GraphNeT 2.0
A Deep Learning Library for Neutrino Telescopes

Rasmus F. Ørsøe
Technical University of Munich
`rasmus.orsoe@tum.de`

Aske Rosted
ICEHAP - Chiba University
`askerosted@hepburn.s.chiba-u.ac.jp`

On behalf of
The GraphNeT Team [*]

Abstract

Neutrino telescopes, an extension of traditional multiwavelength astronomy, provide a complementary view of the universe using neutrinos. Differences in detector geometry and detection medium mean that improvements to reconstruction techniques made at one experiment are not readily applicable to another. Recently, deep learning has been shown to improve prediction speed and accuracy and offer indifference to detector geometry and detection medium, providing a unique opportunity for collaboration. This work introduces GraphNeT 2.0, an open-source, detector-agnostic deep learning library for neutrino telescopes and related experiments. GraphNeT enables inter-experimental collaboration on the use and development of advanced methods based on major deep learning paradigms like transformers, normalizing flows, graph neural networks, and more.

1 Introduction

In order to minimize the dominant background of atmospheric muons, neutrino telescopes are constructed in exotic sub-surface locations, such as at the bottom of the Mediterranean Sea or deep into the antarctic ice [1], [2]. They span volumes up to the cubic-kilometer scale with instrumentation,

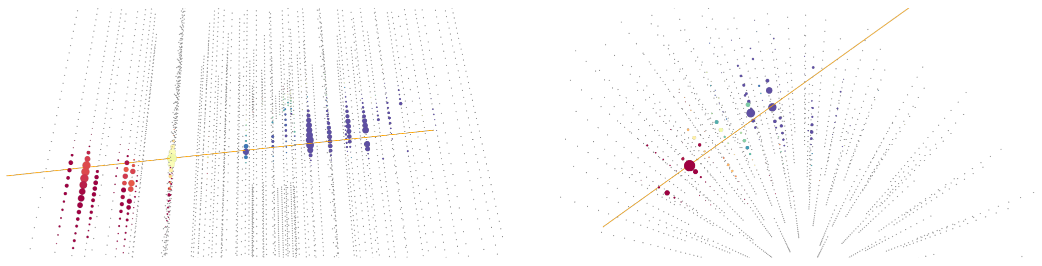
Figure 1: Two simulated neutrino events in two different detectors with different detector geometries and detection mediums. Left) A simulated neutrino event from IceCube, a neutrino telescope installed in the ice at the south pole [1]. Right) A simulated neutrino interaction in a detector installed in water with a geometry similar to KM3NeT-ARCA [2].

making them among the largest human-made objects by volume. Within these detector volumes,

[*]Many individuals have made direct contributions to GraphNeT. A full list of contributors can be found here.

Optical Modules (OMs) with one or more Photo-Multiplier-Tube (PMT) are often placed on vertical lines arranged in irregular geometries spanning the detector volume. These OMs detect Cherenkov radiation emitted by charged particles induced by neutrino interactions.

Despite their differences in geometry and detection medium, the detection principle and resulting low-level observations are virtually identical for neutrino telescopes. The low-level data consists of triggered events representing individual neutrino interaction candidates. Each event is a geometric time series, where every step represents a PMT at a given location x,y,z that observed Cherenkov photons at time t. The topology and length of the sequences depend primarily on the neutrino energy, detector geometry, and number of OMs. The length of the sequences may range from a few photons to upwards of a million. Illustration of events for IceCube and a water-based detector can be seen in Figure 1.

Inference of the physical properties of a neutrino inducing a detector response, such as its energy and direction, is a central task of most experiments. Maximum-likelihood estimators (MLEs) have predominantly been used for this task. However, the reconstruction likelihood is often intractable, and the MLEs, therefore, rely on complex approximations that are both time-consuming and require assumptions on detector geometry and detection medium, making cross-experimental collaboration impractical [3], [4].

In recent times, it has been shown that phrasing neutrino event reconstruction and classification as supervised learning tasks can provide both superior accuracy and reconstruction speeds, without relying on assumptions on either geometry or detection medium [5]–[8], and has enabled high profile results in the field [9], [10]. The indifference to geometry and detection medium in deep learning methods provides an opportunity for various neutrino experiments to collaborate on using and developing deep learning-based techniques, but incompatible code bases and closed data policies have made it challenging.

2 GraphNeT

GraphNeT [11] is an open-source deep learning library written in Python for use by neutrino telescopes and related experiments. It is built as a purpose-specific extension of popular deep learning libraries, such as PyTorch [12], PyTorch Geometric [13], Lightning [14] and employs code conventions such as Black [15] and type hints [16] to harmonize readability of code from many contributors. It is designed to enable re-usability of models developed for one telescope to another and provides a way for physics domain experts to apply complex methods to their physics analyses without being an expert in them. Reconstruction tasks are phrased in a way that is accessible to

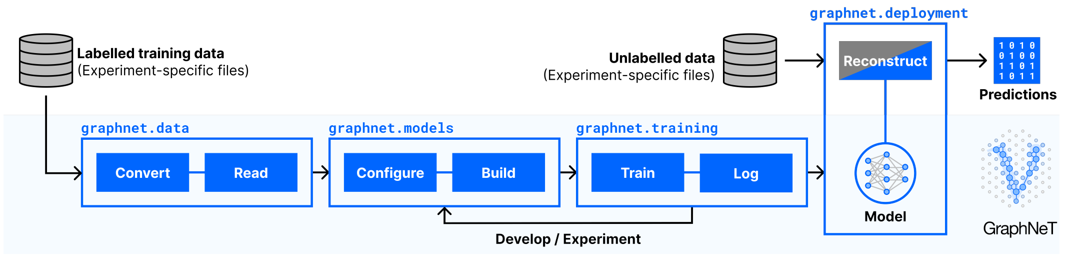

Figure 2: High-level overview of the different library components of GraphNeT. The library's functionality extends from converting experiment-specific files to configuring and training deep learning models. Lastly, parts of GraphNeT are dedicated to aiding users in the deployment of trained models.

members of the general deep learning community, which may contribute with techniques we, in return, may use for physics. As seen in the high-level overview of GraphNeT in Figure 2, the library contains functionality for converting experiment-specific files to formats suitable for deep learning and for designing and training models. Finally, GraphNeT contains functionality to aid users in evaluating trained models on unlabelled, experiment-specific files. The major changes introduced in this coming update of GraphNeT are support for models from most established deep learning paradigms (GraphNeT 1.0 supported just GNNs [17]) and data conversion functionality. Each of these two areas of the library is elaborated upon in the following sections.

2.1 Data conversion in GraphNeT

GraphNeT contains modularized data conversion. This optional data conversion functionality, referred to as the DataConverter henceforth, follows a reader/writer scheme, as illustrated in Figure 3. The reader is a small piece of code able to parse a single experiment-specific file and extract relevant quantities from it. The DataConverter presents the extracted data in a standardized way to the writer

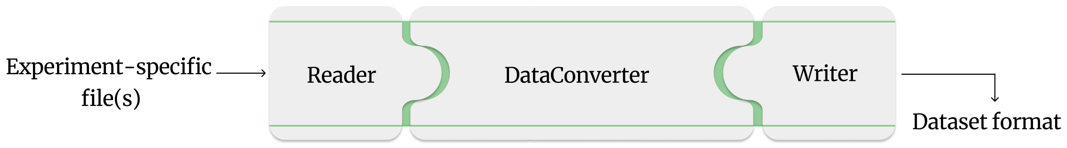

Figure 3: Illustration of the reader/writer structure of the DataConverter.

module, which saves the extracted data to a specific file format suitable for deep learning. Because experiment-specific details related to data conversion is isolated in the reader module, extending support for a new experiment only requires users to provide a new reader. Similarly, support for exporting data to new file formats requires only adding a new writer module.

Currently, GraphNeT contains readers for IceCube [1] and LiquidO [18], and writers for Apache Parquet and SQLite [19]. Both formats are accompanied with corresponding PyTorch Dataset classes [20].

2.2 Models in GraphNeT

A central idea of GraphNeT is to enable users to reuse models across collaboration boundaries and to re-purpose models to solve different problems. In order to achieve this, GraphNeT provides standards for model design that seek to compartmentalize model functionality into reusable modules, referred to as model components henceforth. The designs are self-contained, meaning the models depend only on raw observations so that collaborations can deploy them in either offline or real-time settings. In this section, we elaborate on the standards and components that make up models in GraphNeT.

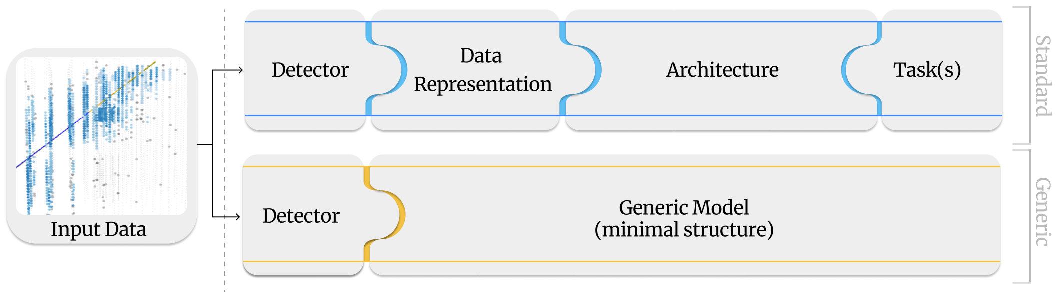

Figure 4: An illustration of model components in GraphNeT. GraphNeT allows users to reconfigure existing methods to solve new problems by abstracting a deep learning solution into reusable components.

In a typical deep learning workflow, raw data is loaded into memory and then processed into a suitable representation that is then passed to a neural network, which returns a prediction. Along the way, details specific to either the dataset or its source may linger.

In GraphNeT, this overall process is broken into 1 or more interchangeable components, depending on the method, as visualized in Figure 4. The Standard model (top Figure 4), which encompasses the vast majority of methods in GraphNeT, consists of 4 model components: Detector, Data Representation, Architecture, and Task(s).

The Detector component holds all experiment-specific details, which include column names of input features, detector geometry, and standardization functions to map raw data into a numerical range suitable for stochastic gradient descent. The Detector component is the only part of a Model with experiment-specific details, allowing the remaining components to be detector-agnostic.

The Data Representation component contains functionality for transforming raw observations into a suitable data representation on an event-by-event basis and is used directly in the data-loading procedure. The transformation from raw observations to a chosen data representation happens in real-time, parallelized by PyTorch DataLoader, and is independent of file formats. Such representations could be images, sequences, and graphs.

The Architecture represents the bulk of learnable parameters in the Model and may rely on methods from established deep learning paradigms such as graph neural networks, convolutional neural networks, and sequence-based methods such as RNNs, transformers, etc. This component receives the data in the chosen representation and produces a latent representation of the input that is passed to the Task(s).

The Task component defines the part of a model that is specific to a particular way of solving a particular deep learning task. This includes problem-specific handling of the latent representation, final activation layers, and mapping the latent representation to the target space. Most Tasks in GraphNeT act as learnable prediction heads but come with additional logic for loss function evaluations, loss weighting, and loss regularization. The Task takes the loss function as an argument, which allows users to experiment with different loss functions for the same Task.

By categorizing model functionality into the model components, users can experiment with different choices in these components for a given problem. For example, evaluating the impact on a particular task by varying the choice in Data Representation or configuring the model to function in a new experiment by changing the Detector component.

While the structure of the Standard model in Figure 4 can accommodate most deep learning applications, certain techniques may be impractical to abstract into those 4 model components. Such techniques may include auto-encoders, which rely on both an encoder and a decoder architecture, and with a particular Task (reconstructing the input given the encoding), hybrid methods that combine techniques from both deep learning and traditional MLE, etc. To accommodate these relevant methods, a second Model design, referred to as the Generic model in Figure 4, imposes only the existence of an interchangeable Detector component, which allows the method to be applicable to different experiments.

3 Recent Applications

At the time of writing, the authors are familiar with specific applications of methods in GraphNeT by the user community in at least six different experiments. These experiments range from neutrino telescopes such as IceCube [1] and KM3NeT-ARCA [2] to related experiments such as SNO+ [21], MAGIC/CTAO [22], [23], ESSνSB [24] and Liquid-O [18]. In SNO+, GNNs in GraphNeT are being applied in a dark matter search for event classification. In MAGIC, a technical study is underway using GNNs in GraphNeT to distinguish between proton, gamma, and muon events and to apply these methods to events that are detected by multiple Cherenkov telescopes in the larger CTAO network, combining multiple telescope observations into a single graph object [25]. In ESSνSB, methods in GraphNeT has been applied for fast event reconstruction [26], and for gamma/positron classification and interaction vertex reconstruction in LiquidO [27].

In the following sections, a few examples of published applications of GraphNeT are mentioned.

3.1 Noise cleaning & reconstruction for IceCube's detector extension Upgrade

IceCube Upgrade is a planned detector extension of IceCube aimed at significantly improving capabilities in the GeV energy range. It will feature new multi-PMT OMs and will more than triple the number of PMT channels. Due to the increased number of PMTs and higher levels of radioactivity in the new OMs, traditional noise-cleaning methods were insufficient, and MLE-based methods used in this energy range were incompatible with multi-PMT OMs. To address this, the events were represented as point-cloud graphs, and the noise-cleaning task was phrased as a binary node classification problem. A Graph Neural Network (GNN) nicknamed DynEdge [28] was trained to distinguish between signal- and noise-induced OM output could reduce the amount of noise by around a factor of 10 with only a minor signal loss. After cleaning, a separate instance of DynEdge was trained to provide predictions on event-level tasks for the analysis, such as neutrino energy, zenith angle, and particle/interaction-type identification[29].

3.2 Kaggle Competition "IceCube - Neutrinos in Deep Ice"

In the Kaggle competition "IceCube – Neutrinos in Deep Ice" [30], close to a thousand participants competed to develop the best reconstruction algorithm for direction reconstruction on around 140 million simulated neutrinos from IceCube. The competition metric was defined as the mean opening angle between the true and estimated direction vector computed over a large sample of events [31]. During the competition a baseline utilizing DynEdge was shared with the participants. The baseline was trained on less than 8% of the data. Many participants, including the winning solutions, took inspiration from methods used in the baseline and produced their own versions for the competition. Performance of the 1st and 2nd place solutions on up- and down-going track events can be seen in Figure 5. Here the baseline is shown in grey, and the kinematic angle between the neutrino and out-going muon, which represents the expected information limit, is added in dotted grey [32].

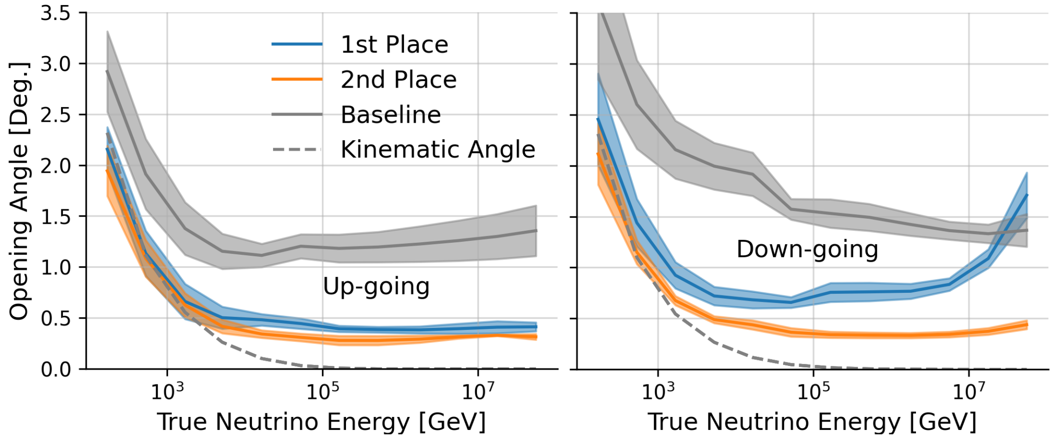

Figure 5: Primary mode of opening angle distribution vs. neutrino energy for both up- and down-going $\nu_{\mu,CC}$ events.

After the competition, the 1st and 2nd place solutions, alongside other models, were added to GraphNeT and are now available to the wider community.

4 Conclusion & Outlook

GraphNeT is an open-source deep learning library built by and for physicists working at the intersection of deep learning and neutrino physics. GraphNeT enables researchers across different neutrino experiments to develop, share, and adapt models to their work. By using popular deep learning libraries, GraphNeT makes it possible for members of the deep learning community to make meaningful contributions to the field without expertise in neutrino physics.

At the time of writing, users of GraphNeT have applied techniques from the library to problems in at least 6 different experiments. Particularly, methods in GraphNeT have outperformed traditional MLE methods on several tasks in the GeV energy range of IceCube, such as angular reconstruction and event classification [28]. Recently, methods in GraphNeT were used to remove stochastic noise induced by radioactive decays in the glass housing of OMs and to project the sensitivities of IceCube Upgrade to atmospheric neutrino oscillations [29]. In addition, GraphNeT was used by both organizers and participants in "IceCube - Neutrinos in Deep Ice"[32] and in reconstruction of neutrino energy for point source searches [33].

To address the challenge of closed-data policies for low-level data in the field, more than 100 million simulated neutrino events for at least 6 different detector geometries in both ice and water detection mediums are expected to be released at the beginning of 2025 through GraphNeT for benchmarking deep learning based techniques [34].

Acknowledgments and Disclosure of Funding

We'd like to acknowledge the work carried out by the many contributors to GraphNeT for their valuable contributions both large and small. The work carried out by Rasmus F. Ørsøe was supported by the PUNCH4NFDI consortium via DFG fund "NFDI39/1". The work by Aske Rosted is partially supported by the Institute for Advanced Academic Research of Chiba University.

References

[1] R. Abbasi, Y. Abdou, T. Abu-Zayyad, *et al.*, "The design and performance of IceCube DeepCore", *Astroparticle Physics*, vol. 35, no. 10, pp. 615–624, 2012, ISSN: 0927-6505. DOI: `https://doi.org/10.1016/j.astropartphys.2012.01.004`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0927650512000254`.

[2] S. Aiello *et al.*, "Astronomy potential of KM3NeT/ARCA", Feb. 2024. arXiv: `2402.08363 [astro-ph.HE]`.

[3] R. Abbasi *et al.*, "Low energy event reconstruction in IceCube DeepCore", *Eur. Phys. J. C*, vol. 82, no. 9, p. 807, 2022. DOI: `10.1140/epjc/s10052-022-10721-2`. arXiv: `2203.02303 [hep-ex]`.

[4] M. G. Aartsen *et al.*, "Improvement in Fast Particle Track Reconstruction with Robust Statistics", *Nucl. Instrum. Meth. A*, vol. 736, pp. 143–149, 2014. DOI: `10.1016/j.nima.2013.10.074`. arXiv: `1308.5501 [astro-ph.IM]`.

[5] R. Abbasi *et al.*, "A Convolutional Neural Network based Cascade Reconstruction for the IceCube Neutrino Observatory", *JINST*, vol. 16, P07041, 2021. DOI: `10.1088/1748-0221/16/07/P07041`. arXiv: `2101.11589 [hep-ex]`.

[6] M. Huennefeld *et al.*, "Combining Maximum-Likelihood with Deep Learning for Event Reconstruction in IceCube", *PoS*, vol. ICRC2021, p. 1065, 2021. DOI: `10.22323/1.395.1065`. arXiv: `2107.12110 [astro-ph.HE]`.

[7] C. Mo, F. Hu, L. Li, and D. Xu, "Reconstruction of Track-like Event in TRIDENT Based on a Submanifold Sparse Convolutional Network", *PoS*, vol. ICRC2023, p. 1205, 2023. DOI: `10.22323/1.444.1205`.

[8] S. Aiello *et al.*, "Event reconstruction for KM3NeT/ORCA using convolutional neural networks", *JINST*, vol. 15, no. 10, P10005, 2020. DOI: `10.1088/1748-0221/15/10/P10005`. arXiv: `2004.08254 [astro-ph.IM]`.

[9] R. Abbasi, M. Ackermann, J. Adams, *et al.*, "Evidence for neutrino emission from the nearby active galaxy NGC 1068", *Science*, vol. 378, no. 6619, pp. 538–543, 2022. DOI: `10.1126/science.abg3395`. eprint: `https://www.science.org/doi/pdf/10.1126/science.abg3395`. [Online]. Available: `https://www.science.org/doi/abs/10.1126/science.abg3395`.

[10] I. Collaboration*†, R. Abbasi, M. Ackermann, *et al.*, "Observation of high-energy neutrinos from the galactic plane", *Science*, vol. 380, no. 6652, pp. 1338–1343, 2023. DOI: `10.1126/science.adc9818`. eprint: `https://www.science.org/doi/pdf/10.1126/science.adc9818`. [Online]. Available: `https://www.science.org/doi/abs/10.1126/science.adc9818`.

[11] A. Søgaard, R. F. Ørsøe, M. Holm, *et al.*, *GraphNeT*, version 1.0.0, May 2023. DOI: `10.5281/zenodo.6720188`. [Online]. Available: `https://github.com/graphnet-team/graphnet`.

[12] J. Ansel, E. Yang, H. He, *et al.*, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation", in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, ACM, Apr. 2024. DOI: `10.1145/3620665.3640366`. [Online]. Available: `https://pytorch.org/assets/pytorch2-2.pdf`.

[13] M. Fey and J. E. Lenssen, *Fast Graph Representation Learning with PyTorch Geometric*, May 2019. [Online]. Available: `https://github.com/pyg-team/pytorch_geometric`.

[14] W. Falcon and The PyTorch Lightning team, *PyTorch Lightning*, version 1.4, Mar. 2019. DOI: `10.5281/zenodo.3828935`. [Online]. Available: `https://github.com/Lightning-AI/lightning`.

[15] Langa and contributors to Black, *Black: The uncompromising Python code formatter*. [Online]. Available: `https://github.com/psf/black`.

[16] G. van Rossum, B. Warsaw, and N. Coghlan, "Style guide for Python code", PEP 8, 2001. [Online]. Available: `https://www.python.org/dev/peps/pep-0008/`.

[17] A. Søgaard, R. F. Ørsøe, L. Bozianu, *et al.*, "Graphnet: Graph neural networks for neutrino telescope event reconstruction", 2022. arXiv: `2210.12194 [astro-ph.IM]`.

[18] A. Cabrera, "LiquidO: First Opaque Detector for $\beta\beta$ Decay?", *PoS*, vol. NOW2018, A. Marrone, A. Mirizzi, and D. Montanino, Eds., p. 028, 2019. DOI: `10.22323/1.337.0028`.

[19] R. D. Hipp, *SQLite*, version 3.31.1, 2020. [Online]. Available: `https://www.sqlite.org/index.html`.

[20] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in PyTorch", 2017.

[21] V. Albanese *et al.*, "The SNO+ experiment", *JINST*, vol. 16, no. 08, P08059, 2021. DOI: `10.1088/1748-0221/16/08/P08059`. arXiv: `2104.11687 [physics.ins-det]`.

[22] C. Bigongiari, "The magic telescope", *PoS*, vol. HEP2005, G. Barreira, Ed., p. 020, 2006. DOI: `10.22323/1.021.0020`. arXiv: `astro-ph/0512184`.

[23] W. Hofmann and R. Zanin, "The Cherenkov Telescope Array", May 2023. arXiv: `2305.12888 [astro-ph.IM]`.

[24] A. Alekou *et al.*, "The European Spallation Source neutrino super-beam conceptual design report", *Eur. Phys. J. ST*, vol. 231, no. 21, pp. 3779–3955, 2022, [Erratum: Eur.Phys.J.ST 232, 15–16 (2023)]. DOI: `10.1140/epjs/s11734-022-00664-w`. arXiv: `2206.01208 [hep-ex]`.

[25] J. Green, *Deep learning for cherenkov event reconstruction*, Talk presented at Erice, Slides available at: `https://slides.com/astrojarred/green-jarred-erice-2024#`, 2024.

[26] K. E. Iversen, *Classification of essnusb wc near detector events using graph neural networks*, Talk presented at the HAMLET Conference, Slides available at: `https://indico.nbi.ku.dk/event/2067/contributions/15533/attachments/4742/7594/HAMLET2024_ESSnuSB.pdf`, 2024.

[27] G. Wendel, *Deep learning event reconstruction techniques for the cloud liquido based experiment*, Poster presented at Neutrino2024, Available at: `https://doi.org/10.5281/zenodo.13498314`, 2024.

[28] R. Abbasi, M. Ackermann, J. Adams, *et al.*, "Graph neural networks for low-energy event classification & reconstruction in icecube", *Journal of Instrumentation*, vol. 17, no. 11, P11003, 2022. DOI: `10.1088/1748-0221/17/11/P11003`. [Online]. Available: `https://dx.doi.org/10.1088/1748-0221/17/11/P11003`.

[29] P. Eller *et al.*, "Sensitivity of the IceCube Upgrade to Atmospheric Neutrino Oscillations", *PoS*, vol. ICRC2023, p. 1036, 2023. DOI: `10.22323/1.444.1036`. arXiv: `2307.15295 [astro-ph.HE]`.

[30] A. Chow, L. Heinrich, P. Eller, R. Ørsøe, and S. Dane, *IceCube - Neutrinos in Deep Ice*, 2023. [Online]. Available: `https://kaggle.com/competitions/icecube-neutrinos-in-deep-ice`.

[31] P. Eller *et al.*, "Public Kaggle Competition "IceCube - Neutrinos in Deep ice"", *PoS*, vol. ICRC2023, p. 1609, 2023. DOI: `10.22323/1.444.1609`.

[32] P. Eller, "Public Kaggle Competition "IceCube – Neutrinos in Deep Ice"", in *38th International Cosmic Ray Conference*, Jul. 2023. arXiv: `2307.15289 [astro-ph.HE]`.

[33] C. Bellenghi *et al.*, "Extending the IceCube search for neutrino point sources in the Northern sky with additional years of data", *PoS*, vol. ICRC2023, p. 1060, 2023. DOI: `10.22323/1.444.1060`. arXiv: `2308.12742 [astro-ph.HE]`.

[34] J. Lazar, S. Meighen-Berger, C. Haack, D. Kim, S. Giner, and C. A. Argüelles, "Prometheus: An open-source neutrino telescope simulation", *Comput. Phys. Commun.*, vol. 304, p. 109 298, 2024. DOI: `10.1016/j.cpc.2024.109298`. arXiv: `2304.14526 [hep-ex]`.