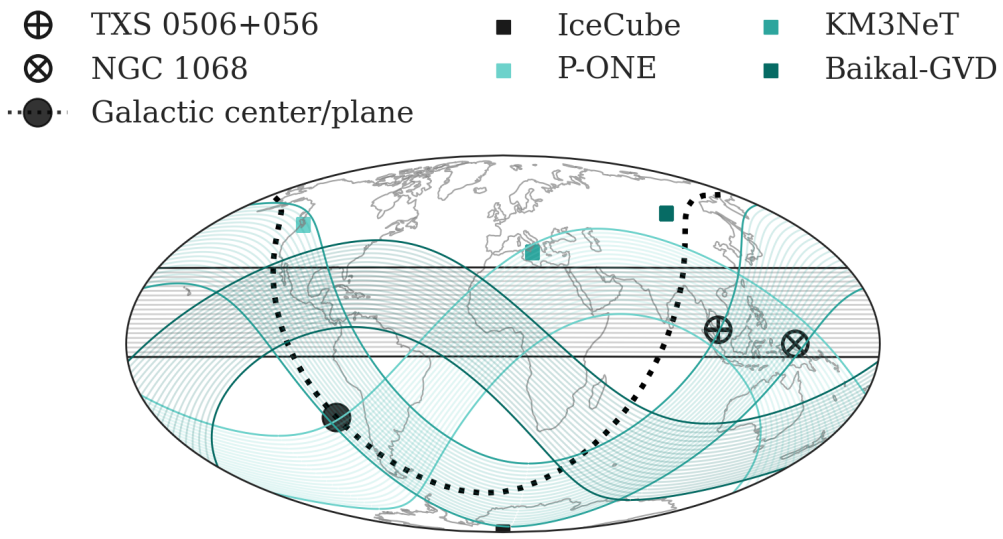

Figure 3 The combined sensitive area of Planetary Neutrino Monitoring System (PLEnuM) will almost cover the entire sky. The image also maps the reconstructed direction of the detected neutrinos that were linked to the blazar TXS 0506+056 and the AGN NGC 1068. Image courtesy of L. Schumacher (TUM).

The focus of this thesis is on the IceCube Neutrino Observatory and new statistical methods for making discoveries that link detected neutrinos to astrophysical sources. A detailed description of IceCube's detector design and data analysis methodology will be provided in the next chapter.

3. The IceCube Neutrino Observatory

3.1. Research Mission

The IceCube Neutrino Observatory, situated at the Amundsen-Scott South Pole Station in Antarctica, operates as a cubic-kilometer neutrino detector [17]. The current configuration of the detector has been taking data for almost 14 years (full operations began on May 13, 2011). The facility is managed by an international collaboration of over 350 scientists from 58 institutions across 14 countries [18]. Its location at the geographic South Pole requires specialized infrastructure and dedicated personnel for continuous operation in the polar environment [19].

The observatory's primary scientific objectives include the detection and study of high-energy neutrinos from astrophysical sources, identification of cosmic ray accelerators, and exploration of neutrino properties at energies beyond terrestrial accelerators [17]. Additional research goals encompass neutrino oscillation studies and other searches for physics beyond the Standard Model, including potential dark matter signatures [18]. As the biggest neutrino observatory in the world, IceCube serves as a key component in multi-messenger astronomy, coordinating observations with gamma-ray, optical, X-ray, and gravitational wave facilities [18].

3.2. Design

The IceCube detector comprises over 5,000 DOMs embedded in ultra-transparent glacial ice at depths between 1,450 m and 2,450 m at the geographic South Pole [19]. Each DOM houses a Photomultiplier Tube (PMT), onboard digitization electronics, and calibration light sources. The modules are connected to vertical strings lowered into the ice via a hot-water drilling technique designed to create holes more than two kilometers deep. These strings are arranged on a hexagonal grid to achieve a cubic-kilometer detector volume. Additional sensors in the denser DeepCore region allow for lower energy threshold studies, while IceTop detectors at the surface measure cosmic-ray air showers up to EeV energies [19]. A scheme of the detector can be found in Figure 4.

Prior to deployment, each DOM undergoes rigorous testing to ensure high reliability in the harsh polar environment. Once installed, DOMs operate independently, while onboard triggering and local coincidence logic filter spurious or isolated hits, reporting only relevant information to the surface data acquisition system [19]. IceCube achieves uptimes of around 99% and boasts over 98% of modules functioning successfully after many years of continuous operation [19].

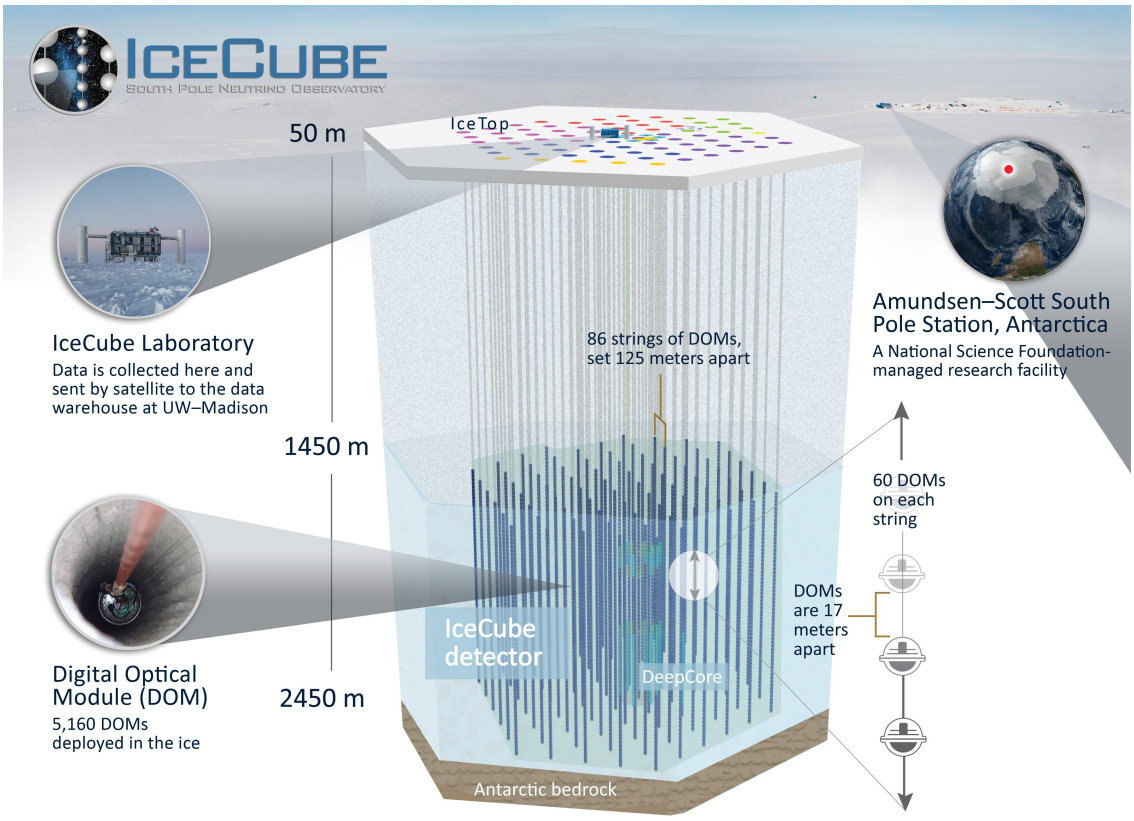

Figure 4 Schematics of the IceCube Neutrino Observatory. It instruments a volume of roughly one cubic kilometer of clear Antarctic ice at the South Pole. The observatory includes a densely instrumented subdetector, DeepCore, and a surface air-shower array, IceTop. Credits: IceCube/NSF

Two major upgrades are planned to enhance IceCube's capabilities. The IceCube Upgrade, scheduled for 2025/2026, will deploy 750 advanced photodetectors and calibration devices [20]. These improvements will enhance the detector's sensitivity and angular resolution, with benefits applicable to both new and existing data [20].

IceCube-Gen2 is envisioned to become the second phase of expansion. This upgrade would increase the instrumented volume to approximately 8 cubic kilometers through 120 additional strings with enhanced optical sensors. The project includes a 500 square kilometer surface array and aims for a fivefold increase in sensitivity to astrophysical neutrinos [21]. However, IceCube-Gen2 is yet to be funded and would need years of development and construction.

3.3. Research Results

The IceCube Neutrino Observatory has achieved several groundbreaking discoveries in neutrino astronomy. A key milestone was reached in 2013 with the first observation of a diffuse flux of astrophysical neutrinos extending to PeV energies[12]. Recent analyses have successfully identified several neutrino sources. The Seyfert galaxy NGC 1068 emerged as the first steady neutrino source, detected at 4.2σ significance [18]. Additionally, the blazar TXS

0506+056 was identified during a multi-messenger campaign, providing the first evidence for neutrino emission coincident with a gamma-ray flaring event [13]. A significant milestone was reached with the detection of neutrino emission from the Galactic Plane, raising questions about the contribution of yet unknown cosmic-ray accelerators within our galaxy [18].

The observatory has expanded its reach to lower energies through novel detection techniques. Using starting track events, IceCube has successfully measured the astrophysical neutrino flux down to 3 TeV [22]. This achievement required sophisticated background rejection methods, as cosmic neutrinos are outnumbered by atmospheric backgrounds by almost a factor of 1 billion to one (3000 Hz atmospheric background rate vs. 100 expected astrophysical neutrinos per year) [22].

3.4. Directional Event Reconstruction

A big factors for the success of a large-scale particle experiment like IceCube is the event reconstruction. Event reconstruction bridges the gap between the raw signal that the detector measures and the actual physical event in the detector that creates the measured detector response. Based on the PMT photoelectron pulses and time stamps, the event reconstruction of IceCube tries to estimate features like direction of the neutrino and energy. First simple reconstructions are already happening during data processing stages in order to apply quality cuts to suppress background events in the data. After processing, more sophisticated algorithms are used for the final reconstructions [23].

As described in Section 2.4, the different types of neutrino interactions create different patterns in the detector response. The most important type of event for analyses that lead to the discovery of point sources like NGC 1068 are CC ν_μ interactions. The muons produced in these interactions create long track-like charge patterns that make accurate directional reconstruction possible. While the muon's trajectory does not precisely align with the original neutrino direction, the mean kinematic opening angle ψ between the neutrino and muon decreases at higher energies [24]:

$$\psi = 0.7^\circ \left(\frac{E_\nu}{1\,\text{TeV}} \right)^{-0.7} \tag{3.1}$$

This opening angle imposes a fundamental limit on the achievable angular resolution of any reconstruction method, particularly at lower energies.

The reconstruction algorithm of choice for track-like events in IceCube is called SplineMPE. This method calculates a likelihood function based on photon arrival time residuals in the different DOMs. For that, a Monte Carlo photon propagation simulation creates time residual Probability Density Functions (PDFs) for light sources at different locations and detector ori-

entations [23]. These high-dimensional PDFs are interpolated by so called penalized basis splines which pose as smooth and memory efficient representation of the PDFs [25].

While under ideal conditions, all photon hits at each DOM could be included in the likelihood function. However, SplineMPE only incorporates first photon arrival times $t^1_{\text{res},i}$. This is due to the fact that PMT effects like late pulses or afterpulses have too much of a negative effect on the accuracy of the "Multi Photo Electron (MPE)" likelihood function [23], which is defined as

$$\mathcal{L}_{\text{MPE}} = \prod_i^{N_{\text{DOMs}}} p(t^1_{\text{res},i}|\mathbf{x_i}, H) \left(\int_{t^1_{\text{res},i}}^{\infty} p(t|\mathbf{x_i}, H) dt \right)^{N\text{pulses},i-1} . \tag{3.2}$$

In this equation, for every DOM at position $\mathbf{x_i}$, the likelihood of the first hit under the track Hypothesis H, $p(t^1_{\text{res},i}|\mathbf{x_i}, H)$, is multiplied by the probability of subsequently measuring the given number of additional pulses. To obtain the best fitting direction, the likelihood function is maximized (usually by minimizing the negative log-likelihood for numerical stability). This results in median angular resolutions of below $0.5\,^\circ$ for TeV energies and above [23].

IceCube employs two primary methods to estimate the uncertainty of reconstructed event directions, crucial for assessing the probability that an event originates from a specific location, such as an astrophysical source: the Paraboloid algorithm and a multivariate estimation method based on Boosted Decision Trees (BDTs). The Paraboloid algorithm approximates the SplineMPE log-likelihood function around the best-fit position using a two-dimensional parabola [23]. This approach assumes that the angular resolution follows a normal distribution around the true direction. To account for the kinematic angle ψ, an energy-dependent correction factor estimated from the simulations shifts the median angular uncertainty. The BDT method, on the other hand, uses 17 observables, including the Paraboloid angular uncertainty and angular differences between different track reconstruction methods. The BDT then approximates median angular separation between the reconstructed and true muon directions, improving reliability, especially when the Paraboloid estimation fails [23].

3.5. Energy Reconstruction

The second most notable observable is the energy. It is crucial in distinguishing between atmospheric and astrophysical events and in characterizing the energy spectrum of neutrino sources, which could hint at the different mechanisms within the source that give the neutrinos their energy. However, reconstructing the energy is a difficult task, because usually not the whole muon track is fully contained in the detector (compared to a cascade event). For muon energy reconstruction, IceCube employs multiple techniques. Either technique tries to find the average energy loss per propagation length $\frac{dE}{dx}$, from which the muon energy can be

inferred.

The Truncated Energy algorithm tries to mitigate the impact of PMT saturation. This approach calculates energy losses for each DOM within a $60\,\mathrm{m}$ radius of the track and excludes the $50\,\%$ with the highest observed-to-expected pulse ratio. The energy losses of the remaining DOMs is then averaged to find the estimated $\frac{dE}{dx}$. While high-energy events are typically well reconstructed, the algorithm has problems with low-energy events, likely due to truncating an already small number of pulses [23].

A second method is based on Dense Neural Networks (DNNs), which is a ML technique further explained in Chapter 4.2. This technique leverages Artificial Intelligence (AI) to extract energy information from various detector observables, showing enhanced performance at lower energy levels. These networks take DOM pulse maps as input and utilize input features such as the total charge and charge distribution to estimate the muon energy upon entering the detector volume. By training on a large dataset and optimizing network parameters, DNNs achieve energy resolutions that surpass traditional methods, particularly for events in the lower energy range [23].

With emerging computational and ML techniques, event reconstruction in IceCube continues to evolve. A tool that is being developed within the group is the python framework GraphNeT, which applies newest methods like Graph Neural Networks (GNNs) and transformer networks to all kinds of reconstruction tasks. These methods leverage spatial and temporal relationships between detector signals to improve reconstruction accuracy and computational efficiency, particularly for low-energy events where traditional methods face challenges. GraphNeT's flexibility in handling irregular detector geometries and real-time processing capabilities positions it as a promising tool for future IceCube analyses and next-generation neutrino telescopes like IceCube-Gen2. GraphNeT's approach will be described in detail in Chapter 5.

3.6. Point source analyses in Neutrino Astronomy

Obtaining the results described in Chapter 3.3 was only possible through advanced methods of statistical inference. This chapter aims to describe the process of unbinned maximum likelihood analyses which are widely used for the search of neutrino emission from point-like sources like NGC 1068 and TXS 0506+056 [23].

Point-like neutrino sources are thought to create clusters of astrophysical events against the atmospheric neutrinos and diffuse astrophysical background [23]. When analyzing detected events, the point-source search method compares two scenarios, also called hypotheses, through the frequentist approach of likelihood ratio tests. The null hypothesis H_0 defined by the set of parameters $\boldsymbol{\theta}_0$ postulates that observed events consist solely of atmospheric neutrinos and diffuse astrophysical background. In contrast, the alternative hypothesis H_1

with the parameters $\boldsymbol{\theta}_s$ incorporates an additional component from a point source at celestial coordinates (right ascension α_s, declination δ_s), emitting neutrinos with energy spectrum $\Phi(E) = \Phi_0 E^{-\gamma}$ [23].

Under the Neyman-Pearson lemma, the most powerful Test Statistic (TS) for distinguishing $\boldsymbol{\theta}_0$ from alternative hypotheses is the likelihood ratio test statistic Λ [26]:

$$\Lambda(\mathbf{x}) = \frac{\sup_{\theta \in \boldsymbol{\theta}_0} \mathcal{L}(\boldsymbol{\theta}|\mathbf{x})}{\sup_{\theta \in \boldsymbol{\theta}_s} \mathcal{L}(\boldsymbol{\theta}|\mathbf{x})}. \tag{3.3}$$

Here, the supremum is taken over all parameters that describe each hypothesis. In order to evaluate this TS, the unbinned likelihood $\mathcal{L}(\boldsymbol{\theta}|\mathbf{x})$ has to be found. For a given set of parameters $\boldsymbol{\theta}$ and observed data $\boldsymbol{x}$, it is defined as

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{x}) = \prod_i f(x_i|\boldsymbol{\theta}), \tag{3.4}$$

in which the product over the PDFs $f(x_i|\boldsymbol{\theta})$ for all observations x_i is taken [23].

For point source analyses in neutrino astronomy, the parameters $\boldsymbol{\theta}_0$ for the background-only hypothesis are fixed by the atmospheric and astrophysical diffuse event rates [23]. The signal hypothesis is described by a set of four free parameters $\boldsymbol{\theta}_s = (\alpha_s, \delta_s, \Phi_0, \gamma)$, in which spatial information of the source are incorporated via right ascension α_s and declination δ_s, and information regarding the energy spectrum of the source are incorporated via flux normalization Φ_0 and spectral index γ. While γ depends on the mechanisms with which a source gives the neutrinos their energy, Φ_0 is dependent on the total expected number of signal events n_s via the event rate equation [23]:

$$n_s = T \int_0^\infty dE_\nu A_{\text{eff}}(E_\nu, \delta_\text{s}) \times \phi_0(E_\nu). \tag{3.5}$$

This equation incorporates the exposure time T, the effective area of the detector for the source location A_{eff} as well as the flux normalization Φ_0 and integrates over all neutrino energies E_ν to obtain the expected number of signal events n_s. Given a background flux, the same can be done for the expected number of background events n_b. This can be used to extend the likelihood function by creating a mixture model of signal PDF $f_s(\mathbf{x}|\boldsymbol{\theta}_s)$ and background PDF $f_b(\mathbf{x}|\boldsymbol{\theta}_0)$ weighted by the expected number of signal and background events [23]:

$$\mathcal{L}(n_s, \gamma|\mathbf{x}) = \frac{(n_s + n_b)^N}{N!} e^{-(n_s+n_b)} \prod_{i=1}^{N} \left[\frac{n_s}{n_s + n_b} f_s(x_i|\boldsymbol{\theta}_s) + \frac{n_b}{n_s + n_b} f_b(x_i|\boldsymbol{\theta}_0) \right]. \tag{3.6}$$

The term in the beginning of the equation describes a Poisson distribution with a mean at the total number of expected events, i.e. $n_s + n_b$. As described in [27], the likelihood function can be extended like this to account for statistical fluctuations in the total number of observed events. The formulation of the likelihood reduces to the background-only hypothesis for $n_s =$

0, which shows that the two hypothesis that are being tested in a point-source analysis are nested. This fact validates Wilks' theorem, which states that for large $N \to \infty$, two times the negative logarithm of the likelihood-ratio (3.3) assuming the null-hypothesis to be true asymptotically approaches a χ^2 distribution [23]. The degrees of freedom n_{dof} of the χ^2 distribution equals the difference between the number of free parameters in both hypothesis, which in our case is $n_{dof} = 2$. This arises from the fact that we are evaluating a specific point in the sky, i.e. a fixed right ascension and declination, while in the signal hypothesis the flux normalization Φ_0 (or similarly the number of signal events n_s) and spectral index γ remain free parameters. This fact helps in assigning the obtained TS a significance which makes the results of the hypothesis tests interpretable. Putting all of this together, the TS typically used in these point source analyses reads [23]

$$TS = -2 \log \left[\frac{\mathcal{L}(n_s = 0)}{\sup_{n_s,\gamma} \mathcal{L}(n_s, \gamma)} \right]. \tag{3.7}$$

At the heart of this TS are the PDFs for both background and signal. They are based on the events' reconstructed energy and direction. Accurately modeling them is crucial for distinguishing potential astrophysical sources from background noise. The signal PDF, $f_s(x_i|\boldsymbol{\theta}_s)$, describes the probability of observing an event with characteristics x_i (reconstructed direction, energy, angular uncertainty) given a source with right ascension α_s, declination δ_s and spectral index γ. It often appears factorized into a spatial and energy term. The background PDF, $f_b(x_i)$, represents the expected distribution of atmospheric neutrinos and other background events, typically modeled as independent of right ascension due to the detectors location at the South Pole and the inherited detector symmetry over long periods of time [23].

Analytically constructing these PDFs is exceedingly difficult due to complex detector effects and energy-dependent angular resolution. Traditional methods used Gaussian approximations for the spatial term, but modern analyses employ multidimensional Kernel Density Estimation (KDE) to better capture non-Gaussian tails and the dependence on the spectral index γ and the reconstructed energy [23]. KDEs estimate the PDF directly from Monte Carlo simulations without the need for assumptions about the data distribution. It does so by sampling and weighting a large set of simulated events for a given range of spectral indices. A KDE approximation is then fitted to the corresponding Monte Carlo distributions in order to model the spatial and the energy term of the signal PDF for each spectral index γ. These KDE based PDFs provide a more accurate representation of the signal and background distributions, leading to improved sensitivity and discovery potential, especially at lower energies and for softer energy spectra [23].

The modeling of the PDFs plays a key role in the success of statistical data analyses for experiments like IceCube. This thesis aims to explore further methods to construct the PDFs and subsequently the likelihood functions for these type of analyses. The methods are based

on ML techniques and aim to leverage the advantages that AI models can bring, such as fast and flexible inference and an amortization of initial simulation and training. The term "likelihood-free" stems from the fact that no explicit construction of the likelihood function has to be performed.

The next chapters give an overview over the fundamentals of ML, before introducing the Deep Learning library for neutrino astronomy called GraphNeT. GraphNeT will be used to implement the likelihood-free methods described in Chapter 6. Afterwards, the actual implementations as well as the process of training the methods will be described, before the methods are tested example alert events and a pseudo dataset from a source similar to NGC 1068.

4. Machine Learning

4.1. Basic Principles of Machine Learning

ML is a sub-field of AI dedicated to creating algorithms capable of automatically learning patterns from data, with minimal explicit human instruction. Instead of following static rules, ML models utilize mathematical optimization techniques to refine their parameters and enhance predictions over time.

In principle, ML aims to map inputs - such as images, text sequences, or sensor signals - to outputs that for example can be labels, real-valued predictions, or latent structures. For that, a ML model takes the input or a representation of the input and applies (often times non-linear) mathematical operations that aim to return the desired output.

For a ML model to being able to do that, it usually has to be trained on examples, which is a fundamental part in ML. Typically, two main frameworks of learning (and combinations of those) are being used. The distinction lies within whether the training data is labeled, i.e. whether there is a known ground truth label for each training data input. If that is not the case, unsupervised learning methods such as clustering or feature learning can be used to learn intrinsic patterns in the given data. Supervised learning on the other hand has access to the target labels and can therefore compare the output of the ML model with the desired output for that training example [28].

In many cases, the objective of a ML model that has been trained by supervised learning can be categorized into the two main branches, classification and regression [28]. In classification, the model is trained to assign inputs to one of several predefined categories. An example would be labeling images as containing "cats" or "dogs". Classification tasks often measure performance using metrics such as accuracy, precision, and recall, reflecting how reliably the model places instances in correct classes. In regression, the model predicts a continuous-valued output. Typical scenarios include forecasting stock prices, estimating temperature, or predicting real-valued sensor readings. Here, performance can be measured by calculating the deviation from the true numeric values.

Whether the task is classification or regression, loss functions quantify how far off the model's predictions are from the desired labels or targets. The choice of loss function directly impacts a model's training dynamics and final performance, as the training algorithm tries to minimize this loss function by adjusting the model's parameters [28].

For classification, a common loss function is the cross-entropy loss, also known as the logistic loss or softmax loss [28]. Suppose the label is given by a one-hot vector $\mathbf{y}$ and the model's

predicted probabilities are $\hat{\mathbf{y}}$. The cross-entropy loss for a single example is:

$$\mathcal{L}_{\mathrm{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i y_i \log(\hat{y}_i), \tag{4.1}$$

where $\hat{y}_i$ is the predicted probability for class i. Minimizing cross-entropy encourages the model to assign high probability to the correct class and low probability to incorrect classes.

For regression, a frequently used loss function is the Mean Squared Error (MSE) [28]. Let t be the true continuous value and $\hat{t}$ be the model's prediction. The MSE for a single example is

$$\mathcal{L}_{\mathrm{MSE}}(t, \hat{t}) = \frac{1}{2}(t - \hat{t})^2. \tag{4.2}$$

The factor of $\frac{1}{2}$ is often included for convenience when taking derivatives. Minimizing MSE drives the model to reduce the squared difference between the true value and its prediction across all training examples [28].

A critical challenge in ML is balancing model complexity to navigate the bias-variance trade-off, manifesting as either overfitting or underfitting. Overfitting occurs when a model learns the training data too precisely, including random noise, leading to poor generalization to new, unseen data. Underfitting arises when the model is too simple to capture the underlying patterns in the dataset. Methods such as cross-validation, L1 or L2 regularization, and systematic hyperparameter tuning help mitigate these issues [28]. While these algorithmic approaches provide essential regularization, the most reliable foundation for robust learning remains a sufficiently large, well-balanced training dataset with good data quality that comprehensively samples the input space and edge cases [28].

4.2. Deep Learning

Deep Learning is a specialized sub field of ML that leverages neural networks with multiple layers, known as DNNs, to automatically learn complex representations of data. Unlike traditional ML models that often require manual feature engineering or extraction, deep learning models can learn features on different levels of abstractions directly from raw input data [29].

At the core of deep learning are artificial neurons, which are the building blocks of neural networks. An artificial neuron computes a weighted sum of its inputs $\mathbf{x}$ based on a weight vector $\mathbf{w}$, adds a bias term b, and then applies an activation function σ to introduce non-linearity [28]:

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b). \tag{4.3}$$

A Multilayer Perceptron (MLP) consists of multiple layers of such neurons arranged in a feed-forward manner. Each layer transforms the input data into higher-level abstractions, enabling the network to model complex relationships [29].

Activation functions are critical in neural networks as they introduce non-linearity, enabling the network to learn and model complex patterns. Without activation functions, multiple layers would effectively collapse into a single linear transformation, limiting the network's expressiveness.

A common activation function is the sigmoid function defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{4.4}$$

which maps inputs to a range between 0 and 1. While historically popular, it can suffer from vanishing gradients, making it less effective for deep networks [28].

More popular nowadays is the Rectified Linear Unit (ReLU), expressed as

$$\mathrm{ReLU}(x) = \max(0, x). \tag{4.5}$$

ReLU is computationally efficient and has become the default activation function in many deep learning architectures due to its ability to mitigate the vanishing gradient problem [29]. However, it can suffer from the dying ReLU problem, which occurs when neurons consistently output zero for all inputs. In this case, the gradient for that neuron will always be zero, so the neuron effectively "died" because it cannot recover during training [30]. To mitigate this issue, modified versions of the ReLU function have emerged, like Leaky ReLU [29]:

$$\mathrm{LeakyReLU}(x) = \max(\alpha \cdot x, x), \tag{4.6}$$

in which α is a constant factor between 0 and 1 (usually around $\alpha = 0.01$). Leaky ReLU allows a small, non-zero gradient for negative inputs, which prevents neurons from becoming completely inactive.

Training deep neural networks involves adjusting the weights and biases to minimize a loss function that measures the discrepancy between the model's predictions and the actual targets. The primary algorithm used for this purpose is backpropagation, which efficiently computes the gradient of the loss function with respect to each parameter in the network [29]. The before mentioned problem of vanishing gradients occurs when gradients become exponentially small during backpropagation through deep networks. This happens primarily due to activation functions with bounded derivatives and the multiplicative nature of the chain rule, resulting in earlier network layers receiving negligible parameter updates and thus failing to learn.

The training process typically follows these steps [29]:

1. **Forward Pass**: Input data is passed through the network layer by layer, computing activations at each neuron to generate the final output.
2. **Loss Computation**: The loss function, such as mean squared error for regression or cross-entropy loss for classification, quantifies the prediction error.
3. **Backward Pass**: Gradients of the loss with respect to each parameter are computed using the chain rule of calculus. This step propagates the error backward through the network.
4. **Parameter Update**: Optimizers like *Adam*[1] adjust the network's parameters in the direction that minimizes the loss.

This iterative process continues until the model converges to a minimum of the loss function or meets predefined stopping criteria.

Beyond the MLPs, several specialized neural network architectures have been developed to handle different types of data and tasks effectively. Convolutional Neural Networks (CNNs) are tailored for processing grid-like data such as images and can capture local spatial hierarchies while reducing the number of parameters and improving computational efficiency [31]. The architecture of CNNs typically consists of convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters to detect features, pooling layers reduce spatial dimensions, and fully connected layers perform high-level reasoning. This structure allows CNNs to automatically learn hierarchical representations of visual data, making them particularly effective for tasks such as image classification, object detection, and image segmentation [31].

GNNs are a class of neural networks designed to operate on graph-structured data, where information is represented as nodes and edges. GNNs can capture complex relationships and dependencies between entities in a graph, making them particularly useful for tasks involving relational data [32]. The key idea behind GNNs is message passing, where each node aggregates information from its neighbors to update its representation. This process allows GNNs to learn both local and global graph properties. Applications of GNNs include social network analysis, recommender systems, molecular property prediction, and protein folding prediction as demonstrated by AlphaFold [33]. GNNs have shown good performance in tasks where the relationships between entities are as important as the entities themselves.

Recurrent Neural Networks (RNNs) are designed for sequential data, like time series or language, by maintaining a hidden state that captures information from previous time steps. Advanced variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)

[1] *Adam* documentation can be found **here.**

address issues like vanishing gradients, enabling the modeling of long-range dependencies [34]. The key feature of RNNs is their ability to process sequences of varying lengths and maintain context across time steps. This makes them particularly suitable for tasks such as natural language processing, speech recognition, and time series forecasting. LSTMs, for instance, introduce a memory cell and gating mechanisms to control information flow, allowing the network to learn when to remember or forget information over long sequences [34].

In recent years however, so called Transformers emerged and surpassed the performance of RNNs. Transformers utilize attention mechanisms to process entire sequences in parallel, significantly improving scalability and performance on tasks such as language translation and text generation. Architectures like BERT and GPT are prominent examples [35]. The core innovation of Transformers is the self-attention mechanism, which allows the model to weigh the importance of different parts of the input when processing each element. This approach eliminates the need for sequential processing, enabling more efficient training on large datasets. Transformers have revolutionized natural language processing, achieving state-of-the-art performance on a wide range of tasks including machine translation, text summarization, and question answering [35].

5. GraphNeT: A Deep Learning Framework for Neutrino Physics

5.1. Introduction

With the advancements in ML in recent years, also science and physics have stepped into a new era leveraging advanced computational techniques. Within astroparticle physics, GraphNeT has emerged as an open-source deep learning library specifically designed for neutrino telescope applications [36]. This chapter explores the architecture, implementation, and applications of GraphNeT, and describes how a public data science competition helped improving state of the art reconstruction models. Afterwards, it is used as the framework to implement the likelihood-free deep learning techniques.

5.2. Library Architecture

GraphNeT's architecture is built around several key components to provide a comprehensive solution for neutrino physics applications. At the heart of the library is the *DataConverter* functionality, which employs a reader-writer scheme to handle experiment-specific file formats and transform them into standardized representations suitable for the GraphNeT framework [36]. All that has to be defined is a method to read and extract the data from the given experiment files for the *DataConverter* to transform the data to a supported backend. Additionally, GraphNeT requires a *Detector* class containing a geometry table of the sensors, standardization functions that map the specific raw data into numerical ranges suitable for deep learning, and a list of names of the relevant columns of the data tables. This approach allows for easy handling of diverse data sources and ensures that the library can adapt to new experimental setups with minimal modifications while also facilitating the transfer of models between different neutrino detectors [36].

GraphNeT's *Model* component adheres to three core ideals: self-containment (portable functionality requiring only raw data inputs), summarizability (fully exportable configurations for reproducibility), and reusability (cross-experiment adaptability) [36]. The framework provides a generic *Model* class supporting diverse deep learning paradigms — from GNNs to Transformers — while enabling modular component replacement. The flexibility is provided through pytorch lightning's *LightningModule* which the *Model* class inherits. For most applications, the *StandardModel* subclass structures workflows through three key components: the *GraphDefinition*, which handles the detector geometry and graph construction via *Detector*, *NodeDefinition*, and *EdgeDefinition* modules, the *Backbone*, which defines the neural architecture like GNN or transformer, and the *Task*, which consists of problem-specific output layers and loss functions [36]. The *GraphDefinition* isolates detector-specific details, allowing physics-

agnostic feature extraction through the *Backbone*. For example, replacing IceCube's Detector module with KM3NeT's preserves core functionality while adapting to new geometries [36]. The *Task* component maps latent representations to physical quantities like energy or direction, supporting supervised tasks through configurable loss functions. Lastly, models are saved as configuration files and trained weights, facilitating deployment in reconstruction pipelines and collaboration across experiments [36].

5.3. DynEdge Architecture

The DynEdge architecture forms the foundation of GraphNeT's reconstruction capabilities for low-energy neutrino events in IceCube, employing an innovative implementation of the EdgeConv graph convolution operator [32]. This operator generalizes traditional convolution to irregular graph structures by processing node features through learned relationships between spatially connected detector elements.

At its core, EdgeConv operates on k-nearest neighbor graphs where nodes represent individual PMT pulses. For each node n_j with features $\mathbf{x}_j$, the operator computes:

$$\tilde{\mathbf{x}}_j = \sum_{i=1}^{k} \text{MLP}\left(\mathbf{x}_j, \mathbf{x}_j - \mathbf{x}_i\right) \tag{5.1}$$

where the MLP processes both the node's features and its relative differences with neighboring nodes n_i [32]. This formulation enables feature learning that combines absolute detector coordinates with relative spatial-temporal relationships between pulses.

DynEdge enhances this operator through four sequential EdgeConv blocks with dynamic edge updates. The DynEdge architecture employs a dynamic graph connectivity approach where each convolutional block recalculates the k-nearest neighbors in the evolving latent feature space. This continuous re-evaluation of node relationships enables the network to adaptively learn optimal connectivity patterns tailored for successive processing stages, effectively capturing both local and global event characteristics as feature representations evolve [32].

The feature hierarchy follows a progressive abstraction paradigm, with initial EdgeConv blocks processing raw detector geometry features D_{xyz}, temporal information t, and charge measurements q. Subsequent layers operate on increasingly abstract latent representations, enabling hierarchical feature learning that transitions from concrete detector-level observations to physics-relevant pattern recognition [32]. To preserve information from each level of abstraction, node features from all four EdgeConv blocks undergo concatenation prior to final pooling operations, ensuring the model retains both fine-grained, local details from early processing stages and high-level, global abstractions from deeper layers [32].

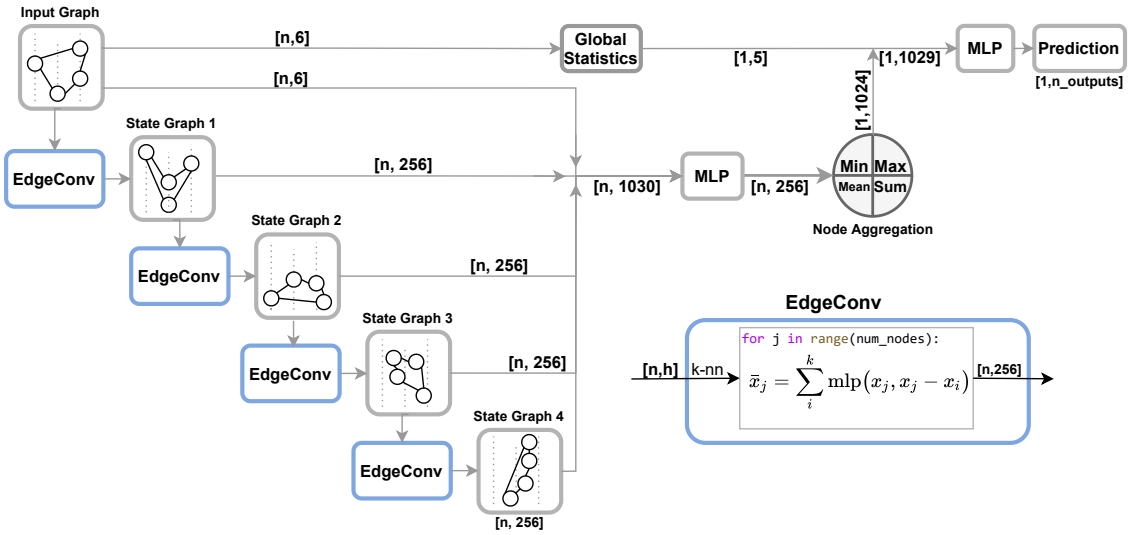

Figure 5 A schematic representation of the DynEdge architecture. The diagram shows a simplified "Input Graph" visualization, followed by "State Graphs" that demonstrate the evolution of node positions and connectivity patterns after EdgeConv operations. Taken from [32] with R. Ørsøe's permission.

In addition to the EdgeConv blocks, two methods have been implemented to further enhance the architecture's performance. First, five global statistics are calculated and injected – including homophily ratios of spatial-temporal correlations and total pulse counts – directly into node feature vectors, providing environmental context for interpreting local detector signals [32]. Second, the adaptive pooling scheme after concatenation of the outputs of the different EdgeConv blocks employs four parallel aggregation operations (minimum, maximum, mean, and summation) across node features, which adds characterization of the distributed pulse information through summary statistics [32]. This also helps in removing the need for zero-padding of the input data and allows for the data to effectively contain any number of pulses.

As shown in [32], this design achieves 13-20% improvement in resolution of energy, direction and interaction vertex reconstruction compared to traditional maximum likelihood methods for 1-30 GeV neutrinos. The dynamic edge updates prove particularly effective for low-energy events where static geometric neighborhoods fail to capture essential light propagation patterns in Antarctic ice.

5.4. Kaggle competition

In early 2023, the IceCube collaboration hosted a data science competition on the competition platform Kaggle. The competition tasked participants with reconstructing neutrino directions from Cherenkov radiation signals detected by IceCube's 5,160 DOMs [37]. The dataset included 140 million simulated neutrino events spanning 100 GeV–100 PeV, with mixed event types (67% cascades, 33% tracks) and raw pulse data containing spatial coordinates, time, charge, and local coincidence status (see Section 3.2). Submissions were evaluated on the

mean opening angle between predicted and true neutrino directions. Over 800 teams submitted 11,206 solutions, with the top three demonstrating significant improvements over the baseline model's score provided by the organizers. The baseline was a simple DynEdge model trained on less than 8% of the competition data without further optimization [37] using a special von Mises-Fisher loss function [38].

The winning solution combined GNNs with Transformers to leverage geometric and sequential relationships in the data. Their architecture used modified EdgeConv layers that processed both absolute node features $(\mathbf{x}_j)$ and relative differences $(\mathbf{x}_j - \mathbf{x}_i)$ between neighboring pulses, followed by transformer blocks with multi-head attention [37]. Key innovations included static edge selection (k-NN computed only once per event) and a hybrid loss function combining angular distance with von Mises-Fisher loss. The model employed mixed-precision training and sequence bucketing to handle long pulse sequences efficiently. A stacking ensemble of six variants achieved a private leaderboard score of 0.960, which translates into the mean opening angle taken over the evaluation dataset being roughly 5.7% smaller than for predictions from the DynEdge baseline model.

The second-place solution focused on transformer architectures and enhanced them with physics-informed components. The team introduced Fourier encoding to embed continuous variables (e.g., time, charge) into high-dimensional representations using $10,000^{2j/d}$ frequencies, improving sensitivity to small input variations [37]. A custom Minkowski attention bias incorporated spacetime intervals ($ds^2 = c^2dt^2 - dx^2 - dy^2 - dz^2$) to weight pulse relationships. The architecture also used a DynEdge-inspired encoder and integrated ice properties (scattering/absorption lengths) as global context. Training utilized stochastic weight averaging and cosine annealing learning rate, with models scaling from 7.5M to 116M parameters. The solution achieved superior track reconstruction, reaching angular resolutions below 0.5° for events above 10 TeV [37].

The third-place team implemented a hybrid classification-regression approach using Transformers and gradient boosting. A transformer backbone processed pulse sequences, producing embeddings that were average-pooled and fed into dual prediction heads: a 128-bin classifier for zenith/azimuth angles and a regression head for 3D direction vectors [37]. Engineered features (e.g., first-pulse timing, DOM quantum efficiency) and transformer embeddings were combined using a BDT for final predictions. The model employed two-phase training, in which training on the classification task was followed by the regression task. This approach excelled at high-energy cascades, achieving resolutions below 5° for events above 10 TeV [37].

All solutions outperformed the standard DynEdge reconstruction model, with the top entries achieving resolutions comparable to IceCube's state-of-the-art reconstructions, especially for cascade events. Transformers demonstrated that they could boost the performance compared to just using GNNs, though their quadratic complexity limited applicability to very long

pulse sequences. The competition highlighted the potential of physics-informed architectures and large-scale training for neutrino astronomy, paving the way for real-time analysis pipelines in future IceCube upgrades.

The success of deep learning models in neutrino astronomy underscores the transformative potential of data-driven approaches in extracting meaningful insights from complex detector responses. It opens new possibilities for tackling challenges in particle physics, particularly challenges that come from the intractability of likelihood functions. New simulation-based methods that leverage deep learning to approximate the likelihood/ Bayesian posterior will be introduced in the next chapter.

6. Likelihood-free Methods

6.1. Statistical Background

Traditional statistical inference operates within two primary paradigms: frequentist hypothesis testing and Bayesian inference.

In the frequentist framework, which for example is being used in classical point source analyses in IceCube (see Section 3.6), hypothesis testing evaluates competing models through the profile-likelihood ratio test statistic:

$$\Lambda(\mathbf{x}) = \frac{\mathcal{L}(\boldsymbol{\theta}_0|\mathbf{x})}{\sup_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|\mathbf{x})}, \tag{6.1}$$

where $\boldsymbol{\theta}_0$ represents the null hypothesis. Under the Neyman-Pearson lemma, $\Lambda(\mathbf{x})$ is the most powerful test statistic for distinguishing $\boldsymbol{\theta}_0$ from alternatives [26]. Rejection regions are constructed based on the asymptotic distribution of $-2\ln\Lambda(\mathbf{x})$, which follows a χ^2 distribution [23].

In contrast, bayesian methods update prior beliefs $p(\boldsymbol{\theta})$ through Bayes' theorem [39]:

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathbf{x}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')\, d\boldsymbol{\theta}'}. \tag{6.2}$$

The posterior $p(\boldsymbol{\theta}|\mathbf{x})$ encodes parameter uncertainties, while the denominator (evidence) normalizes the distribution.

Both methods rely fundamentally on the likelihood function $\mathcal{L}(\boldsymbol{\theta}|\mathbf{x}) \propto p(\mathbf{x}|\boldsymbol{\theta})$, which quantifies the probability of observing data $\mathbf{x}$ given parameters $\boldsymbol{\theta}$. However, the likelihood function becomes intractable for many large-scale physics experiments like IceCube due to three fundamental challenges emerging from their scale and detection principles [26]. First, these detectors generate high-dimensional observations - from particle trajectories in tracking systems to energy depositions in calorimeters. A single interaction might produce $\mathcal{O}(10^3)$ correlated signals with nanosecond timing precision, creating event representations in $\mathbb{R}^N$ where $N \sim 10^3 - 10^5$. Second, many of the physical processes observed are of stochastic nature. Particle interactions involve latent variables like shower development, decay chains, and secondary particle production. These follow quantum mechanical probabilities and material-dependent processes (e.g., bremsstrahlung, hadronization). Third, the detector response adds additional complexity and stochastic nature. In the case of a large-scale neutrino telescope, this includes quantum efficiencies, angular acceptance variations across PMTs, and electronic noise thresholds. All these uncertainties create a probability space that includes billions of possible photon paths and sensor activation patterns, which in theory all have to

be taken into account to analytically construct the likelihood. This is not possible with current methods, making the likelihood intractable.

In order to overcome the problem of intractability, approaches to approximate the likelihoods have come up. They are mainly based on summary statistics that are used to compare observed and data coming from a sophisticated simulator. One prevalent technique is called Approximate Bayesian Computation (ABC), in which observed and simulated data are compared using some distance measure ρ which is based on the summary statistics [26]. In general, a simulator is run with parameters θ drawn from the prior. If the simulated data x_{sim} $p(\cdot|\theta)$ is sufficiently close to the data, i.e. $\rho(\mathbf{x_{sim}}, \mathbf{x_{obs}}) < \epsilon$, θ is kept as a sample of an approximate version of the posterior distribution. In theory, smaller ϵ makes ABC more exact. However, it also makes the acceptance probability smaller, which means that many more simulations are required to approximate the posterior. Because the observed data directly influences the rejection process, this whole algorithm has to be repeated for new observations, making ABC best-suited for cases with at most a few data points [26]. The frequentist pendants to this method are called (kernel) density estimation methods, which are used to approximate the distribution of different summary statistics from samples generated by the simulator [26]. As described in Section 3.6, it is also used in IceCube point source analyses. These methods are amortized, meaning that after an initial computational cost of the simulation and subsequent density estimation, new data points can be evaluated efficiently. Therefore, in contrast to ABC, this method works well for a large number of observations, which is often the case in particle physics experiments [26].

Both methods are fundamentally constrained by the curse of dimensionality [26]. As the complexity of observed data (e.g., high-dimensional sensor readouts in neutrino detectors) increases, the computational cost of simulations grows significantly, in the worst case even exponentially. To mitigate this problems, the data $\mathbf{x}$ is reduced to low-dimensional summary statistics. However, the accuracy of the inference depends critically on how well these summary statistics retain relevant information. Historically, the design of the summary statistics has relied heavily on domain expertise, where physicists manually engineer statistics based on their insights into the underlying processes [26]. While this approach has enabled progress in fields like neutrino astronomy and cosmology, it introduces subjectivity and risks discarding valuable information embedded in the full data structure.

With the rise of AI and deep learning, new methods have come up that tackle these problems. By training neural networks on simulator outputs, more or even full event information (e.g., raw PMT hit patterns through a model like DynEdge) could be used to approximate the likelihood while enabling amortized inference. Additionally, this inference can happen in ML speeds, which could help in giving real-time multi-messenger alerts. The next two sections describe the methods used to approximate the likelihood-to-evidence ratio as well as the posterior.

6.2. Neural Ratio Estimation (NRE)

Neural Ratio Estimation (NRE) addresses the intractability of explicit likelihood evaluation through a clever application of binary classification theory [26]. The method uses a dataset $\mathcal{D} = \{(\boldsymbol{\theta}_i, \mathbf{x}_i)\}$ where parameters $\boldsymbol{\theta}$ and observations $\mathbf{x}$ are drawn either from the joint distribution $p(\boldsymbol{\theta}, \mathbf{x})$ (positive class) or from the product of marginals $p(\boldsymbol{\theta})p(\mathbf{x})$ (negative class). Practically, this can be achieved by keeping data, generated by a forward simulator, as is for the positive class, while the negative class is constructed by randomly permuting parameters $\boldsymbol{\theta}$ across different observations $\mathbf{x}$ from the dataset. This shuffling process breaks the joint correlation between parameters and data while preserving their marginal distributions $p(\boldsymbol{\theta})$ and $p(\mathbf{x})$. A classifier $d(\mathbf{x}, \boldsymbol{\theta})$ is then trained to distinguish the positive and the negative class via binary cross-entropy loss [40]:

$$L = -\mathbb{E}_{p(\boldsymbol{\theta},\mathbf{x})}[\log d(\mathbf{x}, \boldsymbol{\theta})] - \mathbb{E}_{p(\boldsymbol{\theta})p(\mathbf{x})}[\log(1 - d(\mathbf{x}, \boldsymbol{\theta}))]. \tag{6.3}$$

As shown in [41], the optimal classifier d^* under this loss satisfies:

$$d^*(\mathbf{x}, \boldsymbol{\theta}) = \frac{p(\mathbf{x}, \boldsymbol{\theta})}{p(\mathbf{x}, \boldsymbol{\theta}) + p(\mathbf{x})p(\boldsymbol{\theta})}. \tag{6.4}$$

Rearranging yields the likelihood-to-evidence ratio $r(\mathbf{x}, \boldsymbol{\theta})$:

$$\frac{d^*(\mathbf{x}, \boldsymbol{\theta})}{1 - d^*(\mathbf{x}, \boldsymbol{\theta})} = \frac{p(\mathbf{x}, \boldsymbol{\theta})}{p(\mathbf{x})p(\boldsymbol{\theta})} = \frac{p(\mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x})} = r(\mathbf{x}, \boldsymbol{\theta}) \tag{6.5}$$

A classifier that is trained to distinguish the two classes therefore ouputs something which can be formulated to an estimation of the likelihood-to-evidence ratio. All that is needed is data from a sophisticated forward simulation, which usually exists for complex experiments in particle physics. Therefore, this method requires no further assumptions or approximations beyond those already used in the simulation. The better the simulation and the classifier, the better the approximation of the likelihood-to-evidence ratio. Additionally, the classifier can take any form, which means that newest methods in AI and deep learning can be leveraged to build a classifier that takes high-dimensional event data without the need to formulate low-dimensional summary statistics, making this method an appealing approach to counter problems arising in the traditional methods.

6.3. Neural Posterior Estimation (NPE)

Neural Posterior Estimation (NPE) directly approximates the posterior distribution $p(\boldsymbol{\theta}|\mathbf{x})$ via (conditional) neural density estimation. Unlike traditional sampling-based methods, NPE

amortizes inference by training a parametric model $q(\boldsymbol{\theta}|\mathbf{x})$ on simulations generated from the joint distribution $p(\boldsymbol{\theta}, \mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})$. This bypasses explicit likelihood evaluations, making it suitable for stochastic simulators with intractable likelihoods [39, 42].

The core objective is to train the model by minimizing the following loss function [43]

$$L_{\text{NPE}} = \mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta})}\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\boldsymbol{\theta})}\left[-\log q(\boldsymbol{\theta}|\mathbf{x})\right], \tag{6.6}$$

in which training pairs $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ are generated by sampling $\boldsymbol{\theta}_n$ from the prior $p(\boldsymbol{\theta})$ and running a simulator $\mathbf{x}_n \sim p(\mathbf{x}|\boldsymbol{\theta}_n)$.

NPE typically employs (conditional) Discrete Normalizing Flows (DNFs) to model $q(\boldsymbol{\theta}|\mathbf{x})$. A DNF transforms a simple base distribution $q_0(\boldsymbol{\theta}_0)$ (e.g., Gaussian) through a sequence of invertible, autoregressive layers $\{f_{i,\mathbf{x}}\}$ which are conditional on the observation (indicated by the subscript $\mathbf{x}$) [43]:

$$\boldsymbol{\phi}_{\mathbf{x}} = f_{N,\mathbf{x}} \circ \cdots \circ f_{1,\mathbf{x}}, \quad q(\boldsymbol{\theta}|\mathbf{x}) = q_0\big(\phi_{\mathbf{x}}^{-1}(\boldsymbol{\theta})\big) \left|\det \frac{\partial \phi_{\mathbf{x}}^{-1}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right|. \tag{6.7}$$

The Jacobian determinant in Equation 6.7 originates from the change of variables theorem, compensating for volume distortions caused by each invertible transformation $f_{i,\mathbf{x}}$ [43]. This term ensures conservation of probability mass, guaranteeing that $q(\boldsymbol{\theta}|\mathbf{x})$ remains a valid probability density satisfying $\int q(\boldsymbol{\theta}|\mathbf{x})\, d\boldsymbol{\theta} = 1$ [43]. For easy and quick evaluation that is computationally feasible, a normalizing flow should have a simple Jacobian determinant. Usually the Jacobian is kept as a triangular matrix, restricting the choice of transformations $f_{i,\mathbf{x}}$ that can be used.

While DNFs have proven effective for neural posterior estimation, they impose architectural constraints that limit flexibility. Continuous Normalizing Flows (CNFs) overcome these limitations by modeling the transformation from base to target distribution as a continuous process governed by an Ordinary Differential Equation (ODE).

In the CNF framework, we define a time-dependent vector field $\mathbf{v}$ that describes how samples move from the base distribution to the target distribution. This vector field is parametrized by a neural network and can be described by the ODE

$$\frac{d\boldsymbol{\theta}_t}{dt} = \mathbf{v}_{t,x}(\boldsymbol{\theta}_t), \tag{6.8}$$

where $\boldsymbol{\theta}_t$ represents the state of the sample at time $t \in [0, 1]$. The initial condition $\boldsymbol{\theta}_0$ is drawn from a simple base distribution $q_0(\boldsymbol{\theta})$ (typically a standard Gaussian), and the solution at $t = 1$ yields a sample from the target distribution $q(\boldsymbol{\theta}|\mathbf{x})$. To move the samples from the

base distribution to the target distribution, the vector field has to be integrated over time t, typically by using a standard ODE solver [43].

To explicitly evaluate the density $q(\boldsymbol{\theta}|\mathbf{x})$, we must account for how probability mass transforms along the flow. This is governed by the continuity equation [43]

$$\frac{\partial q_{t,\mathbf{x}}(\boldsymbol{\theta}_t)}{\partial t} + \nabla \cdot \big(q_{t,\mathbf{x}}(\boldsymbol{\theta}_t) \cdot \mathbf{v}_{t,\mathbf{x}}(\boldsymbol{\theta}_t)\big) = 0, \tag{6.9}$$

which ensures probability mass conservation. Solving this equation yields the target distribution [43]

$$q(\boldsymbol{\theta}|\mathbf{x}) = q_1(\boldsymbol{\theta}_1|\mathbf{x}) = q_0(\boldsymbol{\theta}_0) \cdot \exp\left(-\int_0^1 \nabla \cdot \mathbf{v}_{t,\mathbf{x}}(\boldsymbol{\theta}_t)\, dt\right). \tag{6.10}$$

In principle, training CNFs by directly maximizing the likelihood similar to the DNF is possible. However, it is computationally expensive due to the need to integrate the divergence term in every training step [43]. Flow Matching (FM) offers an elegant alternative by directly regressing the vector field $\mathbf{v}$ onto a target vector field $\mathbf{u}$, bypassing explicit likelihood calculations. The idea of FM is to construct a time-dependent probability path $p_t(\boldsymbol{\theta}|\boldsymbol{\theta}_1)$ that smoothly interpolates between the base distribution at $t = 0$ and a narrow distribution centered at the target parameter $\boldsymbol{\theta}_1$ at $t = 1$ [43]. The probability paths and therefore the target vector field can be chosen sample-conditionally, which means that the target $\mathbf{u}_t$ for a given training sample $\mathbf{x}$ depends on the corresponding truth value $\boldsymbol{\theta}_1$. For this case, it was shown in [44] that there are some simple choices for the target vector field $\mathbf{u}$ that make the regression equivalent to likelihood maximization.

A common choice is the vector field

$$\mathbf{u}_t(\boldsymbol{\theta}|\boldsymbol{\theta}_1) = \frac{\boldsymbol{\theta}_1 - (1 - \sigma_{\text{min}})\boldsymbol{\theta}}{1 - (1 - \sigma_{\text{min}})t} \tag{6.11}$$

that yields Gaussian probability paths [43]

$$p_t(\boldsymbol{\theta}|\boldsymbol{\theta}_1) = \mathcal{N}\big(\boldsymbol{\theta}_1, t, (1 - (1 - \sigma_{\text{min}})t)^2 \cdot \mathbf{I}_d\big). \tag{6.12}$$

where σ_{min} is the standard deviation of the Gaussian distribution around our target parameters $\boldsymbol{\theta}_1$. Training on many $\boldsymbol{\theta}_1$ and their corresponding target vector fields $\mathbf{u}_t(\boldsymbol{\theta}|\boldsymbol{\theta}_1)$ lets our vector field $\mathbf{v}_{t,\mathbf{x}}(\boldsymbol{\theta})$ transform the base distribution into the target distribution.

The FM loss function minimizes the expected L^2 distance between the predicted vector field $\mathbf{v}$ and the target velocity $\mathbf{u}$ [43]:

$$L_{\text{FM}} = \mathbb{E}_{t\sim p(t)}\mathbb{E}_{\boldsymbol{\theta}_1\sim p(\boldsymbol{\theta})}\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\boldsymbol{\theta}_1)}\mathbb{E}_{\boldsymbol{\theta}_t\sim p_t(\boldsymbol{\theta}_t|\boldsymbol{\theta}_1)}\left[\left\|\mathbf{v}_{t,\mathbf{x}}(\boldsymbol{\theta}_t) - \mathbf{u}_t(\boldsymbol{\theta}_t|\boldsymbol{\theta}_1)\right\|^2\right], \tag{6.13}$$

where the expectation is taken over the time t, the target parameters $\boldsymbol{\theta}_1$ sampled from the prior $p(\boldsymbol{\theta})$, the observation $\mathbf{x}$ coming from a simulator and the intermediate states $\boldsymbol{\theta}_t$ sampled from the probability path $p_t(\boldsymbol{\theta}_t|\boldsymbol{\theta}_1)$.

In practice, the training procedure for each sample proceeds as follows [43]:

1. Sample a parameter $\boldsymbol{\theta}_1 \sim p(\boldsymbol{\theta})$ and generate the corresponding observation $\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta}_1)$ (or pick a $(\boldsymbol{\theta}_1, \mathbf{x})$ pair from a pre-generated training dataset)
2. Sample a time point $t \in [0, 1]$ (e.g. from uniform distribution)
3. Sample an intermediate state $\boldsymbol{\theta}_t \sim p_t(\boldsymbol{\theta}|\boldsymbol{\theta}_1)$
4. Compute the target vector field $\mathbf{u}_t(\boldsymbol{\theta}_t|\boldsymbol{\theta}_1)$
5. Evaluate the predicted vector field $\mathbf{v}_{t,\mathbf{x}}(\boldsymbol{\theta}_t)$
6. Update the model parameters to minimize the squared error between $\mathbf{v}$ and $\mathbf{u}$

After training, the posterior for a given $\boldsymbol{\theta}$ can be evaluated by conditioning the vector field $\mathbf{v}$ on the observable $\mathbf{x}$ and solving equation 6.10.

The primary advantage of CNFs is their architectural flexibility. By eliminating the need for invertible layers with triangular Jacobians, CNFs enable unrestricted neural network designs that can better capture complex posterior distributions. This flexibility translates to improved scalability, as CNFs avoid layer-wise memory bottlenecks and support deeper architectures, resulting in approximately 3× faster training than comparable DNFs [43]. Additionally, CNFs exhibit a valuable mass-covering property that conservatively covers regions of non-zero posterior density [43].

Despite their advantages, CNFs come with notable limitations that create trade-offs in their practical application. The most significant drawback is the substantially increased inference cost compared to DNFs. While CNFs avoid explicit Jacobian determinant calculations during training, density evaluation still requires integrating the divergence of the vector field $\nabla\cdot\mathbf{v}$ over time t. This integration adds computational overhead that cannot be avoided when precise density values are needed and might be a limiting factor when inference speed is essential [43].

7. Pre-training Dynedge Backbone

The general idea that will be discussed in this thesis is to use a deep learning pipeline that goes directly from raw event data to a spatial likelihood representation for each event, skipping the steps of reconstructing point estimates, approximating uncertainties and fitting KDEs. This chapter will describe the steps that have been performed in order to pre-train a DynEdge backbone model, which will be the first component of the pipeline the observable passes through:

$$\text{Observable} \rightarrow \text{Backbone} \rightarrow \text{NRE/NPE} \rightarrow \text{Ratio/Posterior}.$$

The backbone acts as a feature extractor that gives a latent representation of the original event level pulse data. This latent representation should contain all the important information for the task at hand and will be the input into the NRE and NPE networks. For this application, the DynEdge architecture presents an advantageous solution due to its ability to process input sequences of varying lengths while producing a latent representation with a fixed dimensionality. The next sections explore the data the models are trained on as well as the structure and the pre-training of the DynEdge backbone.

7.1. Training data

The dataset used in this thesis contains only simulated track events that SplineMPE reconstructed within the northern sky. In total, the dataset contains 6,341,248 final level events, meaning the data went through all the IceCube processing stages. One of the biggest challenges this data (neutrino telescope data in general) represents is the fact that each event can contain a vastly different amount of pulses. They can vary between dozens and tens of thousands of pulses, as Figure 6 shows. Every pulse contains the euclidean position of the DOM that captured the pulse, the charge it induced as well as the time the pulse was measured.

In this thesis, a cut is being made at a pulse count of 1024, only using events with fewer pulses. This way, more than 98% of the events in the dataset are being utilized while inherently limiting the memory consumption in our model training. Although the data that is being discarded represents only a small fraction of the total dataset, its removal can still introduce a bias to the remaining data. Specifically, events with a high number of pulses are being excluded. These events typically correspond to higher-energy neutrinos, which produce more photons and subsequently result in more pulses in the recorded data. Importantly, such events also tend to offer the best directional reconstruction because the increased number of pulses provides richer information about the track's geometry and direction. This makes them highly valuable for analyses requiring precise localization.

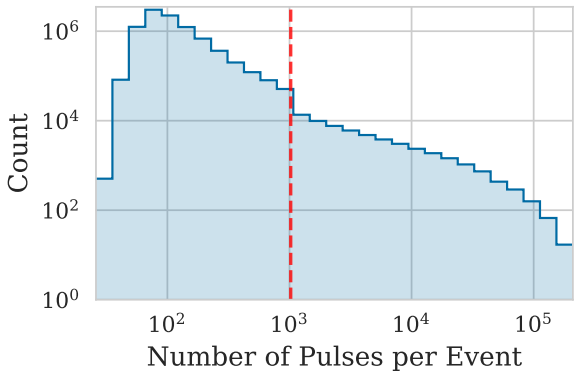

Figure 6 Histogram of the pulse counts of each of the 6,341,248 events in the training dataset. The red dashed line indicates the cut at 1,024 pulses that is being made.

However, excluding these events is necessary due to computational constraints. The large number of pulses significantly increases the sequence length of the input data, which poses challenges for the DynEdge model architecture. GNNs are well-suited for processing variable-length and structured data, but their computational complexity scales with the size of the input graph. For events with exceptionally high pulse counts, this scaling can lead to memory overflow or prohibitively long inference times. By applying a pulse cut at 1024 pulses, we ensure that the model remains computationally efficient and can process events within reasonable timeframes.

Although this approach limits the inclusion of high-pulse events with superior directional information, it reflects an active trade-off to ensure computational feasibility. Additionally, excluding these events makes training more challenging by forcing the model to learn from less informative data, potentially improving robustness. Ongoing research into efficient architectures and data representation aims to make processing high-pulse events more feasible without compromising performance or efficiency.

The remaining dataset contains neutrino events with an uniform true azimuth distribution. The true zenith distribution is mostly limited to the northern sky but has some outliers due to bad reconstruction via SplineMPE. The true neutrino energy is distributed within 100 GeV and 500 PeV with a peak around 1 TeV, as the histograms in Figure 7 show. During training, no weights have been applied.

7.2. Training the Backbone model

In order to let DynEdge extract the features that contain as much information about the direction of the original neutrino as possible, the model is being trained as a regression model that predicts azimuth and zenith. This is one of the main purposes that DynEdge was original developed for. In this case, the DynEdge model is instantiated with its default parameters as described in [32]. The corresponding architecture can be seen in Figure 5 in Section 5.3.

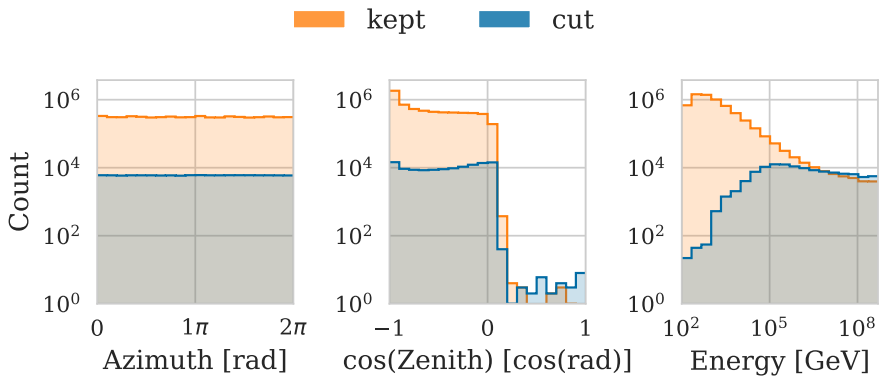

Figure 7 Histograms of true azimuth, cosine of the zenith and the neutrino energy of each of the 6,341,248 events in the full dataset. The final training dataset containing 6,221,963 events is colored in blue.

Most notably, the EdgeConv blocks work on the 8 nearest neighbors and the global pooling scheme includes minimum, maximum, mean and summation. In total, this architecture contains 1.4 million trainable parameters.

The regression task is trained using the von Mises-Fisher loss that was also used for the baseline model in the Kaggle competition (see Section 5.4). The loss funtion utilizes the von Mises-Fisher distribution, which is considered the directional equivalent to a Gaussian distribution, mapping the Gaussian to a spherical embedding. It is scaled by κ, which is a measure of how broad or narrow the distribution is. For low κ approaching zero, the von Mises-Fisher distribution will be distributed evenly on the directional sphere. For $\kappa = \inf$, the distribution will be a point distribution at the target direction. The idea is during training to not only predict the direction but also κ as a type of an uncertainty estimate. The predicted κ will then define a von Mises-Fisher distribution around the target direction. The loss function is then the negative logarithm of this distribution evaluated at the predicted direction [38]. Including the uncertainty into the prediction proved to be useful, especially when the overall goal is to approximate something like the directional likelihood, which is very much dependent on how confident we are in reconstructing the direction for a given event.

Before training, the training dataset is split into a dataset which actually trains the model and a dataset which validates the training after each epoch by calculating the loss for this unseen dataset. The chosen ratio for the split is 9:1. The model is being trained using a batch size of 800. The batch size indicates the number of events in the subset of data that is being used in one training step. Each step updates the parameters of the model according to the gradients of the combined loss of the batch. The learning rate, which scales the gradients, is scheduled using PyTorch's *ReduceLROnPlateau*[1] learning rate scheduler. It monitors the loss of the validation dataset after each epoch and reduces the learning rate by a factor of 0.3 if the validation loss has not decreased within 3 epochs. The starting learning rate is set to 0.001 and the optimizer of choice is *Adam* with standard parameters. To prevent overfitting,

[1] *ReduceLROnPlateau* documentation can be found **here.**

an early stopping mechanism is implemented, which stops the training if the validation loss has not decreased in 7 epochs.

7.3. Performance of the pre-trained model

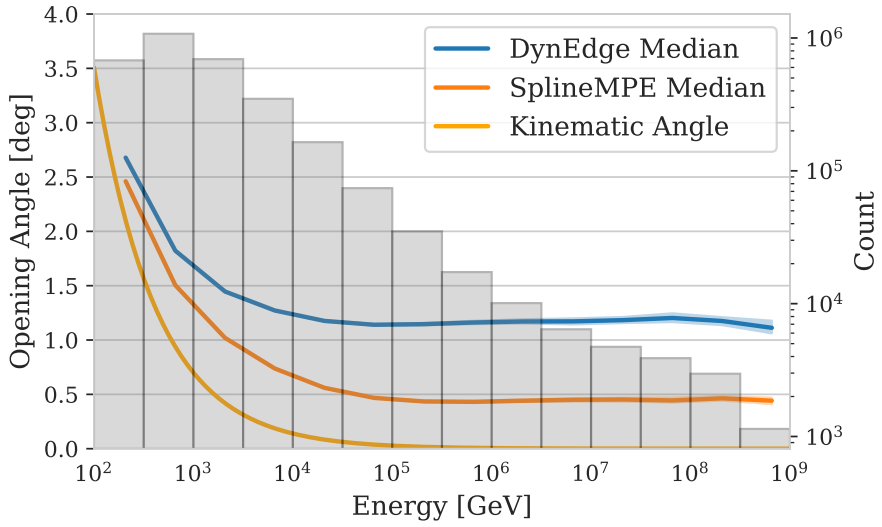

Figure 8 Performance of the DynEdge backbone model and the traditional SplineMPE method in terms of opening angle between point prediction and truth. In the background, the grey histogram represents the events that are being evaluated. These events come from a different dataset that the model has not been trained on. The uncertainty bands come from bootstrapping and represent the 95% bands. Plotted in orange is the mean kinematic angle between neutrino and muon, which represents a lower bound for the resolution of the reconstruction.

In Figure 8, the performance of the model is being shown. The $\sim$ 1 million evaluation events are not part of the training dataset and are therefore unseen by the model. However, they stem from a dataset which has similar azimuth, zenith and energy distributions as the training dataset as well as a pulse cut at 1,024 pulses. The energy histogram of the evaluation events is shown as the gray background in the figure. The evaluation metric is the opening angle between the point prediction and truth. For each energy bin, the DynEdge median as well as the SplineMPE median are shown.

The uncertainty bands come from a method called bootstrapping [45]. In this method, for each bin, 1,000 bootstrap samples are generated by repeatedly sampling with replacement from the original data within that bin. The evaluation metrics are then calculated for each bootstrap sample, resulting in a distribution of each metric. From these distributions, the 2.5th and 97.5th percentiles are determined, representing the boundaries of the 95% confidence interval for each bin's metrics, and therefore providing a measure of the associated uncertainty.

For low neutrino energy, the DynEdge model has more problems in reconstructing the direction than for higher energies. This is expected due to the kinematic angle between parent neutrino and resulting muon. For lower energies, the kinematic angle is quite large, adding

randomness to the direction of the muon which can not be overcome. For energies above 10^5 GeV, this angle is close to zero and does not have a big effect on the reconstruction capabilities anymore. From here on, the resolution is limited by the method itself only. The median opening angle sits slightly above 1 degree and stays nearly constant for the whole energy range above 10^5 GeV. However, the median opening angle for the SplineMPE reconstruction method is significantly smaller for the whole regime.

The DynEdge performance is comparable to other studies that have been done with similar architectures (e.g. in [37]). While this proves that the model is capable of extracting information about the direction of the neutrino, it should be noted that for these type of events and this energy range, it is not the best deep learning based method for directional reconstruction. The Kaggle solutions proved to beat this performance [37]. However, the DynEdge model is still used as the backbone for the likelihood-free methods, because it is a much lighter model in terms of computational cost. This allows for faster training and inference, making it a good backbone for testing the methods and proving their concept.

8. NRE Model Development

This chapter focuses on the development of models for NRE as described in chapter 6. The fundamental approach involves training a classifier to differentiate between data from correlated observation-direction pairs $p(\mathbf{x}, \theta)$ and uncorrelated, scrambled pairs $p(\mathbf{x})p(\theta)$. An optimal classifier subsequently yields a formulation of the likelihood-to-evidence ratio, quantifying the probability that a direction θ and an observation $\mathbf{x}$ correspond to each other.

8.1. Basic NRE

8.1.1. Architecture and Training

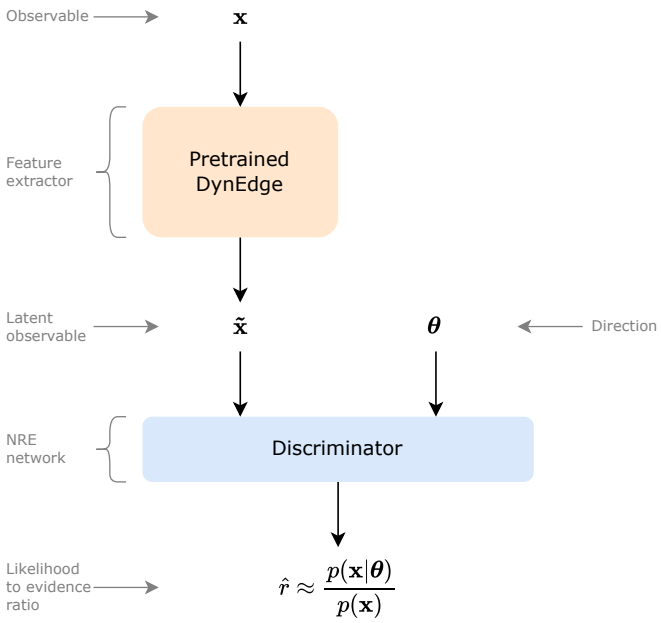

Figure 9 The base architecture for the NRE method. The observed pulsemap will go through DynEdge to generate a latent representation of the observable. This is then concatenated with the direction and put through the discriminator, which then gives yields the likelihood to evidence ratio.

The architecture of the basic NRE model described in this thesis can be seen in Figure 9. The observable $\mathbf{x}$, which in this case is the pulsemap of a neutrino event, will be put through the backbone of the pre-trained DynEdge model from Chapter 7. All parameters of the backbone are frozen during training, meaning the loss function will not change the backbone model. The prediction head of the angular regression task of the DynEdge model is discarded. The DynEdge backbone used in this architecture therefore outputs a 128 dimensional latent representation $\tilde{\mathbf{x}}$ of our observable from which the backbone model is able to regress the direction and the uncertainty.

At this stage, the latent representation is concatenated with the direction of our interest. The

direction is also represented in euclidean coordinates to account for the nature of the sphere. In the azimuthal direction, 0 and 2π represent the same azimuth. However, a standard MLP without special embeddings does not inherently account for this equivalence and may incorrectly interpret these directions as being significantly different. Euclidean coordinates, in which each direction will be represented as a 3 dimensional point on the unit sphere, solve this problem. The concatenated $(\tilde{\mathbf{x}}, \boldsymbol{\theta})$ pairs are now put through the Discriminator network. The Discriminator consists of a simple MLP with 10 hidden layers, leaky ReLU activation functions and roughly 480,000 trainable parameters. It predicts a one dimensional output. During training, the output of the last linear layer of the Discriminator will be pushed through a Sigmoid function (see 4.1) to predict the class of the observable direction pair $(\mathbf{x}, \boldsymbol{\theta})$, 0 for the uncorrelated pairs and 1 for the correlated pairs. For the final likelihood to evidence ratio inference, we can skip the Sigmoid function and directly apply the exponential function on the output of the last linear layer d of the MLP:

$$\frac{p(\mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x})} = \frac{d^*}{1-d^*} = \frac{\frac{1}{1+e^{-d}}}{1-\frac{1}{1+e^{-d}}} = e^d, \tag{8.1}$$

in which d^* represents the output after the Sigmoid function for an optimal classifier (see equation 6.5).

For training, the dataset from Chapter 7 is utilized. This dataset is duplicated, with one copy labeled as class 1, where each event's true direction is used as θ. The other copy is labeled as class 0, where the directions are randomly permuted across all events in the dataset. At the start of each training epoch, a new permutation is performed to introduce variability and prevent overfitting to specific direction assignments. As before, the data is split into a training and validation set with a split ratio of 9:1. The batch size is set to 500. The learning rate scheduler *ReduceLROnPlateau* is being utilized, in this case reducing the learning rate by a factor of 0.1 if the validation loss has not decreased within 4 epochs. The starting learning rate is set to 0.001 and the optimizer is *Adam* with standard parameters. An early stopping mechanism is implemented with a patience of 10 epochs. The loss function in use is the binary cross entropy (see equation 6.3), as our general task is a binary classification.

8.1.2. Classification Performance

To assess the capabilities of this model, both the performance of the classification on which the model is being trained as well as the performance of the actual likelihood to evidence ratio approximation can be analyzed. For this evaluation of the classification performance, 1 million events from the same unseen dataset from 7.3 are utilized.

First, the raw output distribution of the discriminator is examined. The left plot in Figure 10 shows histograms of the model's Sigmoid outputs for both true classes 0 and 1. This visualization reveals the model's ability to separate the two classes. Ideally, predictions for true positive examples should cluster near 1, while predictions for true negative examples should

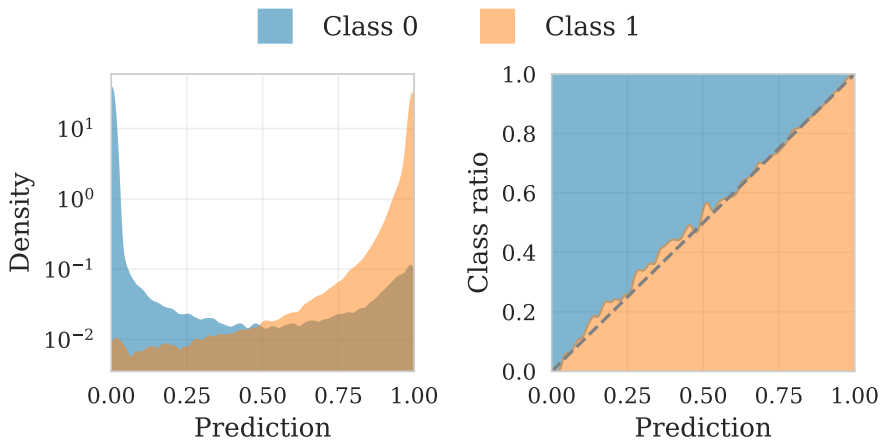

Figure 10 Left: Output distributions of the basic NRE classifier for both classes. **Right:** Calibration curve which indicates how well the classifier output resembles an actual probability.

cluster near 0. The degree of overlap between these distributions provides an immediate visual indication of classification performance — less overlap suggests better discrimination. In this case, the distributions look as expected. The model demonstrates strong class separation, as both classes cluster at their respective end of the prediction spectrum. The overlaps are small, indicating that our model trained well on this task of discriminating between correlated and uncorrelated observable-direction pairs.

The right plot in Figure 10 is the so called calibration curve or reliability diagram [46]. It consists of the ratio of true positive predictions to all predictions across different prediction scores. For this classifier, the ratio is well alligned with the ideal diagonal line, indicating that our classifier performs equally well for both classes across the entire prediction range. The absence of systematic deviations from the diagonal suggests that our model maintains consistent accuracy across different prediction regimes. It also means that the classifier's output can be interpreted as the probability with which the given observable-direction comes from the joint distribution $p(\mathbf{x}, \theta)$. This is a notable strength, as it indicates that the classifier can be trusted equally well whether it's making high- or low-confidence predictions. It shows that the model is well balanced and does not favor one class over the other in any of the prediction regimes.

For a more comprehensive metric to capture the models classification performance we can turn to the so called Receiver Operating Characteristic (ROC) curve. The y-axis depicts the True Positive Rate (TPR)

$$\text{TPR} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}, \tag{8.2}$$

which describes the number of true predictions for samples with the underlying true class 1 divided by the number of all samples with underlying true class 1. The x-axis contains the

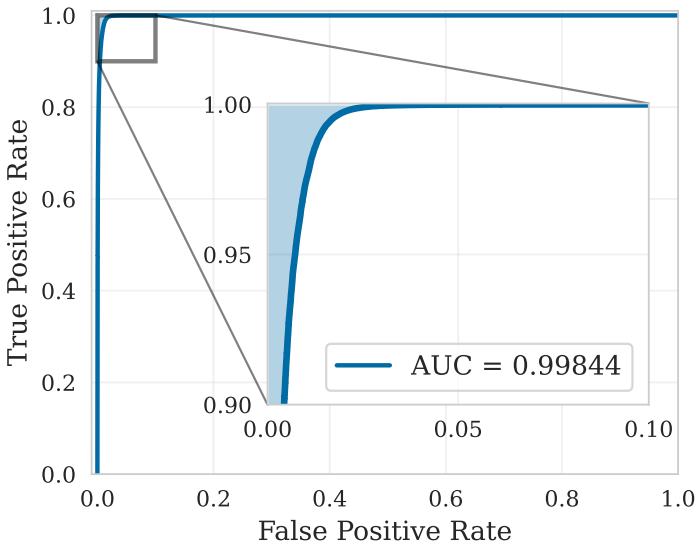

Figure 11 ROC curve for the basic NRE classifier. The curve plots the TPR against the FPR at various threshold settings.

corresponding False Positive Rate (FPR)

$$\text{FPR} = \frac{\text{False Positive}}{\text{False Positive} + \text{True Negative}}, \tag{8.3}$$

which gives the ratio of the number of false predictions for samples with underlying true class 0 and the number of all samples with underlying true class 0. The TPR and FPR pairs are calculated for different decision thresholds that divide the output of the classifier in the predicted classes 0 and 1. Usually, setting the best threshold for a classifier consists of a trade-off between a high TPR and a low FPR. The ROC curve visualizes this trade-off and therefore depicts how confidently a classifier predicts the class and how well it can actually do it. To quantify this, the Area Under Curve (AUC) metric is often times used, which, as the name suggests, is the value of the integral of the ROC curve. An AUC of 1 would indicate a perfect classifier that outputs for all samples with underlying true class 1 the highest value in the prediction range.

Figure 11 shows the ROC curve for our this NRE model. The curve's proximity to the top-left corner of the plot confirms the strong separation capability already observed in the prediction distributions. Our model achieves an AUC of 0.99844, which is almost a perfect AUC and which confirms the good discriminative power of our model.

8.1.3. Likelihood Approximation Performance

After establishing the excellent classification capabilities of our NRE model, we now evaluate its performance as a likelihood estimator for directional reconstruction. Therefore, we want to measure its ability to accurately recover the true neutrino direction as well as its ability to estimate the confidence of its prediction. This part of the evaluation is performed for 100,000 random events from the evaluation dataset described in Section 7.3.

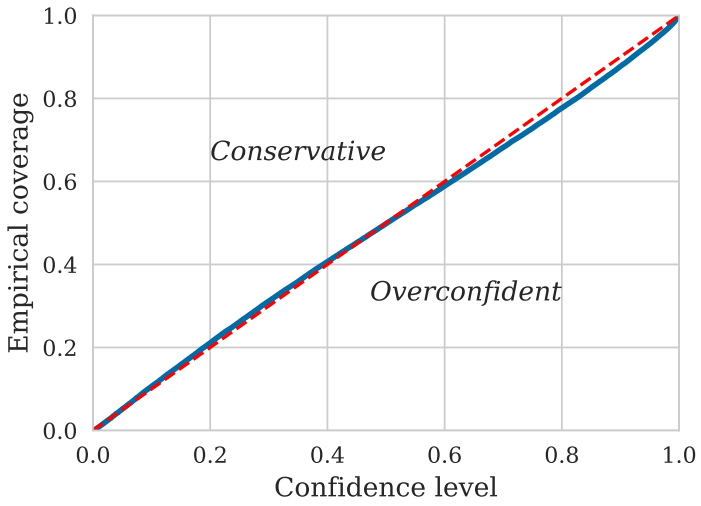

Figure 12 Coverage plot for the basic NRE model. The coverage is evaluated based on 100,000 evaluation events. Ideal coverage, i.e. empirical coverage which exactly matches the coverage given by the χ^2 distribution with 2 degrees of freedom, is given by the dashed line.

A crucial aspect of evaluating our likelihood approximation is its coverage property — whether confidence regions contain the true neutrino direction at the expected rate. For a well-calibrated model, a 90% confidence region should contain the true direction in 90% of cases. We assess coverage using Wilks' theorem, which relates likelihood ratios to the χ^2 distribution (see Section 3.6). First, we identify the maximum likelihood direction by evaluating our model over a directional grid. Then, we can compute the test statistic $-2\Delta \log \mathcal{L}$ between this maximum and the true direction. This is repeated for the whole evaluation dataset, which gives us a distribution of test statistics that in theory should match the χ^2 distribution with two degrees of freedom. To check this property, the cumulative distribution function of our empirical test statistics is plotted against the theoretical χ^2 cumulative distributions. This can be seen in Figure 12.

This analysis reveals almost perfect coverage properties, with the empirical distribution following almost exactly the diagonal line that indicates ideal coverage. This agreement indicates that our NRE model indeed produces properly calibrated likelihoods across all confidence levels. Figure 14 shows these likelihood contours for three example events with their topologies illustrated in Figure 13. The model actually finds a direction close to the truth and draws plausible likelihood contours. However, for these three examples we can already see that the maximum likelihood estimate from our NRE model is much further away from the truth than the Spline MPE directional point estimate.

To fully assess the resolution, we compute the opening angle between the maximum likelihood direction predicted by our model and the true neutrino direction. Because the backbone is pre-trained and does all the relevant feature extraction, we expect the performance to be roughly as good as the pre-trained model.

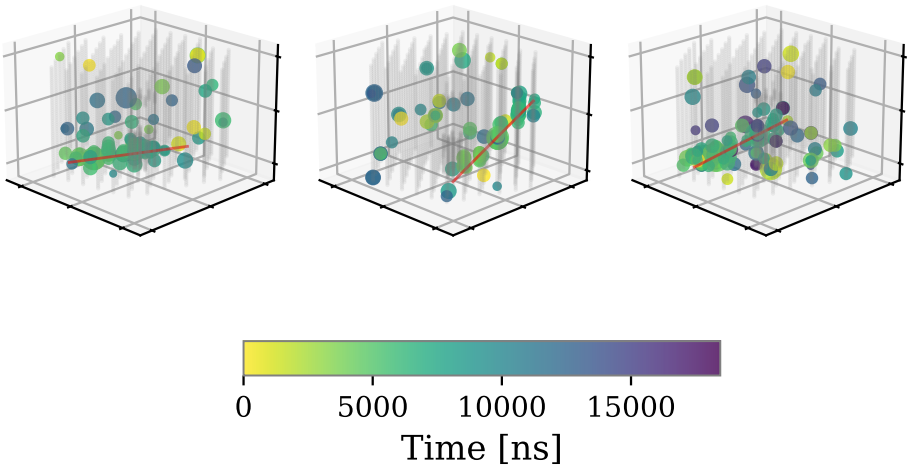

Figure 13 Topologies of the three simulated example events. The spheres represent measured charges at individual DOMs, with size proportional to charge magnitude and color indicating detection time. The true direction of the parent neutrinos are indicated by the red line.

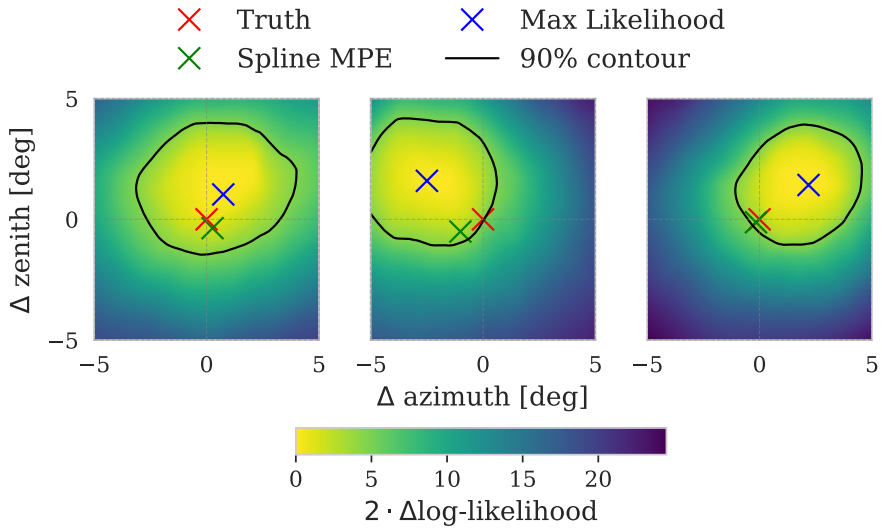

Figure 14 A grid scan of the likelihood for three example events from the evaluation dataset. Here, two times the delta log-likelihood is taken between the maximum likelihood and the likelihood at that given direction. The 90% contour line shows the level where the cumulative χ^2 distribution with 2 degrees of freedom reaches 90%. The red, green and blue marker show the true direction, the reconstructed direction from SplineMPE and the maximum likelihood direction from this model respectively.

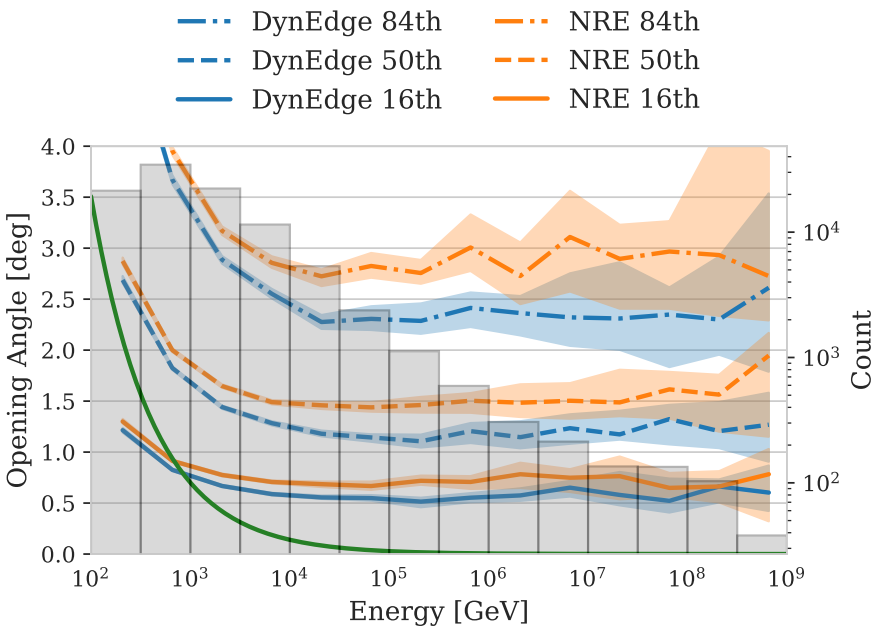

Figure 15 Performance of the NRE model in terms of opening angle between point prediction and truth. This plot shows the 16th, 50th and 84th percentile for each energy bin. In the background, the grey histogram represents the events that are being evaluated. These events come from a different dataset that the model has not been trained on. The uncertainty bands come from bootstrapping and represent the 95% bands.

Figure 15 shows that the resolution of our model is worse than the resolution of the pre-trained model for all energies and all three percentiles. This might be due to the fact that the training objective differs between the pre-trained model and our NRE approach. The pre-trained model was optimized directly for point estimation of direction, whereas our NRE model is trained to approximate likelihoods. This difference in optimization criteria could explain the slight degradation in point prediction performance, which would then come as a trade-off for accurate approximation of the likelihoods.

While the degradation may appear relatively minor, it was not anticipated. It also further highlights the broader challenge that deep learning-based directional reconstruction methods often underperform compared to traditional methods for high-energy tracks. Therefore, various strategies have been tried to mitigate this problem, including changing the discriminator architecture, unfreezing the backbone parameters or parts of it during training, using different learning rates and learning rate schedulers as well as training the whole model including the backbone from scratch. While some strategies may have helped with faster convergence or slight improvements of the angular resolution, the angular resolution of the pre-trained DynEdge model was never reached. Because of that, a few new approaches apart from hyperparameter optimization have been tested, which will be described in the next sections.

8.2. von Mises-Fisher Sampling

As demonstrated in Section 8.1.2, the classifier performs well on the classification task it was trained on. However, one potential concern is that the training task may have been too simplistic, allowing the classifier to achieve high performance without needing to precisely distinguish between the true and randomly permuted directions. This could lead to a loss of precision during training. To address this, the approach is adjusted by obtaining class 0 directions not by permutation, but instead by sampling from a von Mises-Fisher distribution (see Section 7.2) centered around the true direction. This increases the density of samples near the truth, effectively making the task more challenging and encouraging the classifier to achieve finer discrimination between true and sampled directions. The parameter κ can then be used to control how easy or how difficult the classification task should be, as the density distributions show in Figure 16.

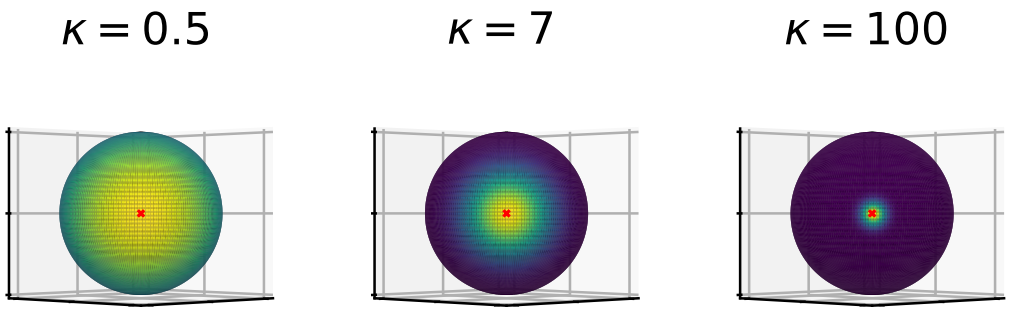

Figure 16 3-dimensional von Mises-Fisher distributions for different κ. The higher the κ, the more concentrated the distribution around the central direction of the distribution μ.

8.2.1. Architecture and Training

The architecture is the same as depicted in Figure 9. The observed pulsemap will go through the same pre-trained DynEdge model, before it gets concatenated with the direction and fed into the discriminator network. The only difference is the direction for the uncorrelated class 0, which does not come from permutations but from a von Mises-Fisher distribution around the truth.

The training is done analogous to the training described in Section 8.1.1. The von Mises-Fisher sampling is only applied to the training dataset, for validation of class 0 we keep the random shuffling of the directons. For class 0 samples, the loss weights during training are defined as the ratio between the probability density of the sample's direction under the von Mises-Fisher distribution and the corresponding probability density in the training dataset distribution. This weighting ensures that the original directional distribution is preserved.

For this model, an additional parameter, κ, must be configured for training. The best results were obtained by gradually increasing κ over the course of training, with its value scheduled to

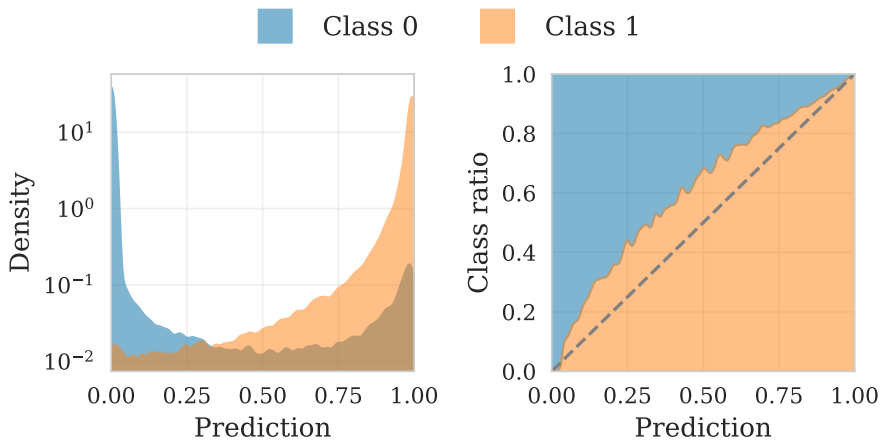

Figure 17 Left: The Sigmoid output distributions of the classifier trained using von Mises-Fisher sampling. As expected, the distributions are highest at their respective target value. **Right:** The calibration curve for this model. The diagonal line indicates a perfectly weighted classifier. In this case, there is a slight imbalance which indicates that the classifier has more difficulty in classifying samples from the uncorrelated class 0.

grow with each epoch. As a result, the von Mises-Fisher distribution used to sample directions for the uncorrelated class becomes progressively narrower, making the classification task increasingly challenging as training progresses. The κ is initially set to 0.5 (see left plot in Figure 16) and increases by 0.1 every epoch.

8.2.2. Classification Performance

Like before, we want to measure the capabilities of our model by both assessing the classification performance and the likelihood approximation performance. The same unseen dataset as before is being used and the class 0 values are created with random shuffling like for validation.

First, we look at the raw classification output again. It is shown in the left plot in Figure 17. As before, the visualization shows that the model is able to effectively distinguish between the classes. However, when turning to the calibration curve on the right, a slight imbalance is visible, which shows that the classifier has more difficulty in classifying samples from class 1 than from class 0. This might be explained due to the different nature of the class 0 samples during training and evaluation. While during training, the observables are matched with directions rather close to the true direction, during validation and also this evaluation, the matching is conducted the same way as before by shuffling all the directions in the training dataset. The model therefore has an easier task in classifying shuffled class 0 samples during evaluation, while for class 1 samples the uncertainty stays the same as during training.

Figure 18 shows the ROC curve for this model. The model achieves an AUC of 0.9972, which is slighly lower than the AUC of the basic DynEdge NRE model, but still very close to 1, indicating almost perfect classification. The slightly lower AUC might also be explained by the difference of the class 0 dataset during training and evaluation.

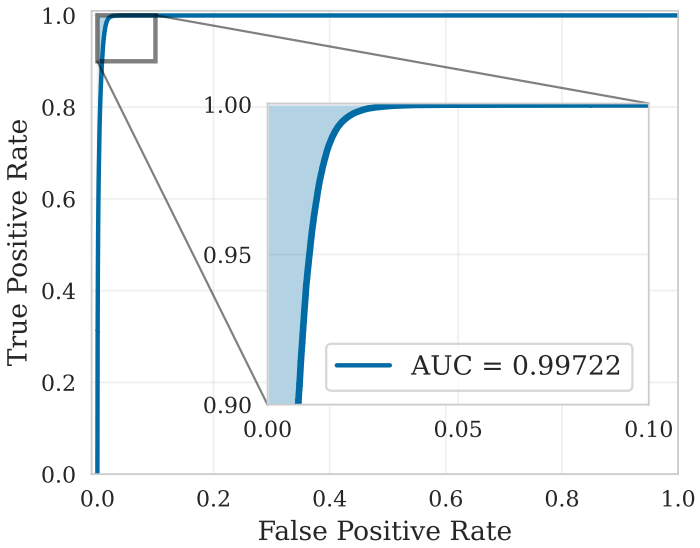

Figure 18 ROC curve for the vMF classifier. The curve shows the TPR against the FPR at various threshold settings.

8.2.3. Likelihood Approximation Performance

Figure 19 illustrates the coverage properties of the von Mises-Fisher NRE model. The coverage aligns closely with the ideal behavior at lower confidence levels but exhibits slight overconfidence at higher confidence levels, where the true direction is contained less frequently than the indicated confidence level suggests. Nonetheless, the overall coverage demonstrates good agreement with Wilks' theorem, supporting the interpretation of the model output as an estimate of the likelihood-to-evidence ratio.

The contours for the same three example events as before can be seen in Figure 20. The truth is contained within the 90% contour for all three events. When comparing with the contours of the basic NRE model in Figure 14, the contour sizes seem to be comparable. For actual comparison of the resolution, the opening angle between maximum likelihood direction and true direction should be considered.

The opening angle is presented in Figure 21. While the overall performance remains worse than the pre-trained DynEdge model, the use of von Mises-Fisher sampling has reduced the disparity. While this represents an improvement, the achieved resolution still falls short of the initially expected level, which was to match the resolution of the pre-trained model. Furthermore, the pure classification performance appears to be slightly worse. Although this only has a small impact on the model, it could become significant in different settings. If this issue becomes too pronounced, one potential mitigation strategy could be to combine von Mises-Fisher sampling with the original direction-shuffling approach.

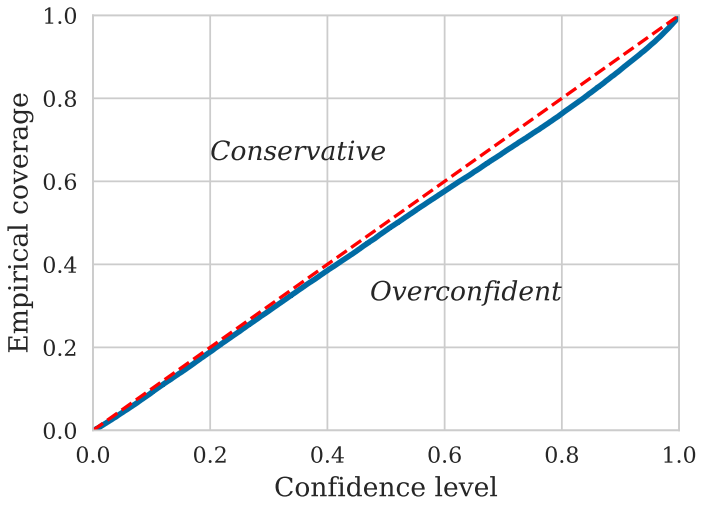

Figure 19 Coverage plot for the vMF model. The coverage is evaluated based on 100,000 evaluation events. Ideal coverage, corresponding to the χ^2 distribution with 2 degrees of freedom, is given by the dashed line.

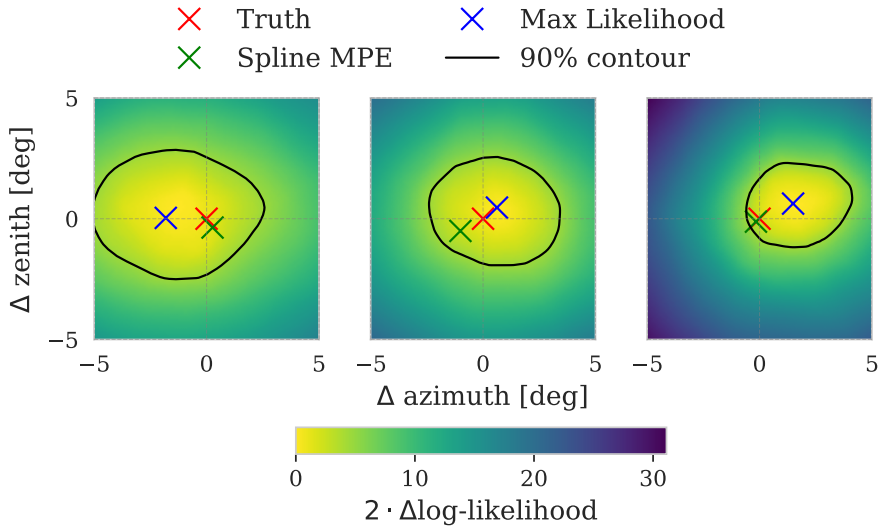

Figure 20 A grid scan of the likelihood for three example events from the evaluation dataset. Here, two times the delta log-likelihood is taken between the maximum likelihood and the likelihood at that given direction. The 90% contour line shows the level where the cumulative χ^2 distribution with 2 degrees of freedom reaches 90%. The red, green and blue marker show the true direction, the reconstructed direction from SplineMPE and the maximum likelihood direction from this model respectively.

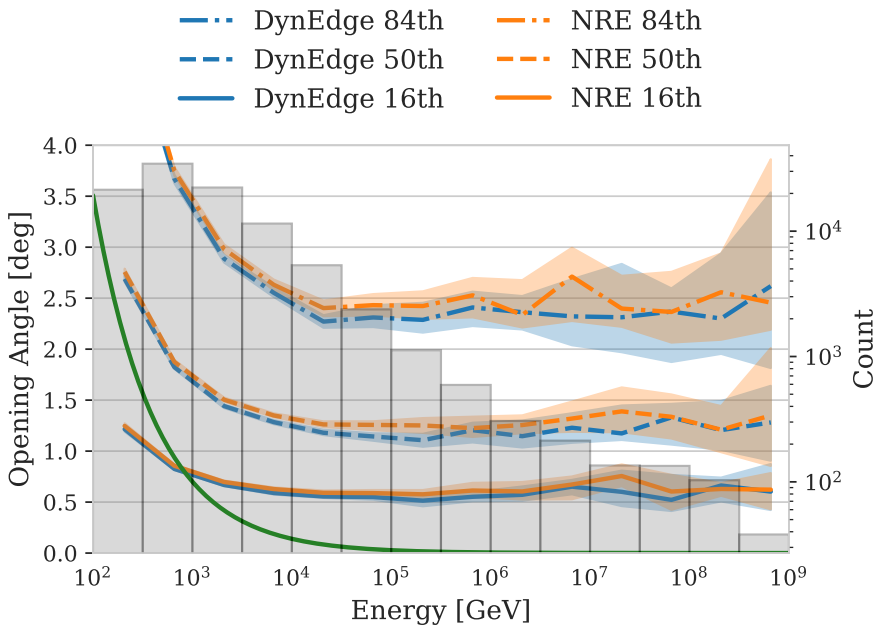

Figure 21 Performance of the von Mises-Fisher model in terms of opening angle between point prediction and truth. This plot shows the 16th, 50th, and 84th percentile for each energy bin. In the background, the grey histogram represents the events being evaluated.

8.3. Sequential NRE

Another approach tested in this thesis is called Sequential NRE. The general idea is to increase the performance by training a sequence of models in which each model refines the predictions of the previous model. It does so by reweighting the samples from the uncorrelated class 0 that come from $p(\mathbf{x})p(\boldsymbol{\theta})$ with the likelihood to evidence ratio $\hat{r}_{n-1}$ predicted by the previous model:

$$p(\mathbf{x})p(\boldsymbol{\theta}) \cdot \hat{r}_{n-1} = p(\mathbf{x})p(\boldsymbol{\theta}) \cdot \frac{p_{n-1}(\mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x})} = p_{n-1}(\mathbf{x}, \boldsymbol{\theta}). \tag{8.4}$$

The classifier now has the task to distinguish between the true joint $p(\mathbf{x}, \boldsymbol{\theta})$ and the joint as predicted by the first model $p_{n-1}(\mathbf{x}, \boldsymbol{\theta})$. An optimal classifier d^* therefore outputs

$$d^*(\mathbf{x}, \boldsymbol{\theta}) = \frac{p(\mathbf{x}, \boldsymbol{\theta})}{p(\mathbf{x}, \boldsymbol{\theta}) + p_{n-1}(\mathbf{x}, \boldsymbol{\theta})}, \tag{8.5}$$

which can be rearranged to

$$\frac{d^*(\mathbf{x}, \boldsymbol{\theta})}{1 - d^*(\mathbf{x}, \boldsymbol{\theta})} = \frac{p(\mathbf{x}, \boldsymbol{\theta})}{p_{n-1}(\mathbf{x}, \boldsymbol{\theta})} = \frac{p(\mathbf{x}|\boldsymbol{\theta})}{p_{n-1}(\mathbf{x}|\boldsymbol{\theta})}. \tag{8.6}$$

We can multiply this to the approximated likelihood-to-evidence ratio from the previous model to get a refined approximation:

$$\frac{p_{n-1}(\mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x})} \cdot \frac{p(\mathbf{x}|\boldsymbol{\theta})}{p_{n-1}(\mathbf{x}|\boldsymbol{\theta})} = \frac{p(\mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x})}. \tag{8.7}$$

In theory, this sequence of models could be extended indefinitely. Eventually, if the discriminator can not differentiate between the output of the previous model and the actual likelihood, it will simply return 1. In this section, we focus on a sequence of two models, to see if the second model actually improves the classification of the first model.

8.3.1. Architecture and Training

In this case, we use the basic NRE model from Section 8.1 as our first model. The second model has the same architecture as the first model. The only difference is that during training, the loss function for class 0 samples will be multiplied by the predicted likelihood to evidence ratio from the first model. Besides this, the training is done similar to the training described in Section 8.1.1.

8.3.2. Classification Performance

To evaluate the Sequential NRE, we analyze both its classification capabilities and its ability to approximate likelihood ratios for directional reconstruction. The classification capabilities are measured by letting the full sequential model classify between correlated and uncorrelated

samples.

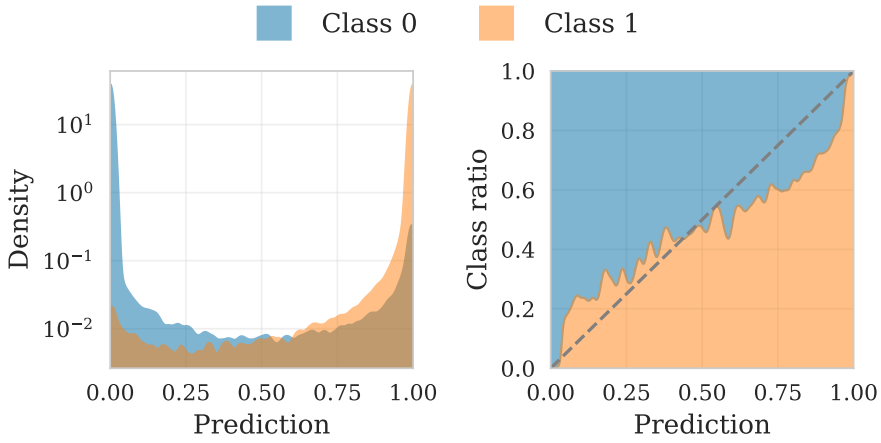

Figure 22 Classification performance of the Sequential NRE model. **Left:** Output distributions of the classifier showing predictions for true positive (blue) and true negative (red) classes. **Right:** Calibration curve showing the ratio of true positive examples to all examples as a funreweightingction of prediction score.

Figure 22 shows the classification performance of the Sequential NRE model. The left plot displays the distribution of classifier predictions for both true classes. As expected, it shows very good discrimination capabilities. The calibration curve on the right however shows slight unbalances. This is due to the fact that there are fewer events which were predicted to be close to their respective target but not entirely close. This can also be seen in the left plot when comparing with the previous classification plots. The densities show a steeper decline before they settle around 10^{-2}.

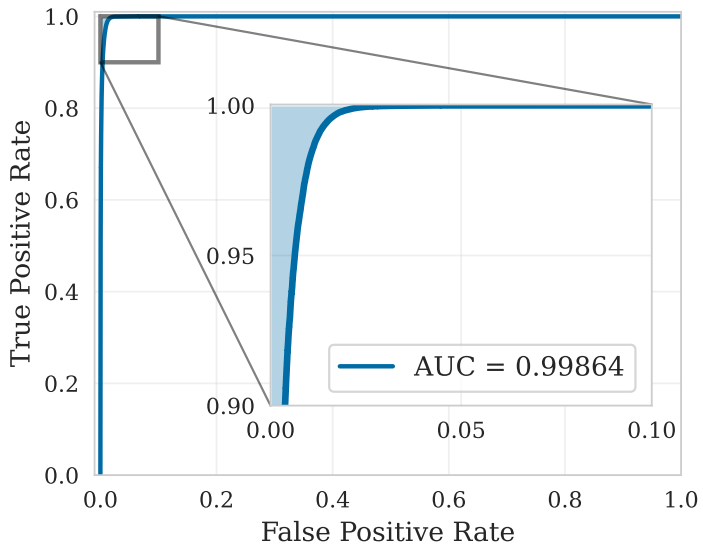

Figure 23 ROC curve for the Sequential NRE classifier. The curve plots the TPR against the FPR at various threshold settings.

Figure 23 shows the ROC curve for this model. The model achieves an AUC of 0.99864, which is an improvement to the previous models.

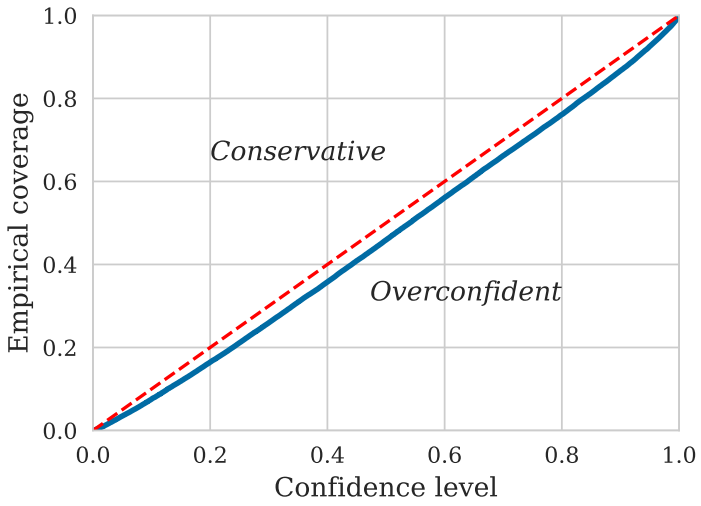

Figure 24 Coverage plot for the Sequential NRE model. The coverage is evaluated based on 100,000 evaluation events. Ideal coverage, corresponding to the χ^2 distribution with 2 degrees of freedom, is given by the dashed line.

8.3.3. Likelihood Approximation Performance

The coverage plot in Figure 24 shows a slight overconfidence across all confidence levels. This might be linked to the findings from the calibration curve and might be the result of the sequential approach. While in this case, the deviations from the ideal lines are only small, it should be further investigated for bigger imbalances.

The contours for the example events can be seen in Figure 25. The contours are in accordance to what was expected. The model found maximum likelihood points that are very similar to the ones found by the von Mises-Fisher model in Section 8.2.3.

The angular resolution in Figure 26 confirms that the sequential approach improved the resolution of the original basic NRE model. However, the resolution still remains slightly worse than the resolution of the backbone model. Further studies could be made by adding more models to the sequence or combining different model architectures.

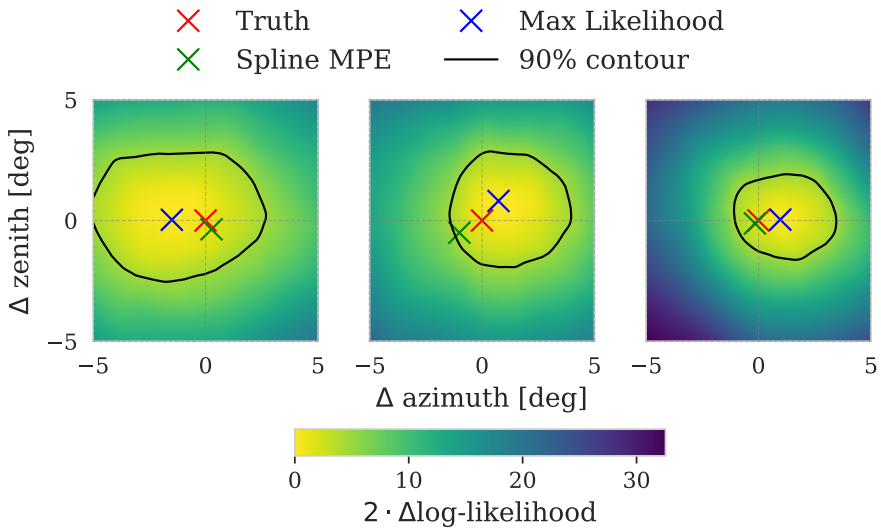

Figure 25 A grid scan of the likelihood for three example events from the evaluation dataset. Here, two times the delta log-likelihood is taken between the maximum likelihood and the likelihood at that given direction. The 90% contour line shows the level where the cumulative χ^2 distribution with 2 degrees of freedom reaches 90%. The red, green and blue marker show the true direction, the reconstructed direction from SplineMPE and the maximum likelihood direction from this model respectively.

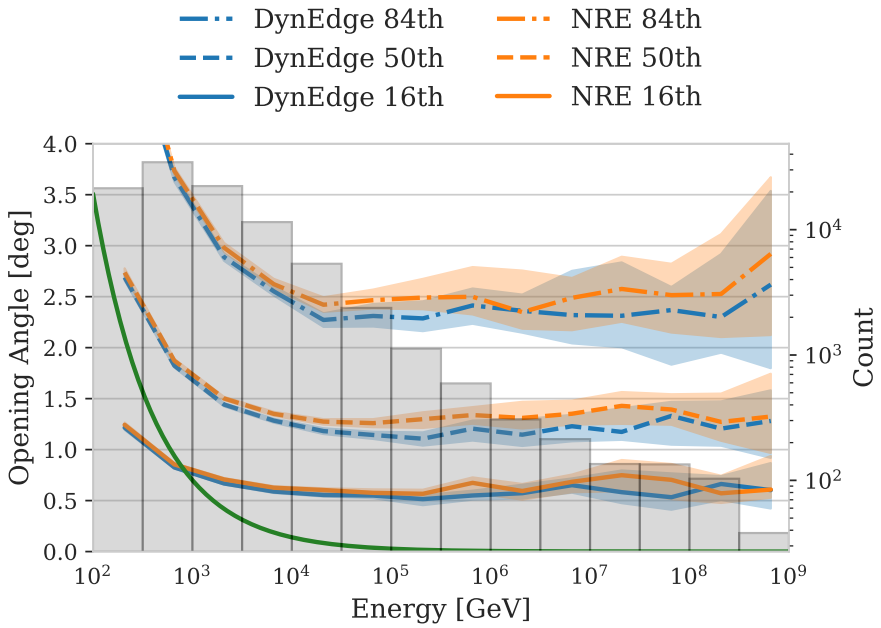

Figure 26 Performance of the Sequential NRE model in terms of opening angle between point prediction and truth. This plot shows the 16th, 50th, and 84th percentile for each energy bin. In the background, the grey histogram represents the events being evaluated.

8.4. NRE with injected DynEdge Point Estimate

In this section we want to analyze whether injecting a Point Estimate (PE) for the direction coming from a model solely trained on directional reconstruction can help the NRE model to not lose angular resolution when approximating the likelihood. We try this method for PEs from the DynEdge model which is used as the backbone.

8.4.1. Architecture and Training

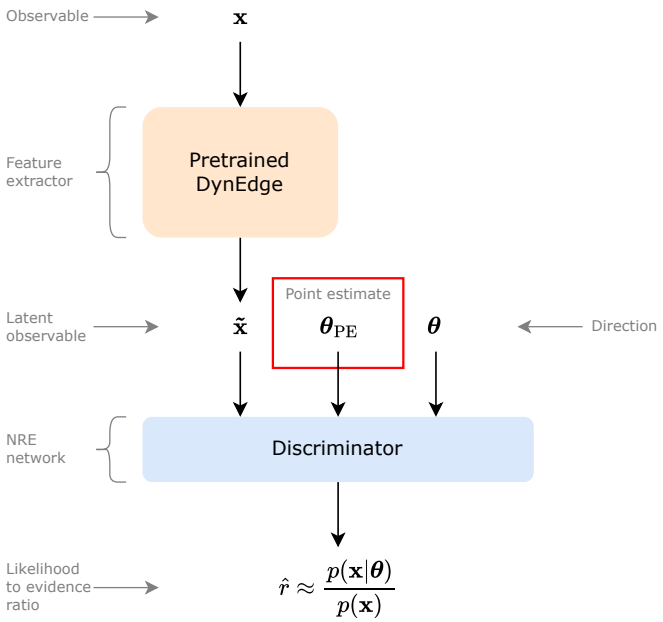

Figure 27 The architecture for the NRE method with injected PE. The observed pulsemap will go through DynEdge to generate a latent representation of the observable. This is then concatenated with the PE and the direction of interest and put through the discriminator, which then gives us the likelihood to evidence ratio.

The injection happens after the event has gone through the feature extractor. Now, we not only concatenate the latent representation and the direction which we are evaluating, but also the given PE. This can be seen in Figure 27. The PE is denoted as θ_{PE}. In principle, the discriminator now should have an easier task to decipher whether the observable and the direction are correlated or not. At the same time, it has direct access to a good prediction of the maximum likelihood direction, which could help in refining the likelihood estimations.

Again, for comparability, the training is performed using the same parameters as described in Section 8.1.1.

8.4.2. Classification Performance

First, the classification performance of a model with DynEdge injection is analyzed.

Figure 28 shows the classification performance of NRE model with injected DynEdge PE. The

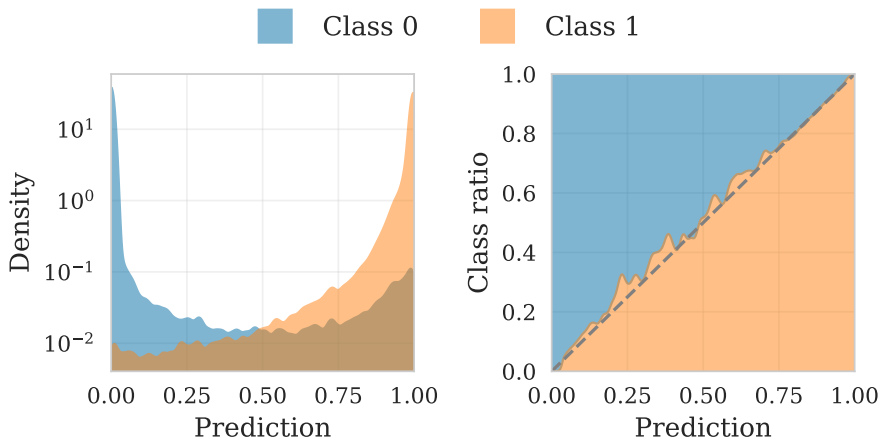

Figure 28 Classification performance of the NRE with injected DynEdge PE model. **Left:** Output distributions of the classifier showing predictions for true positive (blue) and true negative (red) classes. **Right:** Calibration curve showing the ratio of true positive examples to all examples as a function of prediction score.

left plot displays the distribution of classifier predictions for both true classes, while the right plot displays the calibration curve of the same predictions. As before, the classifier worked well in distinguishing both classes. The calibration curve looks near perfect, indicating a well calibrated classifier that does not perform better for either class.

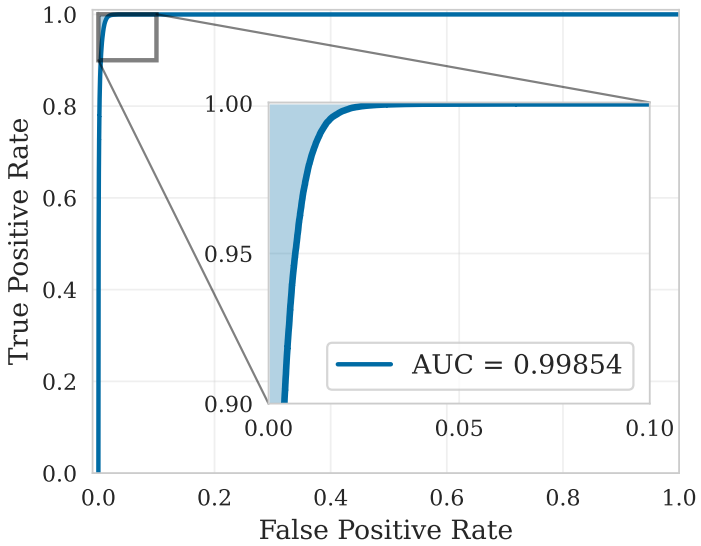

Figure 29 ROC curve for the NRE model with injected DynEdge PE. The curve plots the TPR against the FPR at various threshold settings.

Figure 29 shows the ROC curve for this model. The model attains an AUC of 0.99854, which is comparable to the performance of previous models. The value exceeds that of the basic NRE model, suggesting that the incorporation of the point estimate enhances classification performance.

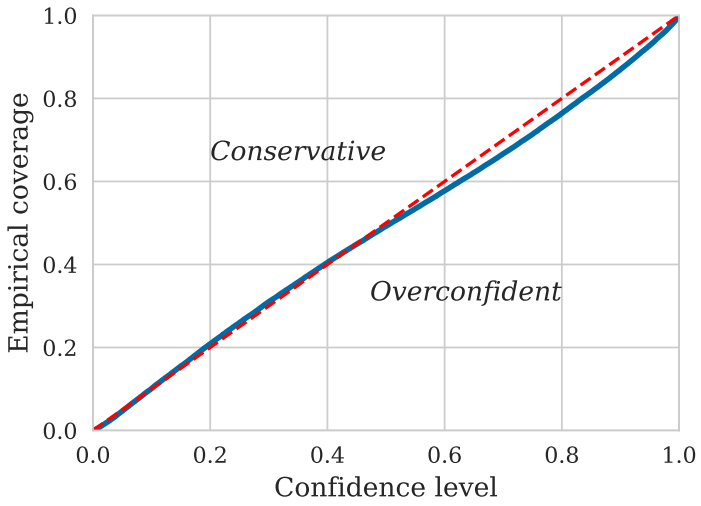

Figure 30 Coverage plot for the NRE model with injected DynEdge PE. The coverage is evaluated based on 100,000 evaluation events. Ideal coverage, corresponding to the χ^2 distribution with 2 degrees of freedom, is given by the dashed line.

8.4.3. Likelihood Approximation Performance

Figure 30 shows the coverage properties of the model with DynEdge PE injection. It shows a slightly overconfident curve for higher confidence levels, very similar to what we have seen for the von Mises Fisher sampling model (see Figure 19). In this case, as for the von Mises Fisher sampling case, the deviations from the ideal coverage are rather small. For larger deviations however, the cause should be studied, because in that case the coverage will not be following Wilks' theorem properly anymore.

Figure 32 shows the resolution of the NRE model with injected PE. It also shows an improvement compared to the basic NRE model. The gap between DynEdge backbone model and NRE is very small, especially for small energies.

Injecting the PE therefore helped the model to distinguish between correlated and uncorrelated classes and improved the resolution of the model. The coverage shows slight overconfidence for higher confidence levels.

A similar model has been trained with injection of SplineMPE PEs. This investigation aimed to determine whether such integration would further reduce the overall resolution of the model, potentially approaching the resolution capabilities of the SplineMPE reconstruction. While this approach improved the resolution compared to the basic DynEdge NRE model, it was not able to exceed the resolution of the pre-trained DynEdge model, even when injecting PEs that are objectively better than what the backbone can achieve. Therefore, the model maintains substantial dependence on the latent event representation, as this encoding contains critical information regarding directional uncertainty. The plots corresponding to the SplineMPE PE injection model can be found in Appendix A.

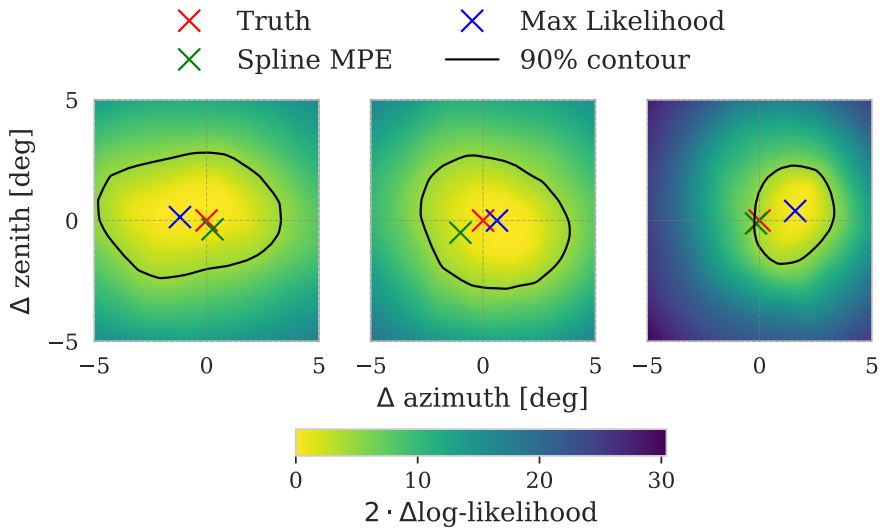

Figure 31 A grid scan of the likelihood for three example events from the evaluation dataset. Here, two times the delta log-likelihood is taken between the maximum likelihood and the likelihood at that given direction. The 90% contour line shows the level where the cumulative χ^2 distribution with 2 degrees of freedom reaches 90%. The red, green and blue marker show the true direction, the reconstructed direction from SplineMPE and the maximum likelihood direction from this model respectively.

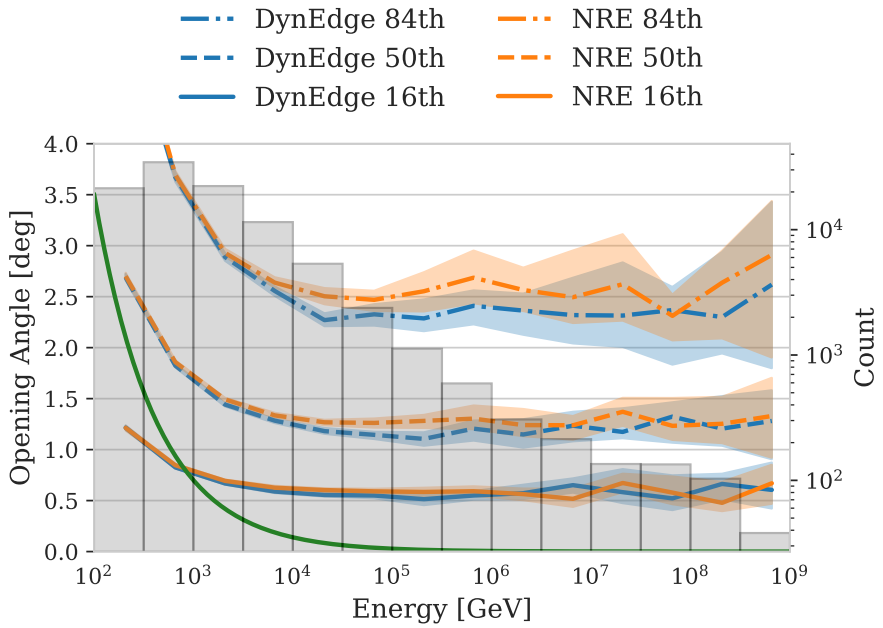

Figure 32 Performance of the NRE model with injected PE in terms of opening angle between point prediction and truth. This plot shows the 16th, 50th, and 84th percentile for each energy bin. In the background, the grey histogram represents the events being evaluated.

8.5. Ensemble of NRE models

Another approach to improving the performance is to combine predictions from several models. This technique is called ensembling and shows good performance in many ML applications. In this case, we want to analyze whether the predictions of the basic NRE model (Section 8.1), the von Mises Fisher sampling model (Section 8.2) and the Sequential NRE model (Section 8.3) combined reach better angular resolution while approximating the likelihood. We utilize the pre-trained models and compute the average of their predictions. Since all models exhibit similar performance, we do not assign weights to their predictions.

8.5.1. Classification Performance

First, the classification performance is analyzed. The left plot in Figure 33 shows the distribution of classifier predictions for both true classes. It comes to no surprise that an ensemble of good classifiers itself is a good classifier, showing very good discrimination between the two classes. The calibration curve on the right shows a slight imbalance of the classes, which indicates that the model has more problems in classifying samples from the uncorrelated class.

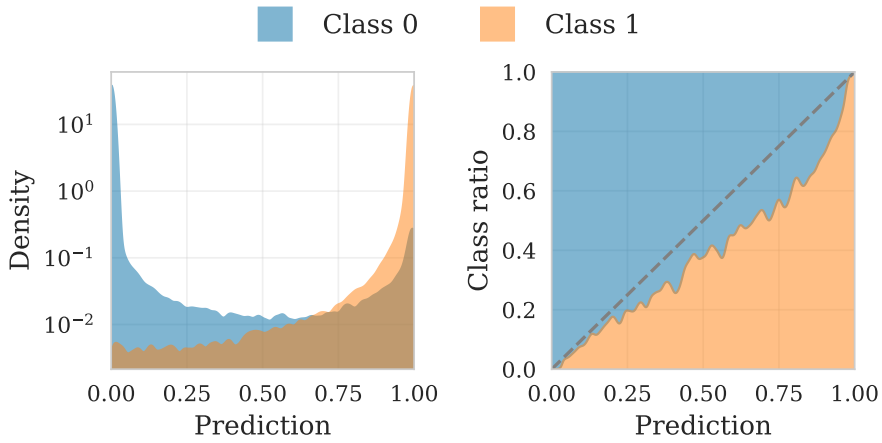

Figure 33 Classification performance of the ensemble model. **Left:** Output distributions of the classifier showing predictions for true positive (blue) and true negative (red) classes. **Right:** Calibration curve showing the ratio of true positive examples to all examples as a function of prediction score.

The ROC curve in Figure 34 also looks as expected. The ensemble shows an almost perfect classification performance.

8.5.2. Likelihood Approximation Performance

Figure 35 shows the coverage properties of the ensemble. It shows a slightly conservative curve, which means the truth is actually more often contained than the confidence level indicates. In principle, conservative is better than overconfident, because the contours can really be trusted. And in this case, the deviation from the ideal coverage is only rather small. When looking at the example events in Figure 36, it is evident that averaging the likelihoods of the models also combines their contour regions, making them slightly bigger than the individual

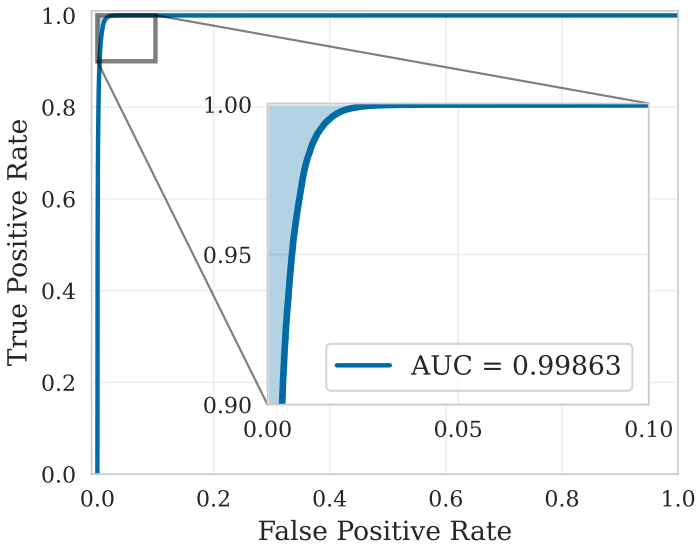

Figure 34 ROC curve for the ensemble classifier. The curve plots the TPR against the FPR at various threshold settings.

ones. In this case, the coverage only shows a small deviation from the ideal coverage. For larger deviations however, the cause should be studied, because the contour lines are larger than necessary and the model misevaluates its own capabilities.

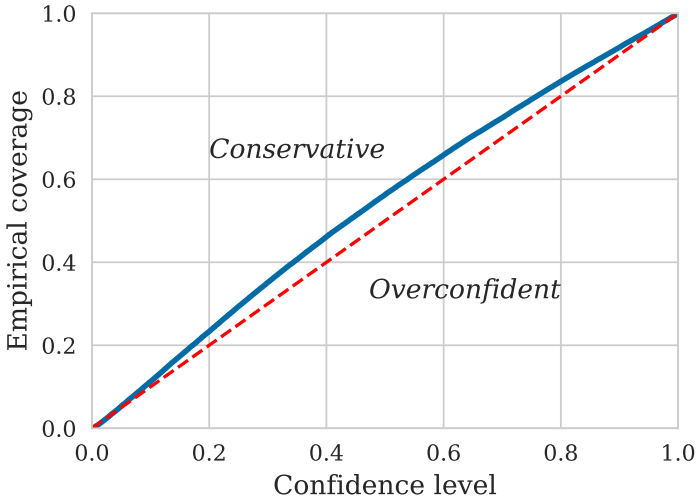

Figure 35 Coverage plot for the ensemble model. The coverage is evaluated based on 100,000 evaluation events. Ideal coverage, corresponding to the χ^2 distribution with 2 degrees of freedom, is given by the dashed line.

A good impact can be seen when looking at the directional reconstruction performance of the ensemble in Figure 37. Here, the advantage of ensembling methods can be visible in a resolution that matches the resolution of the DynEdge backbone. By combining the three predictions, small errors of each model will be compensated by the other models, resulting in more accurate predictions. While this is a good step in the right direction, the angular resolution is still not clearly better than the resolution from the backbone. While this might be an intrinsic limit of the method or architecture in use, combining more models with different pre-

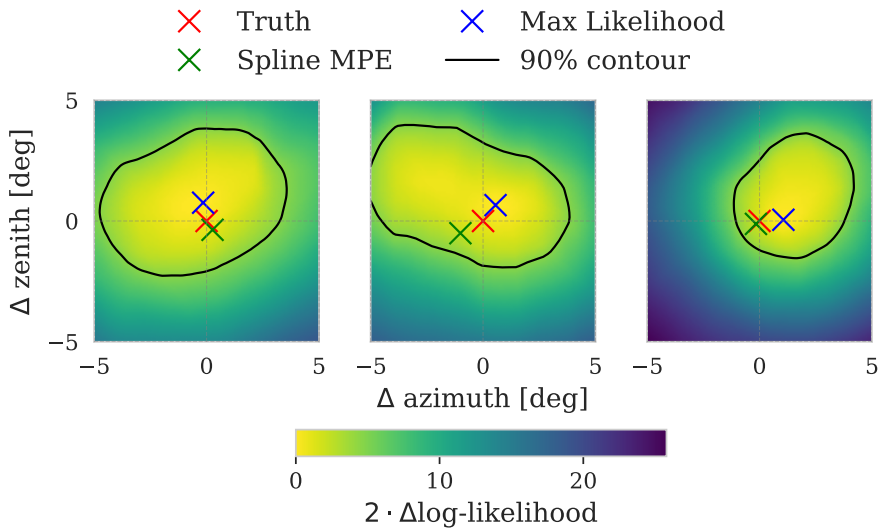

Figure 36 A grid scan of the likelihood for three example events from the evaluation dataset. Here, two times the delta log-likelihood is taken between the maximum likelihood and the likelihood at that given direction. The 90% contour line shows the level where the cumulative χ^2 distribution with 2 degrees of freedom reaches 90%. The red, green and blue marker show the true direction, the reconstructed direction from SplineMPE and the maximum likelihood direction from this model respectively.

trained backbones might help in further improving the resolution and therefore the capabilities of the model when applied to tasks like point source analyses.

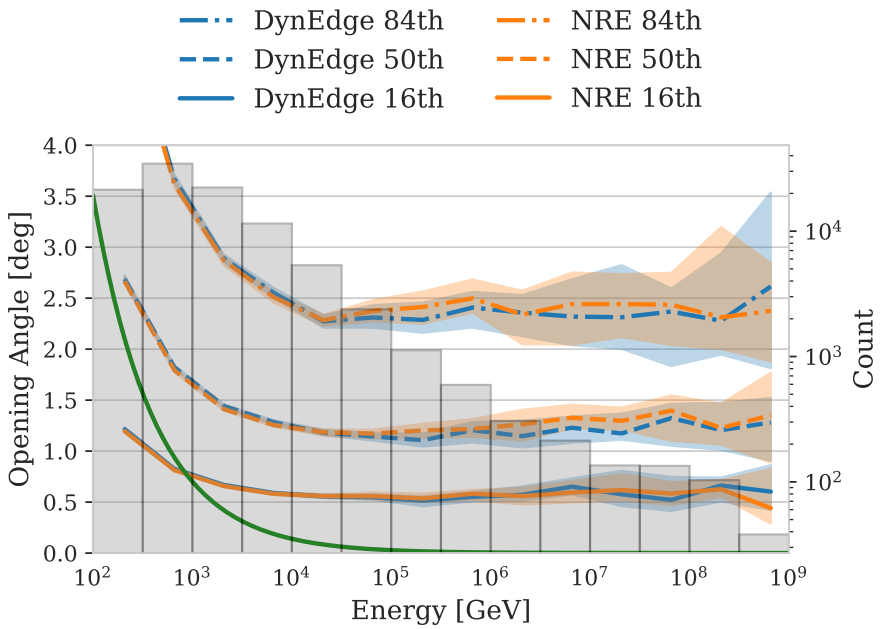

Figure 37 Performance of the ensemble model in terms of opening angle between point prediction and truth. This plot shows the 16th, 50th, and 84th percentile for each energy bin. In the background, the grey histogram represents the events being evaluated.

8.6. IceMix

Lastly, it is analyzed how the NRE method performs with a different, better backbone. For this task, the IceMix model is chosen, which is the second solution from the IceCube Kaggle competition (see Section 5.4). This model performed best on track events and outperformed the basic DynEdge model. It uses Fourier encoding to embed the variables of each pulse, incorporates a DynEdge-inspired encoder for feature extraction and utilizes a superior transformer architecture to predict the direction of the incoming neutrino.

8.6.1. Pre-training the Backbone

The IceMix model is pre-trained on an angular regression task just like the DynEdge model. For that, the IceMix implementation in GraphNeT can be used. Here, default parameters are specified that translate into a model with roughly 160 million parameters.

During training on the same Northern Tracks dataset as before, the pulses have an additional feature which is the hard local coincidence flag. This flag indicates whether a pulse was read out in coincidence with other pulses nearby. Again, we perform a pulse cut at 1,024 pulses. However this time, the pulse cut is applied by sampling 1,024 pulses from the event pulsemap if the event has more pulses. Pulses with the hard local coincidence flag will be preferred when subsampling, because these pulses are more likely to be actual signal pulses rather than random noise. Using this subsampling method, we can train on events with all numbers of pulses without having to discard the events with a high number of pulses.

Due to the model's increased complexity and substantially longer training duration, we only use a dataset containing roughly 1.8 million events, which are split into training and validation dataset using a 9:1 split. The training loss is the von Mises Fisher loss described in Section 7.2. The batch size in use is set to 30 and *ReduceLROnPlateau* schedules the learning rate using a factor of 0.1 and a patience of 2 epochs. The starting learning rate equals 0.001 and the optimizer in use is *Adam*. An early stopping mechanism is implemented with a patience of 5 epochs.

8.6.2. Performance of the pre-trained model

The evaluation of this pre-trained model is done on 100,000 events from the evaluation dataset described in Section 7.3. To facilitate a more direct comparison, we evaluated events with a pulse count of 1,024 or fewer and assigned a hard local coincidence value of 1 to all pulses.

As expected, the IceMix model performs substantially better than the DynEdge model, even though it is trained on much fewer events and does not get actual hard local coincidence information during evaluation. This can be seen in Figure 38. For all percentiles across the whole energy range, this model has a lower opening angle between reconstructed and true direction. The gap to the traditional reconstruction method SplineMPE is significantly reduced

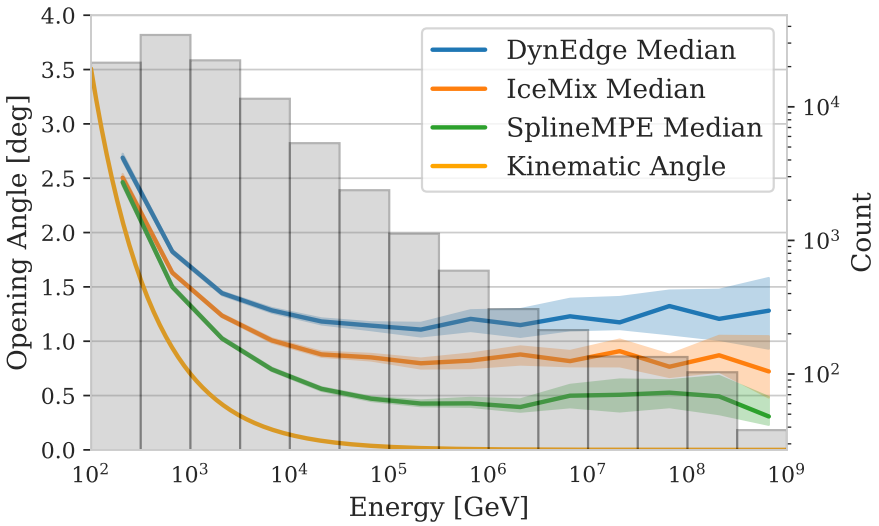

Figure 38 Median opening angle of the directional point estimates of the pre-trained IceMix model in comparison to the pre-trained DynEdge model and Spline MPE. In the background, the grey histogram represents the events that are being evaluated. These events come from a dataset that the model has not been trained on. The uncertainty bands come from bootstrapping and represent the 95% bands. Plotted in yellow is the mean kinematic angle between neutrino and muon, which represents a lower bound for the resolution of the reconstruction.

for all energies.

The higher resolution should give the corresponding NRE model a boost in its performance. However, while these improvements are very significant, they come at the expense of flexibility of the model and training and evaluation time. This should be considered when working with this model.

8.6.3. NRE Architecture and Training

For the NRE model, we use the same architecture as before (see Figure 9 in Section 8.1). The only thing that changes is the feature extractor, which does not use the DynEdge model but the pre-trained IceMix model without the prediction head. The parameters of the IceMix model stay frozen during training except for the last two layers of the MLP which is part of the last transformer block. Unfreezing these layers showed a slight performance improvement. The output of the backbone, which is our latent representation of the event observable, is 128 dimensional, just like before. The direction of interest in euclidean coordinates is concatenated to the latent representation. The discriminator architecture stays the same as for all models, utilizing an MLP with 10 hidden layers and roughly 480,000 trainable parameters.

The training is performed analogous to the pre-training of the IceMix backbone described in Section 8.6.1.

8.6.4. Classification Performance

To evaluate the performance of model IceMix, we analyze both its classification capabilities and its ability to approximate likelihood ratios for directional reconstruction. As for the per-

formance of the pre-trained model, we use the same evaluation dataset as for the DynEdge based models, which does not contain the hard local coincidence flag in its pulsemaps.

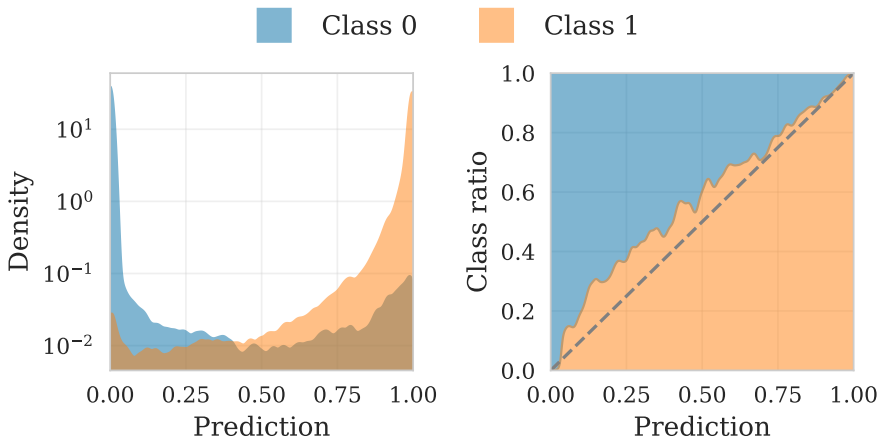

Figure 39 Classification performance of the IceMix model. **Left:** Output distributions of the classifier showing predictions for true positive (blue) and true negative (red) classes. **Right:** Calibration curve showing the ratio of true positive examples to all examples as a function of prediction score.

Figure 39 shows the classification performance of the IceMix NRE model. As all the models before, it shows good discrimination ability. However, the calibration curve shows a slight preference for class 1 at lower prediction scores, meaning the model is more confident in predicting uncorrelated samples from class 0.

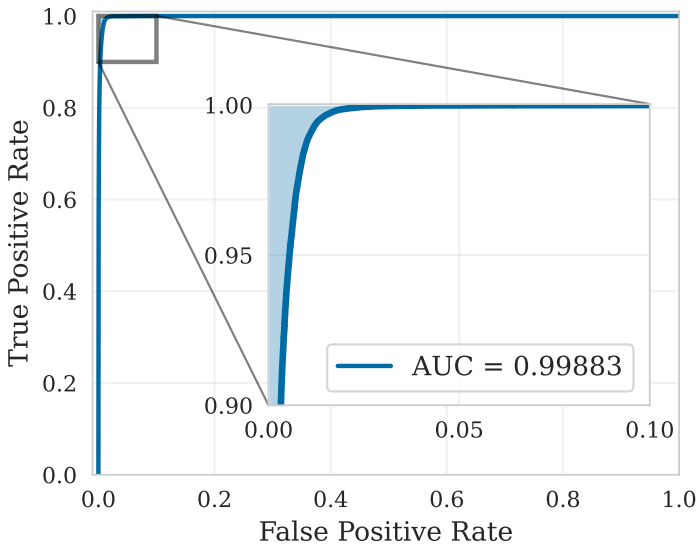

Figure 40 ROC curve for the IceMix classifier. The curve plots the TPR against the FPR at various threshold settings.

Figure 40 shows the ROC curve for this model. The model achieves an AUC of 0.99883, which is the highest of all trained models.

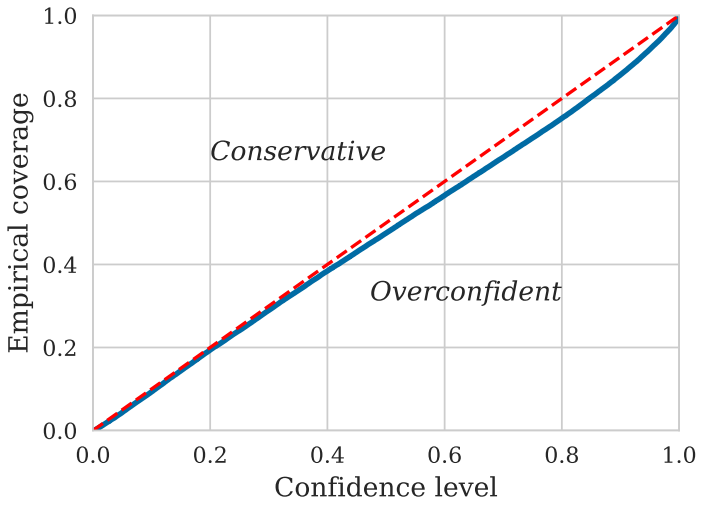

Figure 41 Coverage plot for the IceMix model. The coverage is evaluated based on 100,000 evaluation events. Ideal coverage, corresponding to the χ^2 distribution with 2 degrees of freedom, is given by the dashed line.

8.6.5. Likelihood Approximation Performance

Figure 41 shows the coverage properties of the IceMix model. It shows a slight overconfidence for higher confidence levels, meaning the likelihood contours are slightly too narrow at the base of the likelihood peak.While this may not appear particularly concerning at first glance, it should be examined further for robust analysis. A possible explanation could be the fact that the evaluation dataset does not contain hard local coincidence information.

When looking at the three example events in Figure 42, the model shows good recovery of the truth. For these events, the 90% contour are smaller than the ones from the basic DynEdge NRE model.

When it comes to the actual angular resolution, Figure 43 shows the performance we expected. It is much improved compared to the DynEdge based models. Nevertheless, the maximum likelihood point estimate of the IceMix NRE model also exhibits a marginally larger opening angle compared to the pre-trained backbone model's point estimate, which appears to represent an inherent limitation of this method. While it rules out that the architecture of the backbone causes this loss in precision, the training method of the pre-training could be changed to investigate the impact. Although training the entire architecture from scratch without pre-training represents a theoretically viable approach, our own attempts revealed that this methodology presented significant training challenges and yielded performance metrics inferior to those achieved with the pre-trained backbone architecture. However, the task of the pre-training could be changed. For example, it could be trained on more than just the direction and uncertainty parameter. The energy could also be added to the pre-training process, which might give the resulting latent vector more information about the nature of the event and the corresponding likelihood.

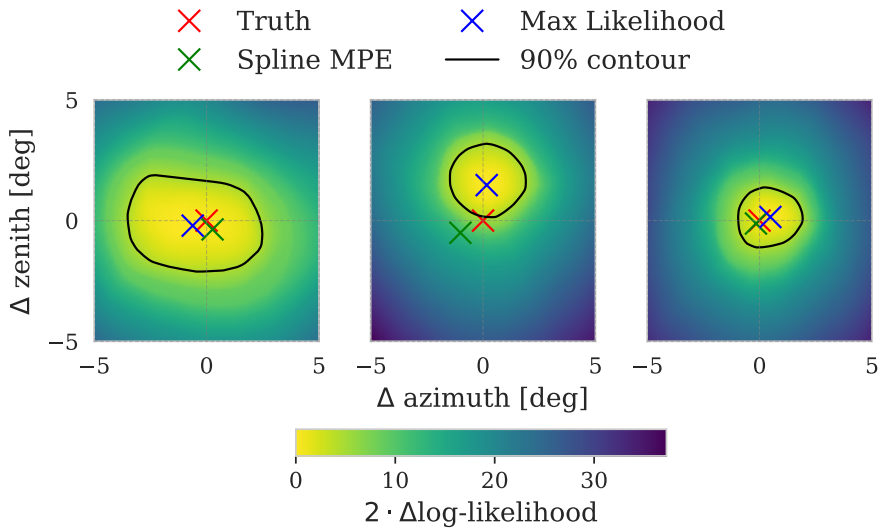

Figure 42 A grid scan of the likelihood for three example events from the evaluation dataset. Here, two times the delta log-likelihood is taken between the maximum likelihood and the likelihood at that given direction. The 90% contour line shows the level where the cumulative χ^2 distribution with 2 degrees of freedom reaches 90%. The red, green and blue marker show the true direction, the reconstructed direction from SplineMPE and the maximum likelihood direction from this model respectively.

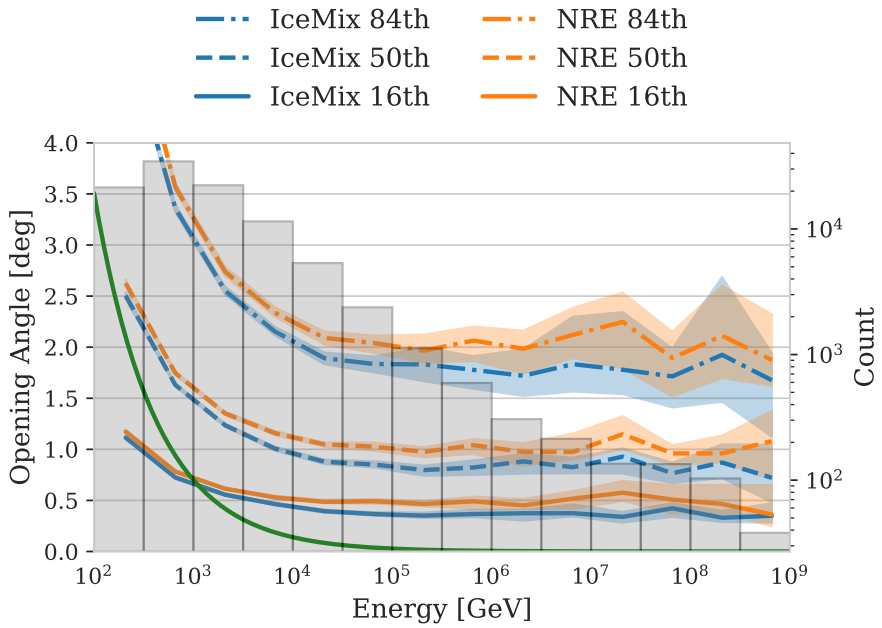

Figure 43 Performance of the IceMix model in terms of opening angle between point prediction and truth. This plot shows the 16th, 50th, and 84th percentile for each energy bin. In the background, the grey histogram represents the events being evaluated.

9. NPE Model Development

As described in Chapter 6, NRE has a Bayesian twin, the NPE. This method approximates the posterior distribution $p(\boldsymbol{\theta}|\mathbf{x})$ rather than the likelihood to evidence ratio $\frac{p(\mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x})}$. This yields the opportunity to make analyses in the Bayesian framework. However, if the frequentist framework is preferred, we can still obtain the likelihood ratio, which is used in the test statistic of our directional analyses of IceCube data. All we need is knowledge of the prior $p(\boldsymbol{\theta})$:

$$\Delta \log \mathcal{L} = \log \frac{\mathcal{L}(\boldsymbol{\theta}_1)}{\mathcal{L}(\boldsymbol{\theta}_2)} = \log \frac{p(\boldsymbol{\theta}_1|\mathbf{x})}{p(\boldsymbol{\theta}_2|\mathbf{x})} - \log \frac{p(\boldsymbol{\theta}_1)}{p(\boldsymbol{\theta}_2)} \tag{9.1}$$

While DNFs represent a viable tool for NPE, this thesis proceeds directly to the examination of CNF and FM. However, the GraphNeT framework recently received an implementation of the DNF library jammy flows[1], which will make research into DNF more accessible for the neutrino astronomy community.

For the implementation of the FM algorithm, we use an adaptation of the Flow Matching for Atmospheric Retrieval (FM4AR) repository[2]. This repository uses CNFs to infer atmospheric properties of exoplanets from observed electromagnetic spectra. While this task in principle is vastly different from reconstructing neutrino directions, the framework still allows for easy adaptation.

9.1. Architecture and Training

The general architecture of the method can be seen in Figure 44. Similarly to the NRE models, we first use a frozen feature extractor to obtain a latent representation of the events. In this case, the pre-trained DynEdge model is utilized. The 128 dimensional latent representation does then go through a simple embedding MLP with 128 dimensions each. The time and the direction are also embedded using a fourier encoder (similar to the one used in the IceMix model) and MLP with 128 total dimensions. The embedded latent observable, time and direction are then concatenated and given as the input to the NPE network with a total of almost 400 million parameters. The NPE network forms the time dependent vector field that based on the given context, in our case the latent observable of an event, transforms the base distribution into the posterior distribution over time.

To solve equation 6.9 in order to evaluate the posterior distribution for a given $\boldsymbol{\theta}$, a DOPRI5[3] ODE solver is utilized. It integrates the divergence of the vector field from $t = 0$ to $t = 1$

[1] The GitHub repository can be found **here.**
[2] The GitHub repository can be found **here.**
[3] The ODE solver makes use of the Dormand-Prince method of order 5. The corresponding repository can be found **here.**

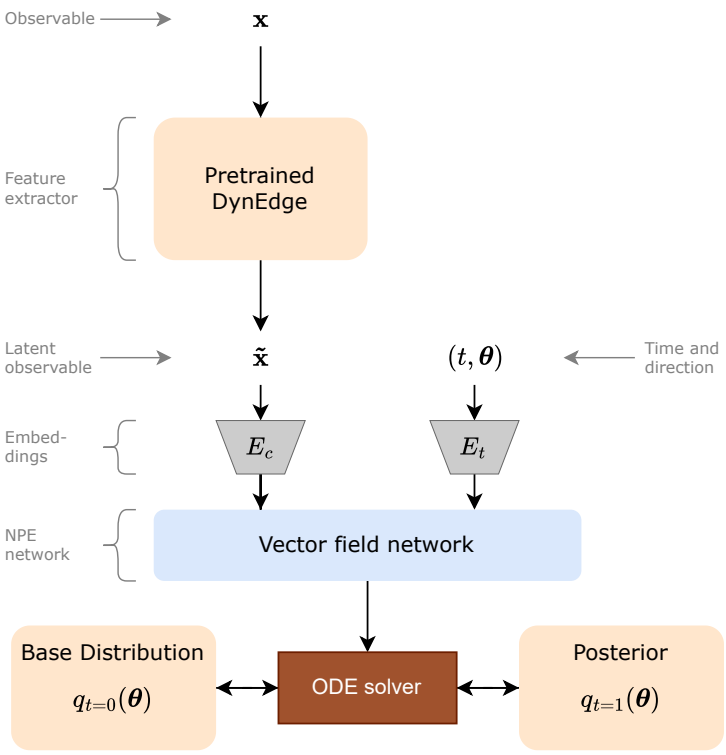

Figure 44 The architecture for the NPE method using CNFs and FM. The observed pulsemap will go through DynEdge to generate a latent representation of the observable. After a few simple embedding layers, the latent representation will act as a context to condition the time dependent neural vector field. An ODE solver will then query the vector field based on t and $\boldsymbol{\theta}$ to transform the base distribution into the posterior distribution (see Section 6.3).

and simultaneously obtains the corresponding $\boldsymbol{\theta}_0$ (this method is shown in Appendix C in [47]), which is all that is needed for the posterior distribution. The base distribution is set as a standard Gaussian in three dimensions and σ_{min}, which is the standard deviation of the Gaussian distribution around our target parameters, is set to 0.0001.

The training procedure is described in Section 6.3. The optimizer chosen is *Adam*. The training utilizes the same dataset as for the DynEdge NRE models. The batch size equals 15,000 and the learning rate scheduler is set to *CosineAnnealingLR*, which reduces the learning rate from a starting learning rate of $5 \cdot 10^{-5}$ following a cosine function over 2000 epochs. An early stopping patience is set to 200 epochs.

Convergence for this model requires a substantial number of epochs, although each epoch has a relatively short training duration of approximately two minutes on an NVIDIA A100 GPU. During the initial 100 to 200 epochs, the loss curves reach a plateau, as the model primarily learns the general directional distribution of the data without incorporating the event observable. Over time, the model manages to escape this plateau; however, it requires approximately 1,500 epochs for the validation loss to reach its minimum. While this suggests

that a higher learning rate or a smaller vector field model could be beneficial, trials revealed that either adjustment significantly increased training difficulty.

9.2. Performance

Because this training task is about adjusting a vector field such that it creates the optimal transport path, there are no classification metrics to analyze as for the NRE models. As previously discussed, the ODE solver can be employed to evaluate points within the target posterior distribution. However, this process is computationally intensive due to the high parameter count of the implemented vector field network. Generating meaningful posterior skymaps or likelihood contours requires several minutes per event, which becomes prohibitive when analyzing coverage and angular resolution, as these analyses need evaluations for thousands of events. Because of these constraints, we will only look at the three example events from before. Figure 45 shows the posterior distributions of these events as inferred by the NPE model.

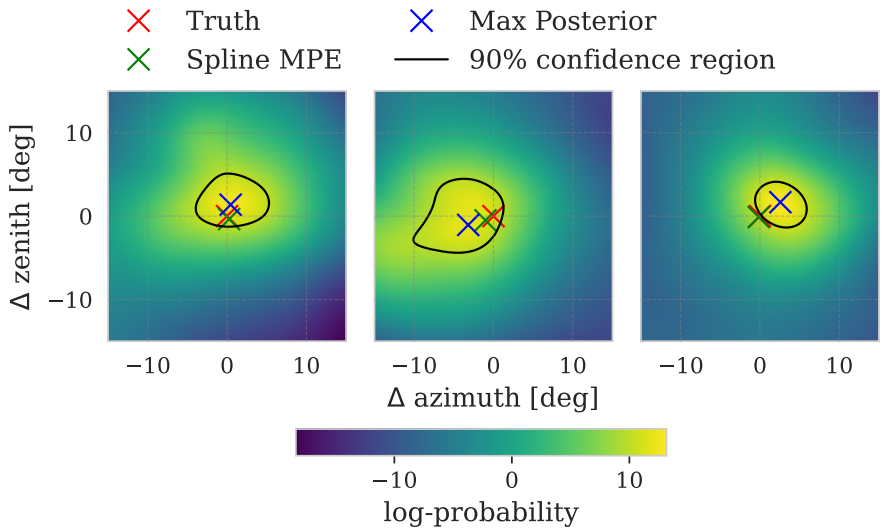

Figure 45 A grid scan of the posterior for three example events from the evaluation dataset. The 90% confidence line shows the level for which the posterior distribution holds roughly 90% of its mass. The red, green and blue marker show the true direction, the reconstructed direction from SplineMPE and the maximum posterior direction from this model respectively.

These plots show that also NPE can be used to recover the direction from neutrino events. The 90% confidence regions are determined by ranking the evaluated grid points based on their probability and sequentially summing them until they comprise 90% of the total probability across the grid. This assumes that the probabilities at points outside of this grid are sufficiently small. The last added probability then indicates the level for the contour line.

Because the prior is known, we can check whether the contour lines that come from Wilks' theorem match the lines for the Bayesian 90% confidence regions. In these cases, a flat prior can be assumed over the evaluated region. The delta log-likelihood can then be calculated

by just subtracting the logarithm of the posterior at a given grid point from the maximum log-posterior. Multiplied by 2, Wilks' theorem can be applied to generate the contours in the frequentist framework. This can be seen in Figure 46.

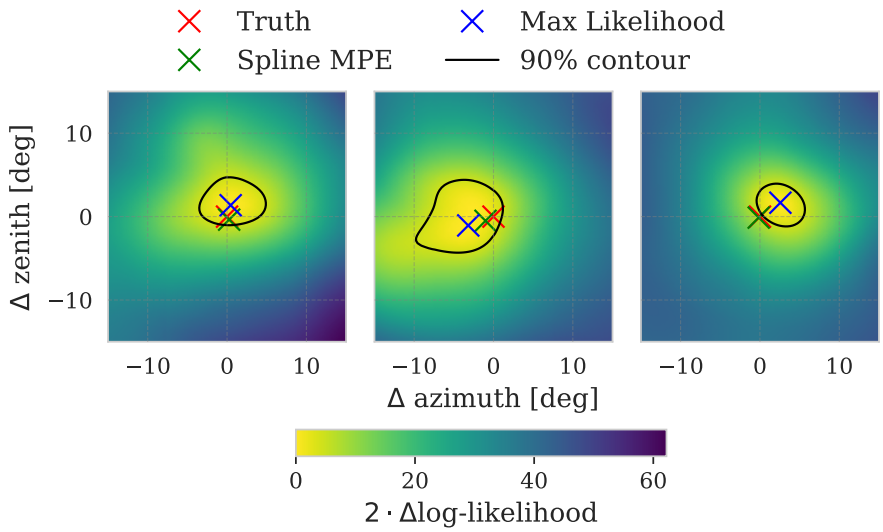

Figure 46 A grid scan of the likelihood for the three example events from the evaluation dataset. Here, two times the delta log-posterior is taken between the maximum probability and the posterior at that given direction. Because the priors are sufficiently flat, this yields the log-likelihood ratio. The 90% contour line shows the level where the cumulative χ^2 distribution with 2 degrees of freedom reaches 90%. The red, green and blue marker show the true direction, the reconstructed direction from SplineMPE and the maximum posterior direction from this model respectively.

The contours look almost identical to the Bayesian confidence regions, which indicates the success of estimating the posterior (and subsequently the likelihood) for the given events. Because of the slow inference time however, this method is not a viable option to analyze the thousands of events from neutrino detectors like IceCube. This might change however with better hardware, faster ODE solvers, better, more lightweight vector field networks or faster inference methods.

While this implementation proved to be working, an improvement could be made in the embedding of the directions. While in principal, only two parameters are inferred, namely azimuth and zenith, this model relied on euclidean coordinates to account for the spherical nature of the directions. These coordinates are not inherently bounded, and while the model quickly adapts to the target distribution on a unit sphere, this representation can still introduce challenges during training. A spherical embedding for the base distribution, the target distribution as well as the probability paths could solve this problem and make it easier for the model to train. This could also mean that a smaller, more lightweight model with fewer parameters may be sufficient to model the probability paths, which in turn would make inference time shorter.

10. Applications

In this chapter, the performance of the NRE models is evaluated in the context of their application to the target data intended for analysis. First, we are performing a point source analysis for a simulated pseudo dataset that mocks an NGC 1068-like source based on the IceCube's study in [48]. The goal is to see whether our model is able to find an excess of signal events at the source location. Second, we want to discuss whether real-time multi-messenger astronomy can benefit from the models discussed in this thesis by analyzing its performance on alert events. Alerts are usually high energy tracks that the first stages of data processing in IceCube already flag as events with high probability of being of astrophysical origin. IceCube then gives out an alert to the multi-messenger community in order for other telescopes to quickly point into the alert direction and observe coinciding events in their respective multi-messenger regime.

10.1. Mock Point Source Analysis for NGC1068 like Pseudodata

The main motivation for this thesis was to find ways of approximating the intractable directional likelihoods of neutrino events in a fast and flexible way without having to rely on summary statistics and other assumptions. The NRE models developed in this thesis can directly infer the likelihood at a given point from the pulse-level event data. While the angular resolutions show room for improvements, the coverages looked promising. In this section we want to investigate how this translates into an actual point source analysis for a NGC 1068 like source.

The events used for this analysis come from the same dataset from which the training and evaluation datasets were taken. In total, about 37,000 simulated events were used to mimic data from a source at a location with right ascension $ra = 60^\circ$ and declination $\alpha = 15^\circ$ (actual location of NGC 1068: $ra = 40.67^\circ$, $\alpha = -0.01^\circ$) as well as a fixed energy spectrum with $\gamma = 3.2$. The events are selected based on the reconstruction of SplineMPE. For the background event flux we would expect, the event selection takes the events that SplineMPE reconstructed within a 30° by 30° window with the source location at the center. However, because our model does not use SplineMPE as a reconstruction basis, we perform our own cut by finding the maximum likelihood estimates from our model and only use those events within a 10° circle around the source location. The cut was made because in the reconstruction of the maximum likelihood points for the background events, an artifact-like bias was found. Cutting these events prevents the bias to influence the analysis, although the source of these artifacts should be investigated and eliminated in future applications.

On top of the background, we injected 81 signal events at the source location. This number corresponds to the actual number measured in the NGC 1068 study [48]. The injection allows

us to test whether our NRE model can detect an excess of events from the source direction amid the background.

The methodology for our point source analysis using the NRE model utilizes the following approach. First, we apply a forward pass of our trained models to the entire dataset, which yields the spatial signal-over-background PDF for each event at a grid of evaluation points around the source location. Next, we fit the number of signal events n_s in our TS

$$TS = 2 \cdot \sum_i \log\left(\frac{n_s}{N}\left(\frac{S}{B}\right)_i + \frac{N - n_s}{N}\right) \tag{10.1}$$

to maximize the TS at the source location. Here, N denotes the total number of events from the dataset from which the mock dataset around the source was taken. For all events that are outside the given window, we assume the signal over background ratio to be zero. The

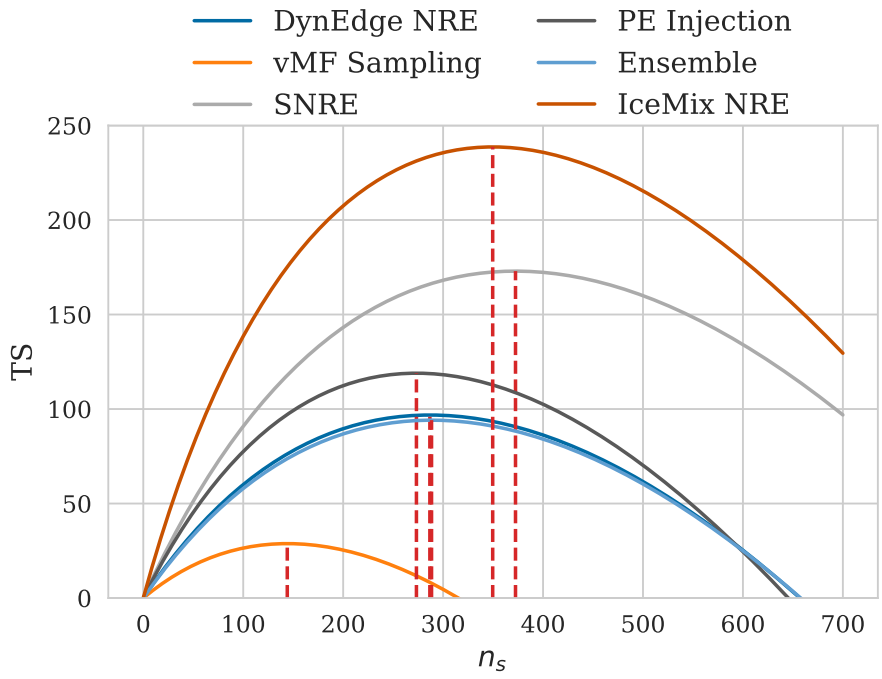

Figure 47 The TS at the the source location for different n_s and different NRE models. The n_s that maximize the TS are chosen as the best fit.

results of the fit can be seen in Figure 47. Except for the von Mises-Fisher model, all models fit an n_s that is 3-5 times higher than the actual number of injected signal events. The von Mises-Fisher model is closest to $n_s = 81$, although it still deviates significantly. Using the SplineMPE and KDE method described in Section 3.6, n_s is fitted almost exactly at the truth. This already indicates that the NRE methods have problems in filtering out the actual signal events from the many background events.

Lastly, we compute the combined TS at every grid point with the fitted number of signal events n_s. The results can be seen in Figure 48.

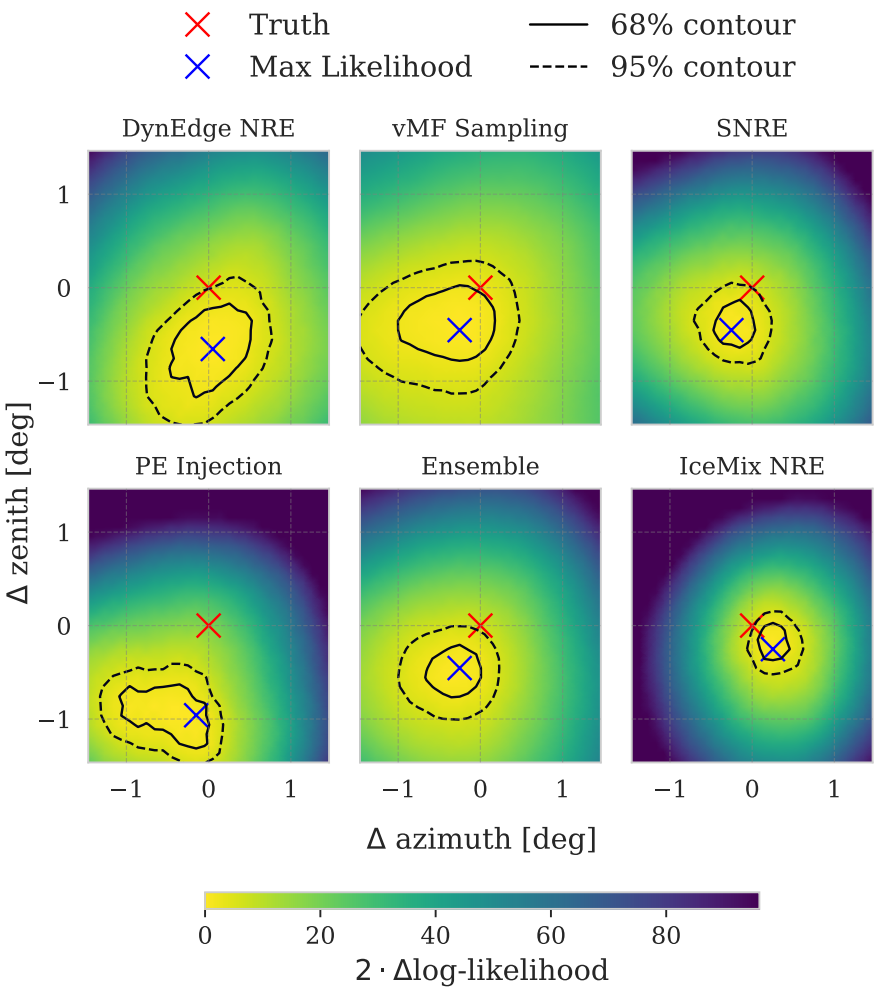

Figure 48 Joint contours for all events in our dataset. The true source location is marked with a cross at $ra = 60^\circ$ and $\delta = 15^\circ$.

The results show contour lines close to the source with shapes and sizes that can be expected from point source analyses. However, the maximum likelihood estimates seem to be slightly off center. To better understand these results, we isolated the signal events and computed their contribution to the likelihood plots, as shown in Figure 49.

In this signal-only scenario, which would correspond to an analysis with a perfectly uniform background, we can clearly identify the signal around the truth. However, in our full dataset analysis, the background events induce bias that washes out a lot of the signal contribution.

In comparison, the contours of the traditional method based on SplineMPE and KDEs can be seen in Figure 50. Although the truth lies just outside the 68% contour, this method performs best in estimating n_s as well as discriminating signal and background events.

Several factors may explain the underperformance of our NRE model. First, the angular resolution may simply be insufficient to identify the relatively small number of signal events

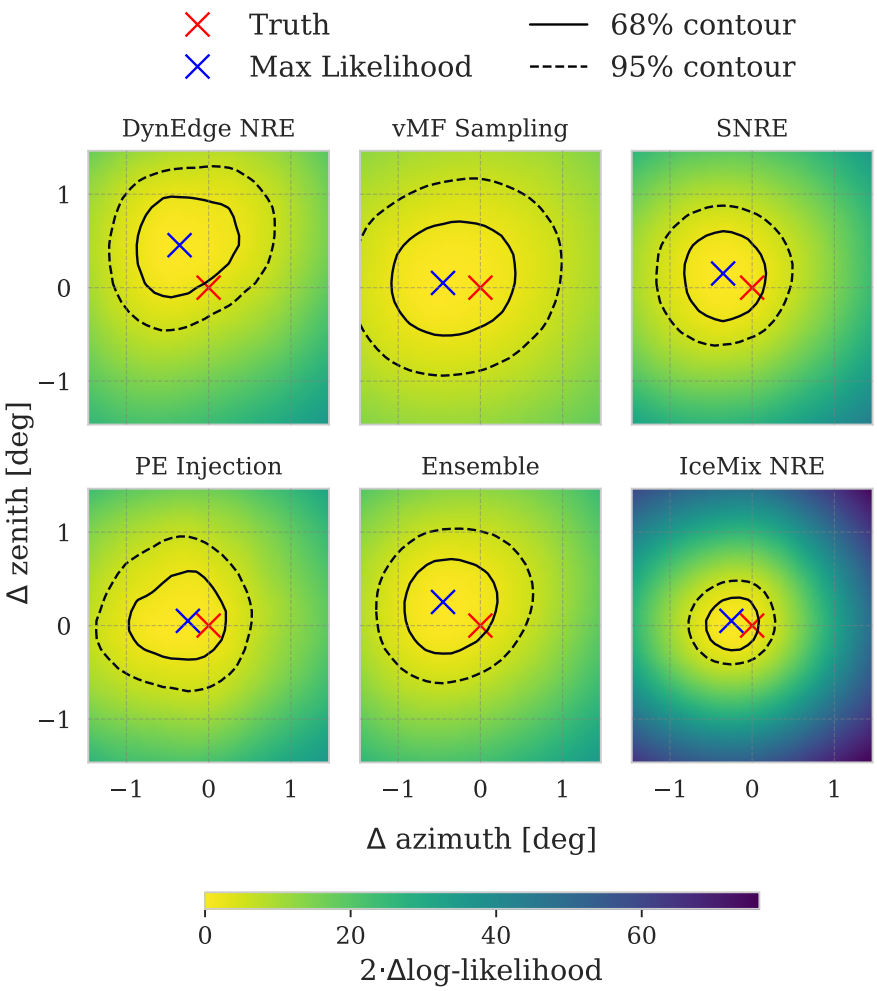

Figure 49 Combined delta log-likelihood plots for only the injected signal events. This represents how the analysis would appear with a perfectly uniform background.

within the large dataset. The angular uncertainty of our model is significantly higher than that of SplineMPE, resulting in less confident predictions and therefore lower signal-over-background ratios for signal events. This uncertainty may prevent our model from achieving the high confidence levels necessary for effective point source identification. The absence of energy information in our analysis represents another potential limitation. Although NGC 1068 has an energy spectrum similar to the atmospheric background - meaning the energy contribution would not be substantial - including energy PDFs could still improve discrimination between signal and background.

Additionally, the artifact in the background inference, which was cut out using the circular cut around the source, might still have introduced a bias into this analysis. The source of the artifact and the influence should be investigated for further improvement of the analysis.

These findings leave room for further investigation. The NRE method itself works in principle, as demonstrated in this thesis. However, it is currently not suitable as an alternative to

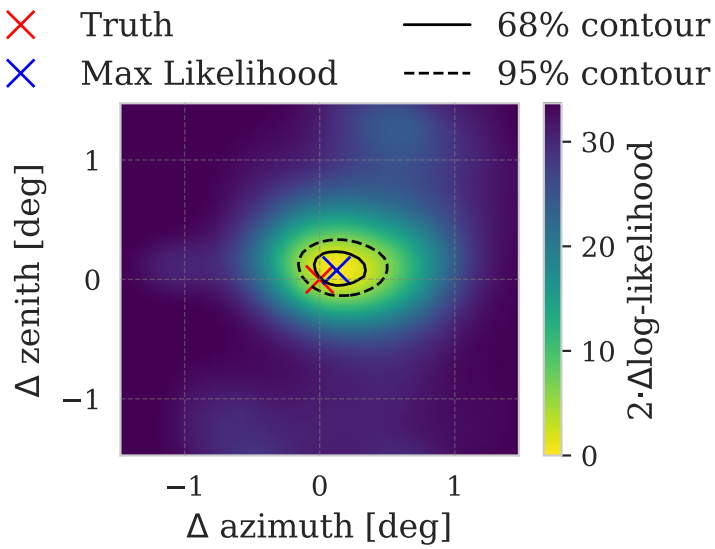

Figure 50 The delta TS plot with the corresponding Wilks contours as given by the traditional SplineMPE approach.

traditional point source analysis techniques because its resolution is substantially worse than that of SplineMPE. The general approach demonstrates theoretical soundness, but the implementation, particularly the feature extraction component, requires significant refinement. We anticipate that continued development within the GraphNet framework for neutrino astronomy, coupled with advancements in deep learning methodologies, will ultimately yield directional resolutions that meet or exceed those achieved by SplineMPE. Such improvements would make NRE a viable option for point source analyses.

10.2. Alerts

Another possible application of the NRE approach is in the context of alert events. These are typically high-energy track events with energies above $\sim 100\,\text{TeV}$ that IceCube's data processing pipeline immediately flags as having a high probability of astrophysical origin. When such events are detected, IceCube issues alerts to other telescopes in multi-messenger astronomy, enabling them to rapidly point their instruments toward the region of interest and measure potential coincident signals in their respective multi-messenger regimes.

This application demands a fast and reliable method for determining the direction of the neutrino with well-defined confidence regions. An NRE model could potentially provide the necessary speed and reliability for this application. To investigate whether the method would be suitable for alert events, we further looked at high energy events specifically. For that task, the IceMix NRE model was chosen, because it was trained on events with all pulse counts, i.e. also the very high-energy events which we would expect to be alert events.

First, the coverage properties of our model for 100,000 events with energies above 100 TeV are analyzed. Being able to quickly give a region of interest with reliable confidence contours is the main task for this application.

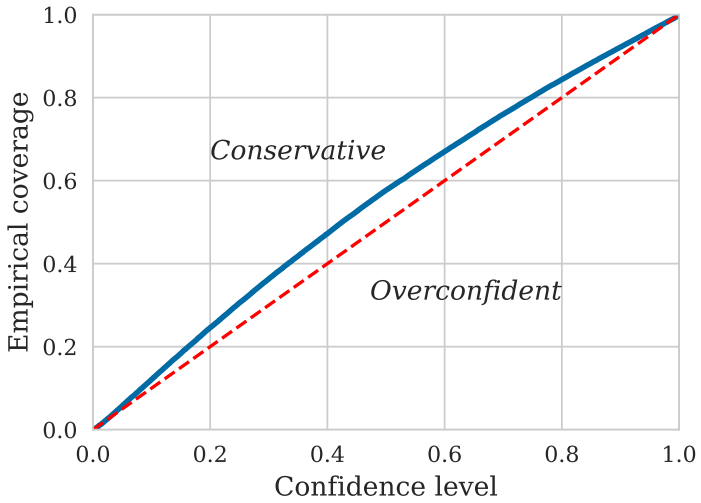

Figure 51 Coverage curves for high-energy (>100 TeV) events using the IceMix model. The coverage is evaluated based on 100,000 evaluation events. Ideal coverage, corresponding to the χ^2 distribution with 2 degrees of freedom, is given by the dashed line.

Figure 51 shows the coverage curve for these high-energy events. The model shows a slight underconfidence, which means the given contours are slightly bigger than ideal contours. This conservative behavior, while not ideal, is not necessarily problematic from a scientific standpoint, as it is generally better to be conservative than overconfident when reporting confidence regions.

One possible explanation is that the model was not explicitly trained on these high-energy

events. Most of the training events had energies lower than 100 TeV, which often times offer less rich directional information and therefore require larger contours. Overall though, the IceMix model might be a good candidate for identifying alert events and their corresponding regions.

To illustrate the capabilities of our model, three example alert events are examined to compare the model's confidence with those provided by IceCube.

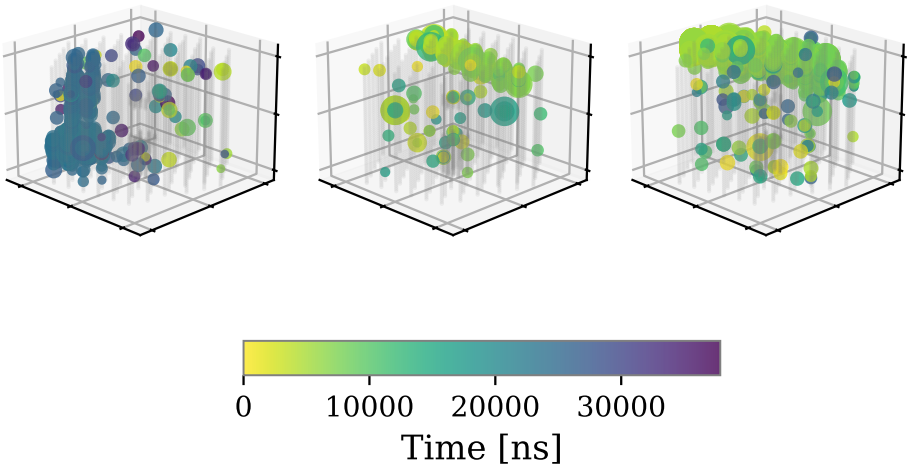

Figure 52 Topologies of the three alert events examined in this chapter. The spheres represent measured charges at individual DOMs, with size proportional to charge magnitude and color indicating detection time. The events, identified by their respective EventIDs 41853263, 56963417, and 62872761, were recorded in early 2013.

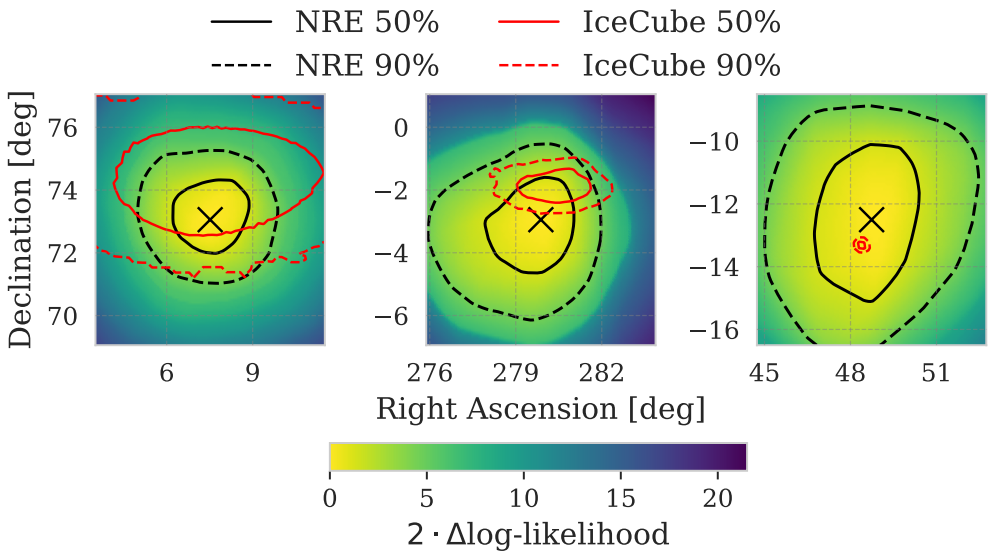

Figure 53 Comparison of likelihood regions for example alert events. Again, the contour levels for the IceMix model are based on the χ^2 distribution according to Wilks' theorem.

Figure 53 shows a comparison between official IceCube likelihood regions given by the

ICECAT-1[1] catalog and those generated by our model for selected alert events. The likelihood regions given in the ICECAT-1 catalog are follow-up reconstructions using a maximum-likelihood-based scan of the entire sky, which usually takes about 1-3 hours [49]. Our model successfully identifies the same directional region as those published by IceCube. These three events however show that for some events, IceCube is able to find much smaller alert regions, while the sizes for the IceMix NRE model are all similar. For events like the one shown in the left plot however, IceMix shows more refined contours than IceCube. It should be noted that the NRE model gives conservative estimates. Therefore, it is demonstrated that the approach can generate reliable contours for high-energy alert events and might find an application within this task.

The speed of inference is a critical factor for alert applications. While our method shows promise in terms of accuracy, its practical utility depends on whether it can deliver results within the time constraints of the alert system. In our trials, the inference only takes a few seconds on appropriate hardware, but this needs to be verified under conditions that mimic the real operational environment, considering factors such as available hardware and any necessary data transformations.

Further studies could substantially enhance the applicability. First, developing models specifically trained on alert events would create specialized systems optimized for these type of events. Alert events have distinct characteristics compared to the broader neutrino event population, including higher energies and better-defined tracks. A model trained exclusively on these events could likely achieve better resolution and more accurate confidence regions than our current general-purpose model.

Second, comprehensive hardware integration testing would provide crucial insights into real-world performance. The models should be tested on the actual hardware used in IceCube's real-time processing pipeline to provide realistic estimates of inference speed under operational conditions. This testing would need to account for the entire processing pipeline, including data preprocessing, model inference, and post-processing of results to generate the final alert information. Optimizing each step of this pipeline for the specific hardware available at the IceCube facility would ensure that the theoretical advantages of our approach translate into practical benefits in the alert system. Additionally, it could help finding an optimal balance between model performance and inference speed.

Further research into this direction could help develop the proof-of-concept approach of this thesis into a potentially useful tool that might complement IceCube's existing capabilities in giving alerts to the multi-messenger community.

[1] The ICECAT-1 database can be found **here.**

11. Conclusion

This thesis has provided an in-depth exploration of methodologies for improving point source analyses in neutrino astronomy, with a focus on leveraging modern ML techniques. The IceCube Neutrino Observatory, the world's largest neutrino detector, served as the context for this investigation. Central to the study was the challenge of approximating directional likelihoods necessary for hypothesis testing in point source analyses, which are traditionally computationally intractable. Two new approaches were presented: NRE using the likelihood ratio trick and NPE employing CNFs.

The proposed methods bypass traditional steps such as reconstructing point estimates, fitting KDEs, and approximating uncertainties. By directly estimating likelihoods from raw event-level data, these approaches aim to simplify and enhance the analysis pipeline. A general architecture was developed, utilizing a pre-trained DynEdge backbone for feature extraction from event data, followed by neural networks tasked with either ratio or posterior estimation.

For NRE, the neural network was trained to distinguish between correlated and uncorrelated pairs of observables and directions. While classification performance was near perfect and coverage closely aligned with ideal expectations, the angular resolution achieved remained slightly inferior to that of the pre-trained backbone. Various strategies were explored to improve resolution, including sampling from von Mises-Fisher distributions to break correlations, sequential model refinement, explicit injection of point estimates alongside latent representations, and ensemble modeling. Although these methods enhanced predictions, none surpassed the resolution achieved by the pre-trained feature extractor. This limitation was further corroborated by experiments with the IceMix model, which demonstrated that better feature extraction directly leads to improved resolution.

To match or exceed the resolution of traditional methods, advancements in feature extraction models are required. Current architectures struggle with events containing large numbers of pulses, indicating that they fail to capture complete event information. Research into improved data representations - such as employing autoencoders or contrastive methods - may yield low-dimensional embeddings that retain comprehensive event details and address these shortcomings.

The NPE approach also showed promise. By training CNFs through FM, Bayesian confidence regions closely matched likelihood contours, suggesting successful posterior approximation. However, inference time remains a significant bottleneck due to the computational demands of integrating over large neural networks representing time-dependent vector fields. While this method holds potential for future applications - especially if lighter network architectures or faster ODE solvers become available - its current computational cost limits practical use.

Two applications of NRE models were explored: point source analysis and multi-messenger alerts. For point source analysis, pseudodata from an NGC 1068-like source was used to construct combined likelihood contours consistent with expectations for such tasks. Despite this success, background events dominated due to insufficient angular resolution and missing energy information that could aid signal-background separation. Nonetheless, this proof-of-concept demonstrated that likelihood contours can be derived directly from raw event data without intermediate steps like point estimation or uncertainty approximation.

In multi-messenger astronomy alerts, NRE models inferred likelihood regions for single events within seconds. The contours generated showed compatibility with the official IceCube alert contours given in the ICECAT-1 catalog. However, the IceCube alert contours displayed considerably greater variability in their spatial extent compared to the NRE-derived contours. Improvements in training methodology and further testing under real-time constraints are necessary to evaluate their viability for this application.

In summary, this thesis has laid the groundwork for integrating deep learning techniques into neutrino astronomy workflows by demonstrating their ability to estimate likelihoods directly from raw data while circumventing traditional limitations. Although challenges remain - particularly in feature extraction and computational efficiency - the findings represent a significant step toward more streamlined and scalable approaches for point source analysis and multi-messenger astronomy alerts. Future work should focus on refining feature extraction models and exploring faster inference techniques to fully realize the potential of these methods.

A. Results for NRE model with injected SplineMPE Point Estimate

A.1. Classification Performance

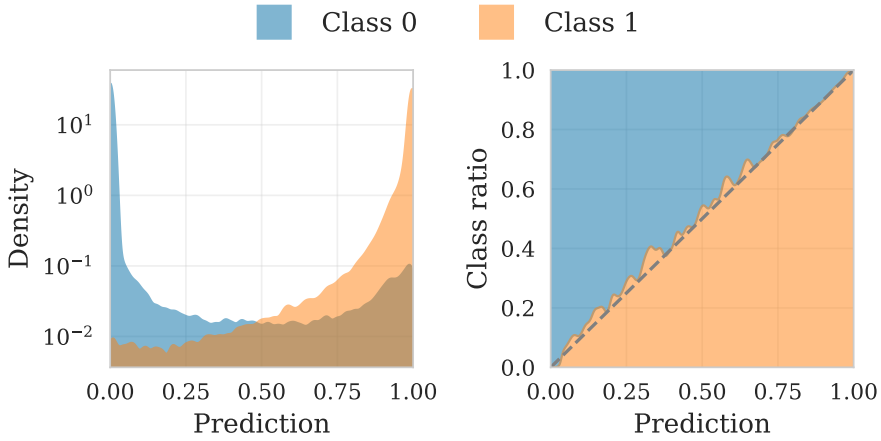

Figure 54 Classification performance of the NRE with injected SplineMPE PE model. **Left:** Output distributions of the classifier showing predictions for true positive (blue) and true negative (red) classes. **Right:** Calibration curve showing the ratio of true positive examples to all examples as a function of prediction score.

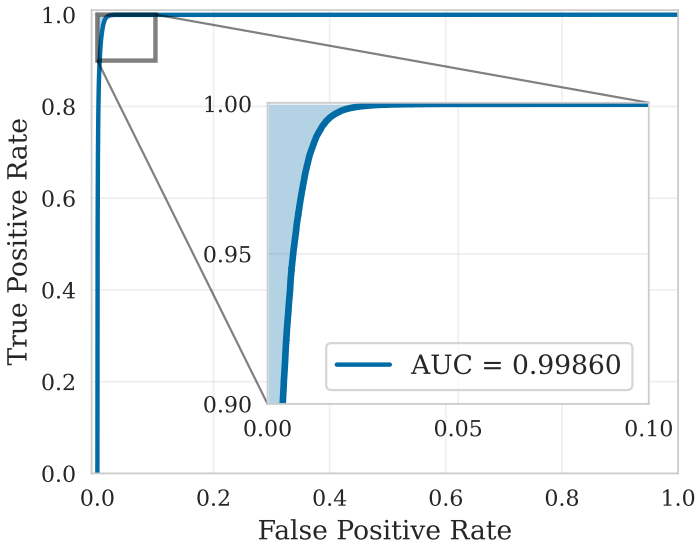

Figure 55 ROC curve for the NRE model with injected SplineMPE PE. The curve plots the TPR against the FPR at various threshold settings.

A.2. Likelihood Approximation Performance

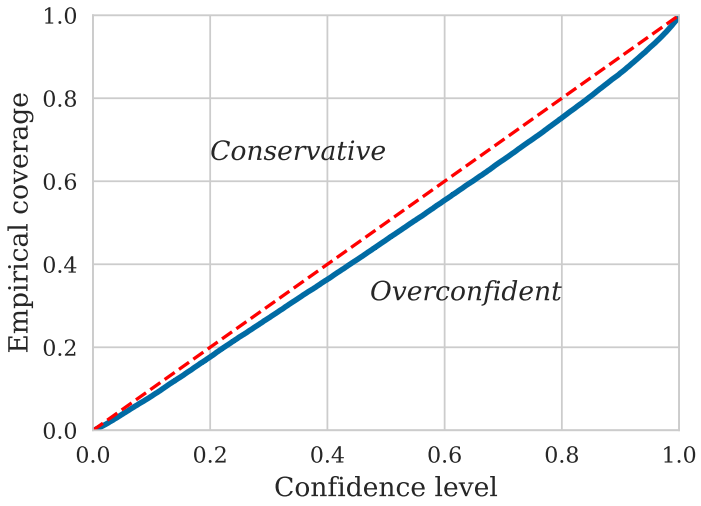

Figure 56 Coverage plot for the NRE model with injected SplineMPE PE. The coverage is evaluated based on 100,000 evaluation events. Ideal coverage, corresponding to the χ^2 distribution with 2 degrees of freedom, is given by the dashed line.

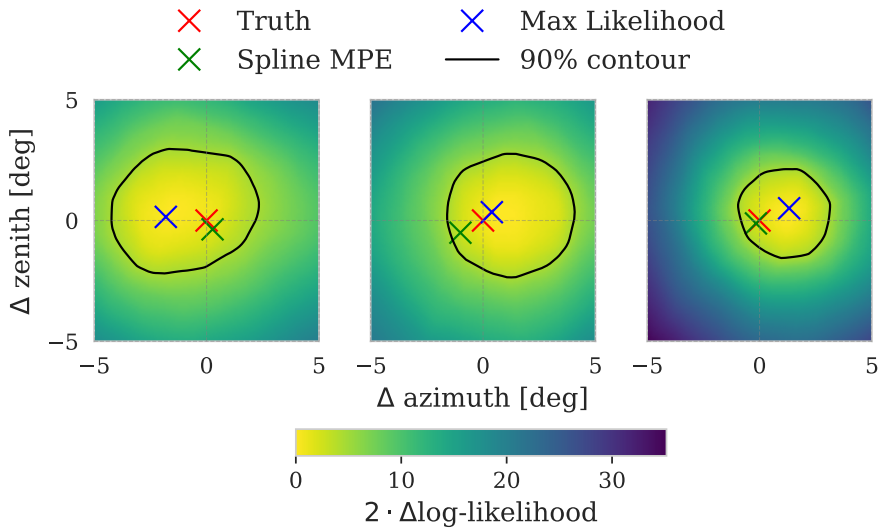

Figure 57 A grid scan of the likelihood for three example events from the evaluation dataset. Here, two times the delta log-likelihood is taken between the maximum likelihood and the likelihood at that given direction. The 90% contour line shows the level where the cumulative χ^2 distribution with 2 degrees of freedom reaches 90%. The red, green and blue marker show the true direction, the reconstructed direction from SplineMPE and the maximum likelihood direction from this model respectively.

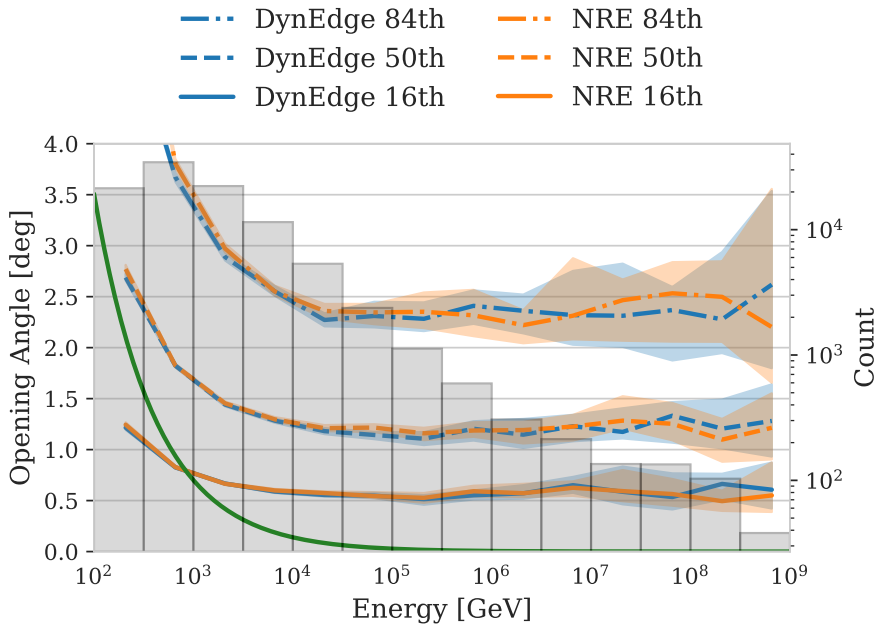

Figure 58 Performance of the NRE model with injected SplineMPE PE in terms of opening angle between point prediction and truth. This plot shows the 16th, 50th, and 84th percentile for each energy bin. In the background, the grey histogram represents the events being evaluated.

List of Abbreviations

ABC Approximate Bayesian Computation
AGN Active Galactic Nucleus
AI Artificial Intelligence
AUC Area Under Curve

BDT Boosted Decision Tree

CC charged-current
CMB Cosmic Microwave Background
CNF Continuous Normalizing Flow
CNN Convolutional Neural Network

DNF Discrete Normalizing Flow
DNN Dense Neural Network
DOM Digital Optical Module

FM Flow Matching
FM4AR Flow Matching for Atmospheric Retrieval
FPR False Positive Rate

GNN Graph Neural Network
GRU Gated Recurrent Unit
GZK Greisen Ztsepin Kuzmin

KDE Kernel Density Estimation

LSTM Long Short-Term Memory

ML Machine Learning
MLP Multilayer Perceptron
MSE Mean Squared Error

NC neutral-current
NPE Neural Posterior Estimation
NRE Neural Ratio Estimation

ODE Ordinary Differential Equation

PDF Probability Density Function
PE Point Estimate
PLEnuM Planetary Neutrino Monitoring System
PMT Photomultiplier Tube

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

TPR True Positive Rate

TS Test Statistic

List of Figures

Bibliography

[1] B. Povh, et al., Teilchen und Kerne, 9th Edition, Springer, 2014.

[2] D. O. Caldwell, Current Aspects of Neutrino Physics, 1st Edition, Springer, 2001.

[3] J. Cao, M. He, Neutrino oscillation: discovery and perspectives, Science Bulletin 61 (1) (2016) 48–51. doi:https://doi.org/10.1007/s11434-015-0969-7.
URL https://www.sciencedirect.com/science/article/pii/S2095927316302286

[4] S. M. Bilenky, The history of neutrino oscillations, Physica Scripta T121 (2005) 17–22. doi:10.1088/0031-8949/2005/t121/001.
URL https://doi.org/10.1088%2F0031-8949%2F2005%2Ft121%2F001

[5] Y. Fukuda, et al., Evidence for oscillation of atmospheric neutrinos, Physical Review Letters 81 (8) (1998) 1562–1567. doi:10.1103/physrevlett.81.1562.
URL http://dx.doi.org/10.1103/PhysRevLett.81.1562

[6] I. Bartos, M. Kowalski, Multimessenger Astronomy, 2399-2891, IOP Publishing, 2017. doi:10.1088/978-0-7503-1369-8.
URL https://dx.doi.org/10.1088/978-0-7503-1369-8

[7] E. Resconi, et al., Cosmic Rays and Particle Physics, 2nd Edition, Cambridge University Press, 2016. doi:10.1017/CBO9781139192194.

[8] K. Scholberg, Supernova neutrino detection, Annual Review of Nuclear and Particle Science 62 (1) (2012) 81–103. arXiv:https://doi.org/10.1146/annurev-nucl-102711-095006, doi:10.1146/annurev-nucl-102711-095006.
URL https://doi.org/10.1146/annurev-nucl-102711-095006

[9] M. Spurio, Particles and Astrophysics: A Multi-Messenger Approach, Astronomy and Astrophysics Library, Springer International Publishing, 2015.
URL https://doi.org/10.1007/978-3-319-08051-2

[10] E. Vitagliano, I. Tamborra, G. Raffelt, Grand unified neutrino spectrum at earth: Sources and spectral components, Reviews of Modern Physics 92 (4) (Dec. 2020). doi:10.1103/revmodphys.92.045006.
URL http://dx.doi.org/10.1103/RevModPhys.92.045006

[11] T. Kajita, Atmospheric neutrinos and discovery of neutrino oscillations, Proc. Japan Acad. B 86 (2010) 303–321. doi:10.2183/pjab.86.303.

[12] J. Stettner, Measurement of the diffuse astrophysical muon-neutrino spectrum with ten years of icecube data (2019). doi:10.48550/ARXIV.1908.09551.
URL https://arxiv.org/abs/1908.09551

[13] M. Aartsen, et al., Neutrino emission from the direction of the blazar TXS 0506+056 prior to the IceCube-170922a alert, Science 361 (6398) (2018) 147–151. doi:10.1126/science.aat2890.
URL https://doi.org/10.1126%2Fscience.aat2890

[14] L. Yacobi, et al., IMPLICATION OF THE NON-DETECTION OF GZK NEUTRINOS, The Astrophysical Journal 823 (2) (2016) 89. doi:10.3847/0004-637x/823/2/89.
URL https://doi.org/10.3847/0004-637x/823/2/89

[15] A. A. Watson, The discovery of cherenkov radiation and its use in the detection of extensive air showers, Nuclear Physics B - Proceedings Supplements 212-213 (2011) 13–19. doi:10.1016/j.nuclphysbps.2011.03.003.
URL https://doi.org/10.1016%2Fj.nuclphysbps.2011.03.003

[16] L. J. Schumacher, et al., PLEnuM: A global and distributed monitoring system of high-energy astrophysical neutrinos, in: Proceedings of 37th International Cosmic Ray Conference — PoS(ICRC2021), Sissa Medialab, 2021. doi:10.22323/1.395.1185.
URL https://doi.org/10.22323%2F1.395.1185

[17] F. Halzen, A. Kheirandish, Icecube and high-energy cosmic neutrinos (2022). arXiv:2202.00694.
URL https://arxiv.org/abs/2202.00694

[18] N. Kurahashi, Highlights from the icecube neutrino observatory (2023). arXiv:2310.12840.
URL https://arxiv.org/abs/2310.12840

[19] M. Aartsen, et al., The icecube neutrino observatory: instrumentation and online systems, Journal of Instrumentation 12 (03) (2017) P03012–P03012. doi:10.1088/1748-0221/12/03/p03012.
URL http://dx.doi.org/10.1088/1748-0221/12/03/P03012

[20] A. Ishihara, The icecube upgrade – design and science goals (2019). arXiv:1908.09441.
URL https://arxiv.org/abs/1908.09441

[21] A. Ishihara, The next generation neutrino telescope: Icecube-gen2 (2023). arXiv:2308.09427.
URL https://arxiv.org/abs/2308.09427

[22] R. Abbasi, et al., Characterization of the astrophysical diffuse neutrino flux using starting track events in icecube, Physical Review D 110 (2) (Jul. 2024). doi:10.1103/physrevd.110.022001.
URL http://dx.doi.org/10.1103/PhysRevD.110.022001

[23] C. Bellenghi, The emergence of a new sky: First associations of icecube high-energy neutrinos with active galactic nuclei, Ph.D. thesis, Technische Universität München (2024).
URL https://mediatum.ub.tum.de/1749345

[24] J. G. Learned, K. Mannheim, High-energy neutrino astrophysics, Annual Review of Nuclear and Particle Science 50 (Volume 50, 2000) (2000) 679–749. doi:https://doi.org/10.1146/annurev.nucl.50.1.679.
URL https://www.annualreviews.org/content/journals/10.1146/annurev.nucl.50.1.679

[25] N. Whitehorn, et al., Penalized splines for smooth representation of high-dimensional monte carlo datasets, Computer Physics Communications 184 (9) (2013) 2214–2220. doi:https://doi.org/10.1016/j.cpc.2013.04.008.
URL https://www.sciencedirect.com/science/article/pii/S0010465513001434

[26] K. Cranmer, et al., The frontier of simulation-based inference, Proceedings of the National Academy of Sciences 117 (48) (2020) 30055–30062. doi:10.1073/pnas.1912789117.
URL http://dx.doi.org/10.1073/pnas.1912789117

[27] R. Barlow, Extended maximum likelihood, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 297 (3) (1990) 496–506. doi:https://doi.org/10.1016/0168-9002(90)91334-8.
URL https://www.sciencedirect.com/science/article/pii/0168900290913348

[28] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, Berlin, Heidelberg, 2006.

[29] D. A. Roberts, S. Yaida, B. Hanin, The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks, Cambridge University Press,

2022. doi:10.1017/9781009023405.
URL http://dx.doi.org/10.1017/9781009023405

[30] L. L. Lu Lu, Y. S. Yeonjong Shin, Y. S. Yanhui Su, G. E. K. George Em Karniadakis, Dying relu and initialization: Theory and numerical examples, Communications in Computational Physics 28 (5) (2020) 1671–1706. doi:10.4208/cicp.oa-2020-0165.
URL http://dx.doi.org/10.4208/cicp.OA-2020-0165

[31] K. O'Shea, R. Nash, An introduction to convolutional neural networks (2015). arXiv:1511.08458.
URL https://arxiv.org/abs/1511.08458

[32] R. Abbasi, et al., Graph Neural Networks for low-energy event classification & reconstruction in IceCube, JINST 17 (11) (2022) P11003. arXiv:2209.03042, doi:10.1088/1748-0221/17/11/P11003.

[33] J. M. Jumper, et al., Highly accurate protein structure prediction with alphafold, Nature 596 (2021) 583 – 589.
URL https://doi.org/10.1038/s41586-021-03819-2

[34] R. M. Schmidt, Recurrent neural networks (rnns): A gentle introduction and overview (2019). arXiv:1912.05911.
URL https://arxiv.org/abs/1912.05911

[35] G. Yenduri, et al., Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions (2023). arXiv:2305.10435.
URL https://arxiv.org/abs/2305.10435

[36] R. F. Ørsøe, et al., Graphnet 2.0 – a deep learning library for neutrino telescopes (2025). arXiv:2501.03817.
URL https://arxiv.org/abs/2501.03817

[37] R. Ørsøe, et al., IceCube – Neutrinos in Deep Ice: The top 3 solutions from the public Kaggle competition, Eur. Phys. J. C 84 (6) (2024) 646. arXiv:2310.15674, doi:10.1140/epjc/s10052-024-12977-2.

[38] S. Kumar, Y. Tsvetkov, Von mises-fisher loss for training sequence to sequence models with continuous outputs, in: International Conference on Learning Representations, 2019.
URL https://openreview.net/forum?id=rJlDnoA5Y7

[39] G. Papamakarios, I. Murray, Fast ϵ-free inference of simulation models with bayesian conditional density estimation (2018). arXiv:1605.06376.
URL https://arxiv.org/abs/1605.06376

[40] P. Eller, et al., A flexible event reconstruction based on machine learning and likelihood principles, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 1048 (2023) 168011. doi:10.1016/j.nima.2023.168011.
URL http://dx.doi.org/10.1016/j.nima.2023.168011

[41] G. Louppe, et al., Likelihood-free mcmc with amortized approximate ratio estimators (2020). arXiv:1903.04057.
URL https://arxiv.org/abs/1903.04057

[42] G. Louppe, et al., Neural posterior estimation for exoplanetary atmospheric retrieval, A I& A 672 (2023) A147. doi:10.1051/0004-6361/202245263.
URL https://doi.org/10.1051/0004-6361/202245263

[43] T. D. Gebhard, et al., Flow matching for atmospheric retrieval of exoplanets: Where reliability meets adaptive noise levels, Astronomy I& Astrophysics 693 (2024) A42. doi:10.1051/0004-6361/202451861.
URL http://dx.doi.org/10.1051/0004-6361/202451861

[44] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, M. Le, Flow matching for generative modeling (2023). arXiv:2210.02747.
URL https://arxiv.org/abs/2210.02747

[45] U. Zrimšek, E. Štrumbelj, Quantifying uncertainty: All we need is the bootstrap? (2024). arXiv:2403.20182.
URL https://arxiv.org/abs/2403.20182

[46] D. S. Wilks, On the combination of forecast probabilities for consecutive precipitation periods, Weather and Forecasting 5 (4) (1990) 640 – 650. doi:10.1175/1520-0434(1990)005<0640:OTCOFP>2.0.CO;2.
URL https://journals.ametsoc.org/view/journals/wefo/5/4/1520-0434_1990_005_0640_otcofp_2_0_co_2.xml

[47] Y. Lipman, et al., Flow matching for generative modeling (2023). arXiv:2210.02747.
URL https://arxiv.org/abs/2210.02747

[48] R. Abbasi, et al., Evidence for neutrino emission from the nearby active galaxy ngc 1068,

Science 378 (6619) (2022) 538–543. doi:10.1126/science.abg3395.
URL http://dx.doi.org/10.1126/science.abg3395

[49] R. Abbasi, et al., Icecat-1: The icecube event catalog of alert tracks, The Astrophysical Journal Supplement Series 269 (1) (2023) 25. doi:10.3847/1538-4365/acfa95.
URL https://dx.doi.org/10.3847/1538-4365/acfa95