Friedrich-Alexander-Universität
Faculty of Sciences

Master‘s thesis

2024

Bastian Setter

Optimisation of energy reconstruction with sample weights for GNNs in KM3NeT/ORCA6

Acknowledgements

I want to to thank all of my colleagues at ECAP and KM3NeT for the wonderful time I had during my master's thesis! Many thanks for the many interesting discussions about neutrino physics and machine learning!

Special thanks go to PD Dr. Thomas Eberl and Dr. Alba Domi for allowing to write this thesis and the excellent support provided throughout its entirety.

I also want to thank Julia Häfner, Anna Eimer, Philipp Laub, Mario Engelmann, Lukas Hennig and Michail Chadolias for proofreading this thesis.

Abstract

With the discovery of neutrino oscillations by Super-kamiokande and SNO it has been shown that neutrinos have mass. This conflicts with the current formulation of the Standard Model of particle physics, where neutrinos are assumed massless. This makes neutrinos an excellent candidate to explore multiple topics in physics beyond the Standard Model, like understanding neutrino oscillations, determining the neutrino mass hierarchy, probing Lorentz invariance or measuring Quantum decoherence.
For the detection of neutrinos, the KM3NeT collaboration is currently building two water Cherenkov telescopes named ORCA and ARCA. With a spacing of 10 m to 20 m between its detection units KM3NeT/ORCA excels at the detection of atmospheric neutrinos in the GeV range. Since KM3NeT/ORCA is still under construction, this thesis uses data for only 6 of the planned 115 detection units.
All of the just mentioned use cases of atmospheric neutrinos require an accurate energy reconstruction to give significant results. The best energy reconstructions in KM3NeT can at the moment be achieved with Graph Neural Networks (GNN).
This thesis tests the effects and possible benefits of applying sample weights in the training of GNNs. Therefore, three different options to calculate the weights are evaluated. The best-found configuration is then compared to the likelihood-based reconstruction algorithms JShower and JMuon, as well as a previous GNN energy reconstruction by Daniel Guderian. The 3 test sample weight options are the ratio of standard run-by-run simulations to additional single-run simulations by Lukas Hennig, the ratio of physical to flat weights for interaction types, and the ratio of physical to flat weights for the spectrum of energy and arrival direction.

Contents

Acronyms

adam adaptive moment estimation

ANTARES Astronomy with a Neutrino Telescope and Abyss environmental Research project

ARCA Astroparticle Research with Cosmics in the Abyss

BAIKAL-GVD Baikal Gigaton Volume Detector

DG Daniel Guderian

DMCR Data/Monte Carlo ratio

DOM digital optical module

DU detection unit

elu exponential linear unit

GNN Graph Neural Network

KM3NeT Cubic Kilometre Neutrino Telescope

LE logarithmic error

LH Lukas Hennig

MEOC main electro-optical cable

MHPO manual hyperparameter optimisation

ORCA Oscillation Research with Cosmics in the Abyss

P-ONE Pacific Ocean Neutrino Experiment

PMT photomultiplier tube

PWS Physical weight share

relu rectified linear unit

RFE relative fractional error

selu scaled exponential linear unit

SGD stochastic gradient descent

TRIDENT The tRopIcal DEep-sea Neutrino Telescope

1 Introduction

Neutrinos are fascinating particles, allowing for advances on multiple fronts in astroparticle physics. Generated in the most violent processes in the universe, cosmic neutrinos improve on cosmic rays and gamma rays as very potent messenger particles in astrophysics. As neutrinos are uncharged particles, they are, in contrast to cosmic rays, not deflected by (inter-)galactic magnetic fields and point directly back to their source. Gamma rays also do not suffer from a deflection but are easily absorbed by matter like gas clouds and galaxies, resulting in a quite opaque view of the universe. Since neutrinos only interact weakly, they traverse most matter without interacting.

The greatest problem in the research of neutrinos is given by their extremely small cross-section. To illustrate this, one can look at the solar neutrino flux, interacting with the human body. Even though about 700 billion solar neutrinos pass through the tip of a finger each second, no effects on one's general health can be observed since almost no neutrinos interact. To still detect a significant amount of neutrinos, enormous detector volumes are required. Furthermore, neutrinos can only be detected indirectly based on the particle cascade produced by their rare interactions. The detector used in this thesis is the water Cherenkov detector KM3NeT/ORCA6 located on the floor of the Mediterranean Sea and operated by the KM3NeT collaboration.

Additionally, neutrinos have great potential to advance our understanding of particle physics. Due the discovery of neutrino oscillations by Super-kamiokande and SNO it has been shown that neutrinos have mass. As this conflicts with the current formulation of the Standard Model of particle physics, where neutrinos are assumed massless, it makes them an ideal candidate to probe our understanding of physics beyond the Standard Model. Be it further constraining the factors contributing to neutrino oscillations, determining the correct neutrino mass hierarchy, investigating CP-violations, or testing concepts of Grand Unified Theory to combine gravity (General relativity) with the other three fundamental forces of electromagnetism (Quantum electrodynamics), weak (Electroweak theory) and strong (Quantum chromodynamics) force, by probing for example Lorentz invariance or quantum decoherence.

For the multitude of these topics, an excellent energy reconstruction of the detected particles is of considerable importance. This thesis will try to improve the performance of the Graph Neural Network (GNN) based energy reconstruction in KM3NeT/ORCA6.

To do this different options of sample weights will be discussed, together with their effects on the performance of the evaluated neural networks.

The thesis starts by explaining the relevant aspects of neutrino physics in chapter 2 followed by a discussion on KM3NeT in chapter 3. Next chapter 4 gives an introduction to Machine learning, starting with the concept of artificial neurons, leading up to the working principle of Graph Neural Networks. chapter 5 will then introduce the concept of weighted training and explain the different options of sample weights. chapter 6 will discuss the way networks are evaluated throughout the thesis and establish a basic agreement with the previous work on GNNs in KM3Net. The main analysis will then be presented in chapter 7. It is comprised of a variation of the sample weight options and an in-depth analysis of networks trained on the best configuration found. At last chapter 8 will conclude this thesis and give an outlook on what can and should be investigated further.

2 Neutrino physics

This chapter will give an overview of the relevant concepts regarding neutrinos in physics by starting with the role of neutrinos in the Standard Model of particle physics (SM), which is followed by an explanation of neutrino oscillations as well as the production process of atmospheric neutrinos. The last section of this chapter will discuss the detection principle utilised by water Cherenkov telescopes and the different types of interactions neutrinos can undergo together with the shape of the respective particle cascade.

2.1 Neutrinos as a part of the Standard Model

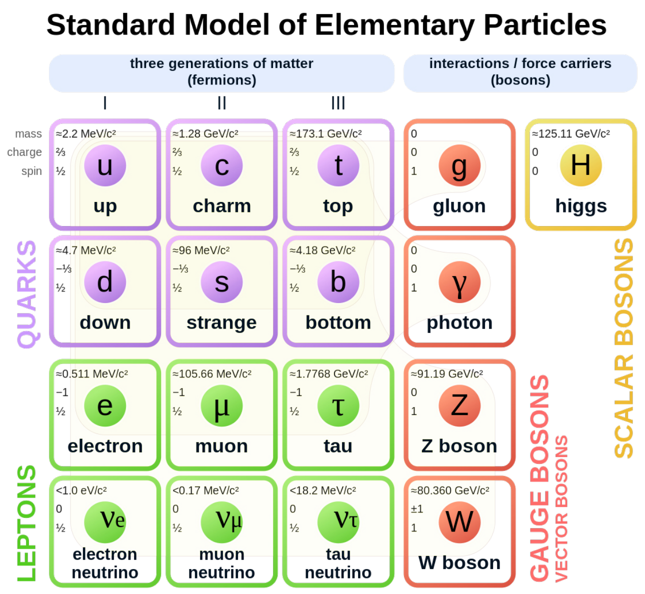

Standard Model of Elementary Particles
three generations of matter (fermions)
interactions / force carriers (bosons)
I
II
III
mass
charge
spin
≈2.2 MeV/c² ⅔ ½ u up
≈1.28 GeV/c² ⅔ ½ c charm
≈173.1 GeV/c² ⅔ ½ t top
0 0 1 g gluon
≈125.11 GeV/c² 0 0 H higgs
≈4.7 MeV/c² −⅓ ½ d down
≈96 MeV/c² −⅓ ½ s strange
≈4.18 GeV/c² −⅓ ½ b bottom
0 0 1 γ photon
≈0.511 MeV/c² −1 ½ e electron
≈105.66 MeV/c² −1 ½ μ muon
≈1.7768 GeV/c² −1 ½ τ tau
≈91.19 GeV/c² 0 1 Z Z boson
<1.0 eV/c² 0 ½ νe electron neutrino
<0.17 MeV/c² 0 ½ νμ muon neutrino
<18.2 MeV/c² 0 ½ ντ tau neutrino
≈80.360 GeV/c² ±1 1 W W boson
QUARKS
LEPTONS
GAUGE BOSONS
VECTOR BOSONS
SCALAR BOSONS

Figure 2.1: *Overview over the Standard Model of particle physics. Shown are the most basic particles of matter (Fermions), together with the force carrying bosons. The faint brown lines indicate, with boson interacts with which fermion. The respective anti particles of the fermions are not shown. From [1].*

Postulated by W. Pauli in 1930 in order to ensure the conservation of energy, momentum, and angular momentum in the double beta decay [2], evidence for its existence has only been found in 1956 by F. Reines and C. Cowan with their discovery of the electron anti-neutrino via the inverse beta decay [3, 4]. Shortly after 1962, L. Lederman, M. Schwartz and J. Steinberger discovered the muon neutrino [5]. Both discoveries were awarded Nobel prizes in 1995 [6] and 1988 [7], respectively. It took, however, another 40 years from the discovery of the muon neutrino, until the existence of the tau neutrino was proven in 2000 by the DONUT collaboration [8]. It was the discovery of the last fermion of the Standard model with three particle generations, predicted by the line width of the Z^0 decay spectrum [9]. An overview of the currently discovered particle zoo is given Figure 2.1.

All neutrinos, ν_e, ν_μ, ν_τ have a spin of 1/2 and carry neither electromagnetic nor colour charge, therefore interacting only via the weak force and gravity. The weak force is mediated through the $W^\pm$ and Z^0 gauge bosons, while gravity is not modelled in the SM, as no model could unit gravity with the current formulation of the SM yet. Assuming neutrinos are Dirac particles, as will be done for this thesis, their antiparticles $\overline{\nu}_e, \overline{\nu}_\mu, \overline{\nu}_\tau$ vary by opposingly signed lepton number, weak isospin and chirality, showing right-handed instead of left-handed behaviour, as is the case for normal neutrinos. If they were Majorana particles instead, they would be their own antiparticles.

2.2 Neutrino oscillations

One of the reasons why neutrinos are so interesting to physics is that they are assumed with zero mass by the SM. However, the observation of neutrino oscillations, by Kamiokando in 1998 [10] and SNO in 2002 [11], requires all three neutrinos to have a different non-zero mass. The discovery of the anti-electron and muon neutrino, this discovery was awarded a Nobel prize in 2015 [12], constituting the total 4th Nobel prize in the field of neutrino physics.
Neutrino oscillations describe the process of a neutrino being generated with one flavour ν_α, but being measured with a potentially different flavour ν_β, with $\alpha, \beta \in \{e, \mu, \tau\}$ and therefore changing its flavour during its journey through space. Describing the neutrino in terms of their flavour eigenstates does not solve the Dirac equation for free particles. Instead, the flavour eigenstates of the system are understood as a linear combination of a different set of states, the mass eigenstates ν_i, $i \in \{1, 2, 3\}$.

$$\nu_\alpha = \sum_i U_{\alpha,i} \nu_i \tag{2.1}$$

Using the mass eigenstates of the neutrino as a base, the Dirac equation can be solved, for a propagating neutrino as a plane wave:

$$|\nu_i(L,E)\rangle = \exp\left(-\frac{im_i^2 L}{2E}\right)|\nu_i\rangle\,, \tag{2.2}$$

with m_i^2 the squared mass of the neutrino, L the distance traveled since its production and E its energy.
While the flavour eigenstates of a neutrino are responsible for its interactions, the mass eigenstates govern its propagation. The general relation between both sets of eigenstates is given by the PMNS-matrix, a complex, unitary 3x3 matrix named after Pontecorvo [13], Maki, Nakagawa, and Sakata [14].

$$\begin{pmatrix}\nu_e\\ \nu_\mu\\ \nu_\tau\end{pmatrix} = \begin{pmatrix}U_{e1} & U_{e2} & U_{e3}\\ U_{\mu1} & U_{\mu2} & U_{\mu3}\\ U_{\tau1} & U_{\tau2} & U_{\tau3}\end{pmatrix}\begin{pmatrix}\nu_1\\ \nu_2\\ \nu_3\end{pmatrix} \tag{2.3}$$

It is typically parameterised in terms of three rotations determined by the three mixing angles $\theta_{12}, \theta_{13}, \theta_{23}$ and the CP-violating phase δ_{CP}, reducing the nine correlated entries of the PMNS-matrix down to only four independent values. In the case of Majorana particles, two additional phases would occur. c_{ij} and s_{ij} denote $cos\left(\theta_{ij}\right)$ and $sin\left(\theta_{ij}\right)$.

$$U_{\text{PMNS}} = \begin{pmatrix}1 & 0 & 0\\ 0 & c_{23} & s_{23}\\ 0 & -s_{23} & c_{23}\end{pmatrix}\begin{pmatrix}c_{13} & 0 & s_{13}e^{-i\delta_{CP}}\\ 0 & 1 & 0\\ -s_{13}e^{i\delta_{CP}} & 0 & c_{23}\end{pmatrix}\begin{pmatrix}c_{12} & s_{12} & 0\\ -s_{12} & c_{12} & 0\\ 0 & 0 & 1\end{pmatrix} \tag{2.4}$$

Without explicit derivation, from the plane wave description in Equation 2.2 and the relation of flavour and mass eigenstate given by the PMNS matrix in Equation 2.3, the probability for a neutrino to oscillate from flavour α into β, is given by

$$P_{\alpha\to\beta}(L,E) = |\langle\nu_\beta|\nu_\alpha(L,E)\rangle|^2 = \sum_{i,j} U_{\alpha i}U^*_{\beta i}U^*_{\alpha j}U_{\beta j}\exp\left(-i\frac{\Delta m^2_{ij}L}{2E}\right), \tag{2.5}$$

with $U_{\alpha,\beta|i,j}$ the elements of the PMNS-matrix, $\Delta m^2_{ij} = m_i^2 - m_j^2$ the squared mass difference of the relevant neutrino mass states, L the distance travelled since its production and E its energy.
The expression in the sum of the oscillation probability can be divided into two parts. First come the components of the PMNS-matrix $U_{\alpha i}U^*_{\beta i}U^*_{\alpha j}U_{\beta j}$ and therefore a combination of the mixing angles θ_{ij} and the CP-violation phase, determining the amplitude of the oscillation. Secondly, the exponential term including the mass differences Δm^2_{ij}, together with the factor $\frac{L}{E}$, determines the oscillation phase. In order

Parameter	Bfv $\pm$ 1σ	3σ Range
θ_{12} [deg]	$33.45^{+0.77}_{-0.75}$	$31.27 \rightarrow 35.87$
θ_{13} [deg]	$8.62^{+0.12}_{-0.12}$	$8.25 \rightarrow 8.98$
θ_{23} [deg]	$42.1^{+1.1}_{-0.9}$	$39.7 \rightarrow 50.9$
δ_{CP} [deg]	230^{+36}_{-25}	$144 \rightarrow 350$
Δm^2_{21} $[1 \times 10^{-5}\,\mathrm{eV}^2]$	$7.42^{+0.21}_{-0.20}$	$6.82 \rightarrow 8.04$
$\lvert\Delta m^2_{31}\rvert$ $[1 \times 10^{-3}\,\mathrm{eV}^2]$	$2.510^{+0.027}_{-0.027}$	$2.430 \rightarrow 2.593$

Table 2.1: *Best fit values (Bfv) for oscillation parameters in normal ordering and their 3σ confidence intervals.*

to maximise the oscillation probability, one aims for an oscillation phase satisfying

$$(\mathrm{k} \cdot 2 + 1) \cdot \pi = 1.27 \cdot \frac{\Delta m^2_{ij}}{eV^2} \frac{L}{km} \frac{GeV}{E} \tag{2.6}$$

with $\mathrm{k} \in \mathcal{N}_0$ and not given in natural units. For experimental design only L and E can be influenced. The other values need to be determined in experiments.
Only two of the three mass differences are independent, since the third can be calculated from the other two by $\Delta m^2_{31} = \Delta m^2_{21} + \Delta m^2_{32}$. Since $\Delta m^2_{21} << \Delta m^2_{32}$, it holds that $\Delta m^2_{31} \approx \Delta m^2_{32}$. In addition to this, the sign of Δm^2_{32} is currently unknown, allowing for two possible orderings of the neutrino masses, namely normal ordering and inverted ordering, as is illustrated in Figure 2.2. In the case of normal ordering, the squared masses are increasing in natural order $m_1^2 < m_2^2 < m_3^2$, while for the inverted ordering $m_3^2 < m_1^2 < m_2^2$ holds. The current best-fit values of these parameters are given in Table 2.1 [15].

The results presented until now only cover neutrino oscillations in vacuum. By including matter effects like the effective potential A in the Ansatz, the oscillation probability from ν_μ to ν_e becomes approximately:

$$P_{\mu\rightarrow e}(L,E) \approx \sin^2\theta_{23} \cdot \sin^2 2\theta^m_{13} \cdot \sin^2\left(\frac{\Delta^m m^2 L}{4E}\right), \tag{2.7}$$

with

$$\sin^2 2\theta^m_{13} \equiv \sin^2 2\theta_{13} \cdot \left(\frac{\Delta m^2_{31}}{\Delta^m m^2}\right)^2 \tag{2.8}$$

$$\Delta^m m^2 \equiv \sqrt{\left(\Delta m^2_{31} \cdot \cos 2\theta_{13} - 2EA\right)^2 + \left(\Delta m^2_{31} \cdot \sin^2 2\theta_{13}\right)^2}. \tag{2.9}$$

While adding a considerable amount of complexity to the expression, the general structure is still the same. For a detailed discussion on the effects of matter on oscillation probabilities see [17] and for discussion on this particular transition see [18].

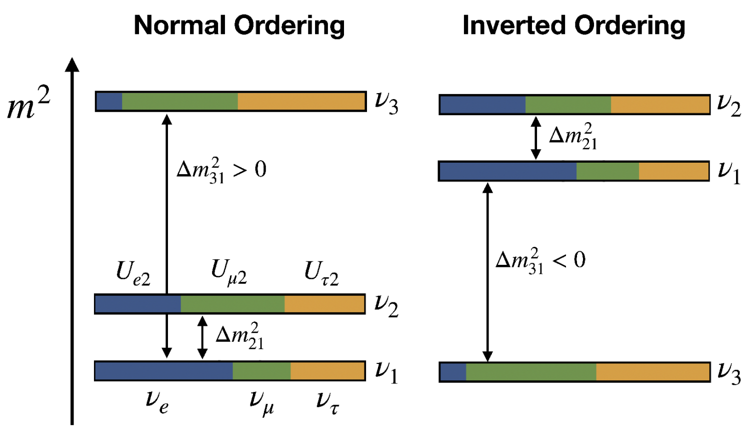


Figure 2.2: *Shown are the 3 mass eigenstates of the neutrinos in the configuration of normal and inverted ordering, caused by the unknown sign of Δm^2_{31}. Indicated are also the ratios of neutrino flavour of each mass eigenstate. From [16].*

2.3 Atmospheric neutrinos

Neutrinos are produced in a multitude of different processes by even more different objects, over a tremendous energy range, ranging from cosmological neutrinos with energies of meV up to cosmogenic neutrinos with energies of EeV, shown in Figure 2.3. The type of neutrino that is most interesting for this thesis, is the atmospheric neutrino, most prominent at GeV energies.
As the name suggests, atmospheric neutrinos are generated in the Earth's atmosphere during collisions of cosmic rays with the atoms in the atmosphere, typically at a height of 15km [20]. In the particle cascade following these collisions, large amounts of charged pions $\pi^\pm$ and kaons $K^\pm$ are produced, first decaying into muons

$$\pi^\pm, K^\pm \rightarrow \mu^\pm + \overset{(-)}{\nu}_\mu, \tag{2.10}$$

which then decay into electrons

$$\mu^\pm \rightarrow e^\pm + \overset{(-)}{\nu}_e + \overset{(-)}{\nu}_\mu. \tag{2.11}$$

During these two decays, a total of two muon neutrinos and 1 electron neutrino is produced. As atmospheric neutrinos are almost exclusively produced in these two decay processes, the expected ratio of neutrino flavours in the resulting flux is approximately 1:2:0 (ν_e: ν_μ: ν_τ). Due to the positive charge of the cosmic rays, in these decays more π^+ than π^- are created, resulting in a ratio of neutrinos to antineutrinos greater than 1 $\frac{\nu_\alpha}{\bar{\nu}_\alpha} > 1$. The ratio of generated muon to electron neutrinos increases with increasing energies since more muons can reach the Earth before decaying, thereby only producing

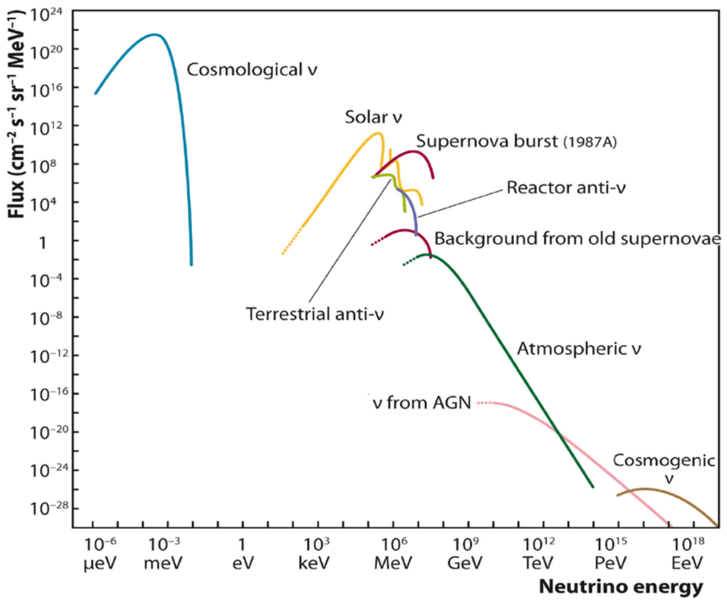


Figure 2.3: *Overview of various neutrino sources, showing the expected or measured flux over energy. From [19].*

a single muon neutrino [20]. Tau neutrinos are only generated at much higher energies, in the decay of other baryons than pions or kaons. Therefore, any ν_τ measured in the detector must have oscillated into this flavour from either a ν_e or ν_μ.
The model of the atmospheric neutrino flux resulting from the cosmic ray flux used for this thesis has been published by the 'HKKM' group [21]. It accounts for seasonal changes in atmospheric density, as well as varying solar activities in multiple sites on the earth's surface. The site used for the flux model of this thesis is the Frejus site located near the KM3NeT/ORCA detector in the south of France. Next to the Honda flux model used in KM3NeT and therefore also this thesis, other neutrino flux models have been calculated, namely the Bartol model by Barr et al. [22] and the FLUKA model by Battistoni et al. [23]. Even though the models agree on the general shape of the atmospheric neutrino flux, they vary up to 15% on the absolute normalization [22].

2.4 Neutrino detection

Carrying no electromagnetic and colour charge, neutrinos interact only via the weak force and can therefore not be detected directly. However, when colliding with nucleons, particle cascades are induced. These particles can then be detected in various ways:

- Scintillators as used by Cowan and Reines [3],

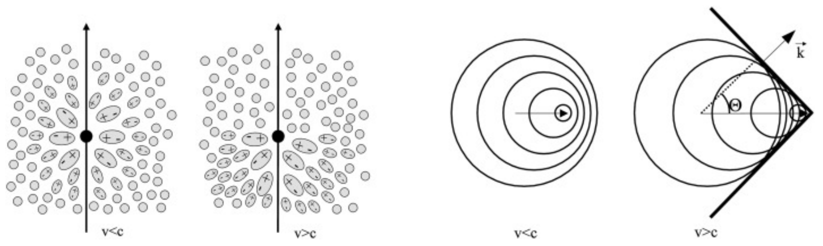


Figure 2.4: *Illustration of Cherenkov radiation. Left: State of polarisation of dielectric particles. Symmetric for $v < c$ and asymmetric for $v > c$.*
Right: Interference of emitted radiation waves. Destructive for $v < c$ and constructive for $v > c$, leading to the formation of a Cherenkov cone with opening angle θ. From [27].

- Cherenkov radiation used by neutrino telescopes such as KM3NeT [18] or IceCube [24],
- Askaryan effect utilised by RNO-G [25].

The extremely small cross-section of about $1 \times 10^{-38}\,\mathrm{cm}^2/\mathrm{GeV}$ allows neutrinos to travel through matter almost untouched, but also makes the task of detecting neutrinos at a reasonable rate very difficult. It therefore requires the use of detectors with a cheap, abundant detection medium of high density, which for detectors based on Cherenkov radiation simultaneously needs to allow for the production and propagation of Cherenkov light [26].

2.4.1 Cherenkov radiation

As illustrated in Figure 2.4 charged particles, moving through a dielectric medium, such as water, cause a polarization in the particles in close proximity to them. Once these particles fall back into their ground state they emit electromagnetic waves. If the particle is moving with a speed $v < c$, these waves, interfere mostly destructively, resulting in no total emission of radiation. However, if the particle is travelling at a speed of $v \geq c$, the individual waves interfere constructively, combining into a cone-shaped wavefront, the so-called Cherenkov cone. The opening angle θ of this cone is dependent on the speed of light in the medium, determined by its refractive index, as shown in Equation 2.12. For water, this angle is 42°.

$$\theta = \arccos\left(\frac{1}{v} \cdot \frac{c}{n}\right) \tag{2.12}$$

The blue Cherenkov light emitted by the charged particles can then be measured by photomultiplier tubes (PMT), capable of detecting even single photons.

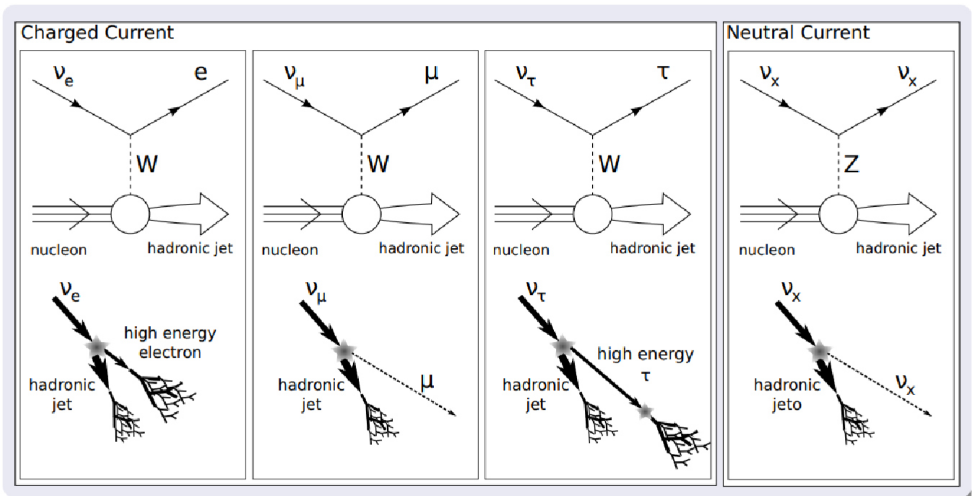


Figure 2.5: *Overview over the different interactions neutrinos can undergo. Mediated by the $W^\pm$ boson in the charged current channel, emitting the respective lepton, as well as a hadronic jet and by the Z^0 boson, resulting in only a hadronic jet. From [28].*

2.4.2 Event topologies

Depending on the neutrino flavour and the gauge boson mediating its interaction (with a nucleus), four different types of particle cascades are produced, as illustrated in Figure 2.5. Interactions mediated by the Z^0 boson are called neutral current interactions, creating only a hadronic jet, independent of the flavour of the neutrino. If the interaction is instead mediated by a $W^\pm$ boson, it is called a charged current interaction, creating a hadronic jet, as well as a lepton corresponding to the neutrinos flavour, resulting in different types of particle cascades. For ν_e it results in an electromagnetic shower, for ν_μ it results in a track-like event and for ν_τ we observe so-called double bang events. In all of these interactions, the incoming neutrino is called the primary particle and all subsequently produced particles are called secondary particles. All four topologies will now be described.

Hadronic jets Hadronic jets are produced in all neutrino interaction types by the initial energy transfer from the neutrino to a quark inside the nucleus, generating a large amount of further hadrons, which generally travel in the direction of the primary particle. Many of the secondary particles in the hadronic jet are uncharged pions, which decay after only a short time span of 8.5×10^{-17} s into two photons, adding an electromagnetic component to the particle cascade. This becomes increasingly significant for higher energies of the primary particle, approaching an identical cascade structure as generated by a ν_e. The attenuation length of the hadronic jet in water, meaning the length over which 1/e% of the particles lose enough energy so they can no longer emit Cherenkov radiation, is about 83 cm [29].

Electromagnetic showers The emission of an electron in the interaction starts a so-called electromagnetic shower. It interacts with its environment mostly via Bremsstrahlung creating photons and thereby losing energy [30]. The emitted photons produce, via pair production and Compton scattering, electrons and positrons, which again interact via Bremsstrahlung and so on. The Cherenkov light emitted by an electromagnetic shower is with an error of 1% proportional to the energy of the neutrino [29], allowing for a very good energy reconstruction given the whole shower is contained in the detector. Compared to the hadronic jet, the attenuation length of the electromagnetic shower is shorter at 36 cm [30], even though the Cherenkov threshold of electrons of 0.8 MeV is considerably lower, than that for protons with 1.4 GeV [31].

Tracks In contrast to the short attenuation lengths of hadronic and electromagnetic interactions, muons travel large distances of about 4 m/GeV in water until they decay, emitting radiation along their path [30]. This elongated shape of the released energy allows for a good direction reconstruction but is also often the reason for a worse energy reconstruction since the muon can leave the detector volume before depositing all of its energy inside. At higher energies (TeV), muons lose their energy in bursts by creating small electromagnetic showers along their way, while at low energies (few GeV), the constant emission of Cherenkov radiation becomes the dominant process of energy loss.

Double Bang If a ν_τ interacts via the charged interaction channel, it creates a τ lepton that travels about 5 cm/TeV in water until it decays into any of the three already discussed particle cascades.

$$\begin{aligned}
\tau^- &\rightarrow \text{hadronic jet} : 64.8\% \\
\tau^- &\rightarrow \nu_\tau + e^- + \overline{\nu}_e : 17.8\% \\
\tau^- &\rightarrow \nu_\tau + \mu^- + \overline{\nu}_\mu : 17.4\%
\end{aligned}$$

Since the decay of the τ creates a second interaction vertex, from which radiation is emitted, these events are called double bang events. For the GeV energies of atmospheric neutrinos, the decay of the created τ is effectively instantaneous, leaving no possibility to spatially resolve this phenomenon.

2.4.3 Neutrino notation

In the following thesis, the just discussed interaction types will be denoted as ν_e^{CC}, ν_μ^{CC}, ν_τ^{CC} and ν_x^{NC}, by the neutrino's flavor and interaction channel. Since the neutral current interaction is independent of the neutrino flavor, it only shows the interaction channel. In order to ensure visibility in plots and equations, the short notation of ν_e, ν_μ, ν_τ and ν_{NC} is used. If not stated otherwise, form here on antineutrinos and neutrinos are referenced together by this notation.

3 KM3NeT

KM3NeT [18] is an international research collaboration for neutrino physics, acting as the successor experiment for ANTARES [32], which was moved into legacy mode in 2022 [33]. At the moment of writing this thesis, KM3NeT has 48 member institutions comprised of 45 postdocs, 81 PhD students and 36 master's students [34]. It is currently building two next-generation water Cherenkov detectors in the Mediterranean Sea. Namely ORCA, short for "Oscillation Research with Cosmics in the Abyss", and ARCA, which stands for "Astroparticle Research with Cosmics in the Abyss". An overview of the member institutions and detector site locations is presented in Figure 3.1.

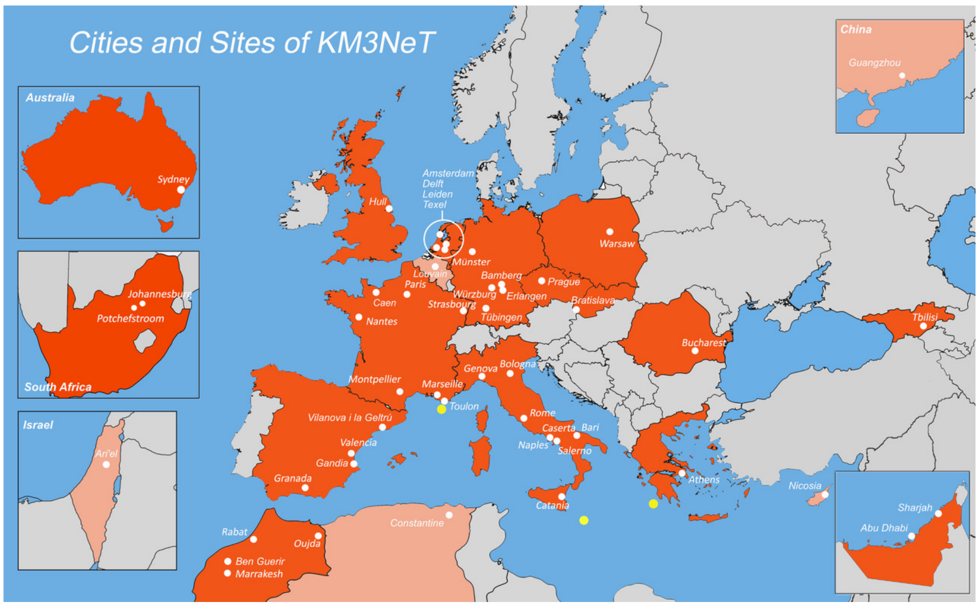


Figure 3.1: *Overview of the member institutes of the KM3NeT collaboration (white dots) and the operated sites near France and Italy (yellow dots). From [34].*

3.1 Detector design

In order to detect neutrinos with energies up to MeV, as is necessary for reactor experiments like Daya Bay [35] and MiniBooNE [36] or solar neutrino experiments like Super-Kamiokande [37], the most common type of detector is a tank filled with a suitable detection medium and surrounded by a large number of PMTs to detect the emitted Cherenkov radiation. The typical interaction medium used is water, which can be doped with gadolinium to further enhance its detection qualities. At the moment the largest of these artificial tanks is employed by Super-Kamiokande, containing about 50 kTons of water.
However, if the energy of neutrinos one wants to measure increases to GeV and above, much larger detector volumes are required, to still generate a high amount of identifiable neutrino events, even though at higher energies the neutrino flux is much lower. Building larger and larger tanks, however, is only feasible up to a certain point. Therefore, the science community has turned to natural media, such as freshwater lakes, oceans and the ice caps at the poles to measure high-energy neutrinos. The two detectors build by KM3NeT for instance use the water of the Mediterranean Sea as their detection medium. It has to be noted, that Super-Kamiokande is also able to measure the atmospheric neutrino flux at GeV energies, as used in the discovery of neutrino oscillations, but the amount of measured events is much lower compared to natural medium detectors.
Even though ORCA and ARCA are optimised for different science cases (see section 3.2), they are built with the same design, varying only by the density of their instrumentation. In the case values vary, they will be denoted as follows: (ORCA value; ARCA value). To explain the individual components of the detectors, a sketch of their design is displayed in Figure 3.2. The main component of the detectors are the so-called digital optical modules (DOMs), containing 31 PMTs arranged approximately equidistant to each other, detecting the Cherenkov light emitted by neutrino particle cascades from all directions. 18 of those DOMs are grouped into a detection unit (DU), fixed along a line with a vertical spacing of (9 m, 36 m) and anchored to the floor of the sea with a horizontal distance of on average (20 m, 90 m) to each other [39]. The DUs are held upright by a buoy made of syntactic foam. The anchors of the individual DUs are then linked together by junction boxes, relaying the data measured by the PMTs and DOMs to the control station on shore through the main electro-optical cable (MEOC) [40]. The exact positions of the DU anchors and cabling for both detectors can be taken from their footprints provided in Figure 3.3.

As the DUs are not fixated at the top, but only straightened vertically by a buoy at the top end of each DU, the individual DOM positions are susceptible to sea currents. In order to measure the 3D drift of the DOMs, a piezo-acoustic sensor is

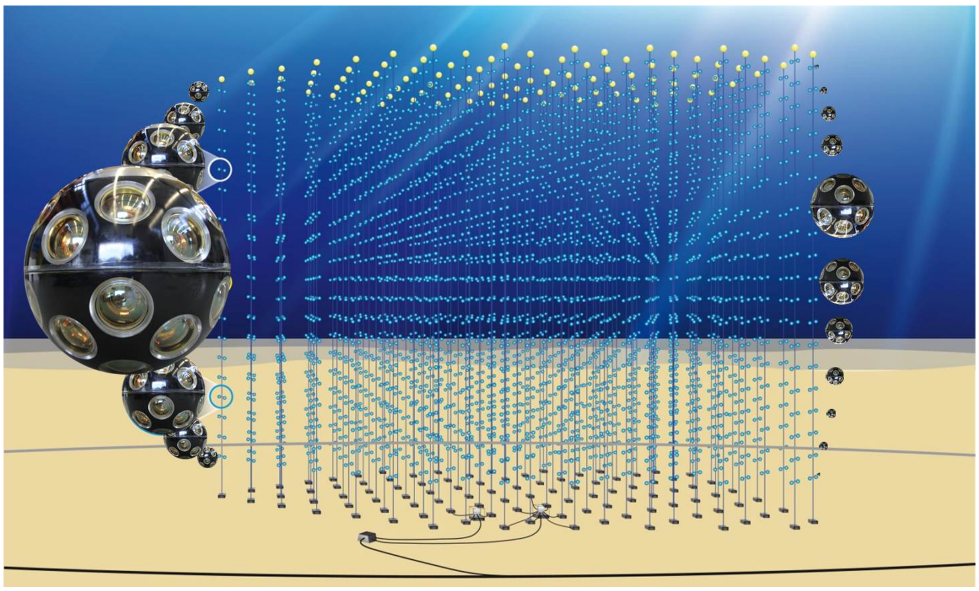


Figure 3.2: *Sketch of the KM3NeT underwater detector design, including DOMs, DUs, anchors, junction box and MEOC. From [38].*

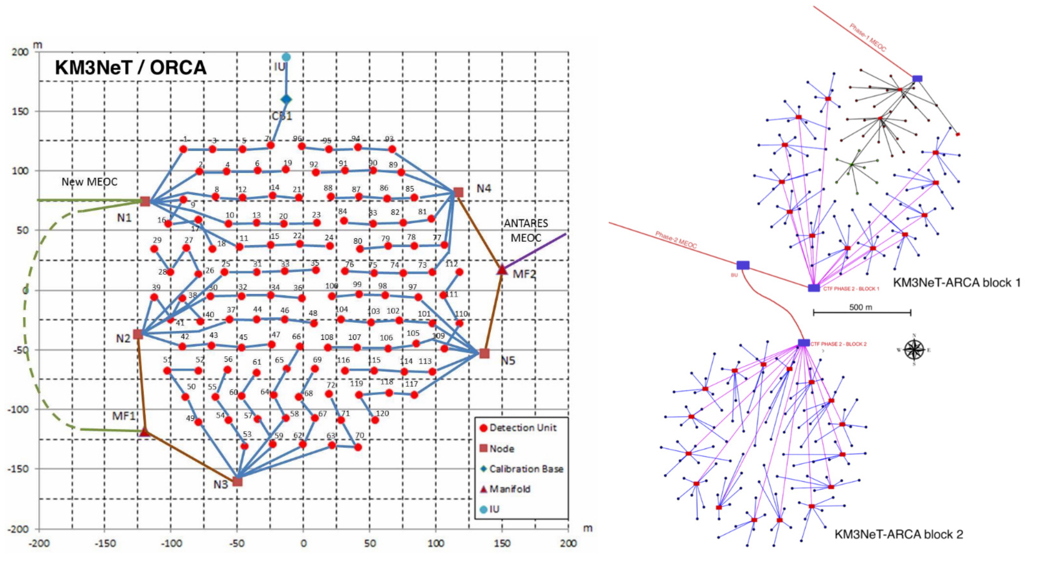


Figure 3.3: *Footprint of the KM3NeT/ORCA (left) and KM3NeT/ARCA (right) detector, showing relative DU anchor positions and cable routing. DUs 1-6 in the top left corner of the ORCA detector will be used for this thesis. From [18].*

used in combination with ground-based sonar. Additionally, the tilt, pitch and yaw are determined by accelerometers. Combining these features allows every PMT to record the following features for an incoming photon:

- position (pos_x, pos_y, pos_z)
- pointing direction (dir_x, dir_y, dir_z)
- time of hit (t)
- Time-over-Threshold (ToT)

After applying triggers to the received PMT signals, they are grouped into single events and stored, together with other high level data, like the number of triggered PMTs in the whole events in the root file format.

3.2 Detector sites

ARCA [18] The site of KM3NeT/ARCA is located 100 km south of the Sicilian coast. As the experiments of IceCube [24], BAIKAL-GVD [41], P-ONE [42] and TRIDENT [43], it employs a detector volume of GTons. This massive detector volume allows for the efficient detection of neutrinos in the energy range of PeV-TeV and, thereby, for a survey of the neutrino sky. By this new insights for the production sites of neutrinos in and outside our galaxy will become available. Additionally, the detector is expected to further extend the catalogue of neutrino sources from the four currently known sources of our sun [44], the supernova 1987A [45], blazar TXS 0506+056 (ICECUBE, 2017) [46] and NGC 1068 (ICECUBE, 2022) [47].

ORCA [18] KM3NeT/ORCA is located 40 km offshore from Toulon on the French coast, 10 km west of the ANTARES site. With a detector volume of only 7 MTon ORCA is much smaller than ARCA, but instead boasts a denser instrumentation, excelling in the detection of atmospheric neutrinos with energies of 3-30 GeV. Together with a travelling distance of thousands of km, the L/E for atmospheric neutrinos is on the order of 1×10^3 GeV/km. Following Equation 2.6, this makes ORCA ideal to measure Δm^2_{31} and therefore determine the correct neutrino mass hierarchy as explained in section 2.2. Additionally, further constraints on the mixing angle θ_{23} (see section 2.2) will be set by this experiment.
Despite the relatively dense instrumentation of ORCA, it is still too sparsely instrumented to spatially resolve the neutrino topologies of the hadronic jet, electromagnetic shower and GeV double bang events (subsection 2.4.2), seeing them as essentially point-like light sources. Furthermore, due to ORCA still being under construction, with currently only 18 DUs deployed and operational, this thesis uses data from the unfinished ORCA6 detector, which has been taking data for a period of 510 days.

3.3 Background sources

All current neutrino observatories are built under a tremendous amount of mass, be it a kilometre thick sheet of ice, as is the case for IceCube, Mountains for the Kamiokande experiments, or the ocean in the case of KM3NeT. This massive amount of matter provides an excellent shield from most unwanted background signals, like sunlight and most low energetic muons. However, even through this shield, the greater part of all events measured in the detector are caused by atmospheric muons with original energies above 55 GeV. Since muons travel about 4 m/GeV in water, the 2300 m of water above the ORCA detector reliably only absorb muons with an original energy of less than 55 GeV [48].
The remaining muons result in an expected ratio of atmospheric muon to neutrino events measured in the detector of about 10000:1, making it very important to reject this atmospheric muon. To accomplish this, while retaining the muons generated during neutrino interactions, most of the time a cut is placed on the reconstructed arrival direction of the event, requiring a zenith angle of greater than 0. Since no particle except for neutrinos can travel through the earth, any upgoing event has to be produced by a neutrino.
In additionally to the contamination with atmospheric muons, the detector also records pure noise events caused mainly by various processes of bioluminescence [49] like dinoflagellates and zooplankton [50] and so-called K40-events [51], the decay of potassium into calcium or argon:

- Beta decay (Branching ratio 89.28%): $^{40}\mathrm{K} \rightarrow^{40} \mathrm{Ca} + e^- + \overline{\nu}_e$
- Electron capture (Branching ratio 10.72%): $^{40}\mathrm{K} + e^- \rightarrow^{40} \mathrm{Ar} + \nu_e$
- Positron emission (Branching ratio $< 0.001\%$): $^{40}\mathrm{K} \rightarrow^{40} \mathrm{Ar} + e^+ + \nu_e$

They are mostly filtered by a minimum requirement of the number of hits in any given event and a score of a noise-signal classifier. The level of noise and contamination events varies throughout the year, depending on for example water temperature and atmospheric densities.

4 Deep Learning

Deep learning is a subfield of computer science and more specifically machine learning. It uses artificial neural networks (ANNs) or just neural nets (NNs) in order to recognise patterns in raw high level data and extract low level quantities from it.
The first comprehensive summary of deep learning, introducing the concept of the multilayer perceptron(MLP), was published by Frank Rosenblatt as early as 1962 [52]. The impressive potential NNs have, could however only be shown in recent times by outperforming most traditional methods of data analysis. The risen popularity has been enabled by the increase in access to fast, abundant and relatively cheap computing power, as well as large amounts of raw data. Applications for NNs comprise for example: making generalised predictions for given data samples as is the case for AlphaFold, where the researchers managed to predict the folding function of thousands of previously unknown protein structures [53, 54], discovering optimal strategies based on the state of a environment or game, with AlphaGo beating the world champion in 2016 [55] or being able to talk and have seemingly profound discussions as is done with large language models [56] like GPT [57].
This chapter will start with an explanation of the idea and basic building blocks of neural networks, namely neurons and the multilayer perceptron. After that, the process of training a neural network, as well as the special types of neural networks used in this thesis. It will close with a description of the way deep learning is implemented in KM3NeT.

4.1 Basics of artificial neural networks

The digital structure of artificial neural networks is inspired by biological brains. Combining the in-and outputs of a large number of artificial neurons into so-called layers and imitating the nonlinear behaviour of neuron cells, complex behaviour emerges, as is the case for a biological brain. While the imitated biological neurons give rise to a similar complex structure and behaviour when combined, artificial neurons by themselves are much less complicated than their biological counterpart.

4.1.1 Artificial neurons

The idea of artificial neurons is to take input data and compute an output value based on its internal parameters. This is achieved in three steps. First, the neurons take in data x^M, typically from other neurons. M denotes here the length of the input data vector. Next, the inputs get multiplied with individual weights w^M and summed up. Then a bias b is added. At last, an activation function ϕ gets applied to the resulting number. The mathematical expression describing the calculation of the neuron output looks as follows:

$$f(x, \theta) = \phi\left(w^{1\times M} \cdot x^M + b\right). \tag{4.1}$$

The set of weights w and bias b together will be addressed as weights θ in the following. By adjusting the weights for every neuron during the training (see section 4.2), the neurons are able to adjust and learn from the inputs. An additional prerequisite for the neurons, in order to be able to learn complex patterns in the data is the non-linearity of the activation function. If it were linear instead, an arbitrary combination of neurons could always be reduced down to a single neuron, removing the essential complex behaviour of NNs. The non-linearity can be seen in two popular activation functions called sigmoid and rectified linear unit (relu):

$$\text{relu: } \phi(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

$$\text{sigmoid: } \phi(x) = \frac{1}{1 + e^{-x}}. \tag{4.3}$$

4.1.2 Dense layer

A set of N neurons, placed in parallel to each other, is called a layer, which can be stacked on top of each other to form the NN. Each neuron in the i-th layer can be connected to neurons in the previous (i-1) and following (i+1) layer. The simplest layer type is called a dense or alternatively fully connected layer. In this layer, each of the N neurons of the i-th layer is connected to every of the M neurons in the previous (i-1)-th layer. Extending the formulation for the output of a single neuron into one for a full layer given by Equation 4.1 and thereby extending 1D vectors to 2D matrices gives the following:

$$f^{(i)}(x^{(i)}, \theta^{(i)}) = \phi\left(w^{N\times M} \cdot x^M + b^N\right), \tag{4.4}$$

with $x^{(i)} = f^{(i-1)}$ the output of the previous layer.
Next to dense layers, other types have been developed, utilising symmetries in the data or using different ways of data representation, than 1D vectors. The for this thesis relevant ones will be explained in section 4.4.

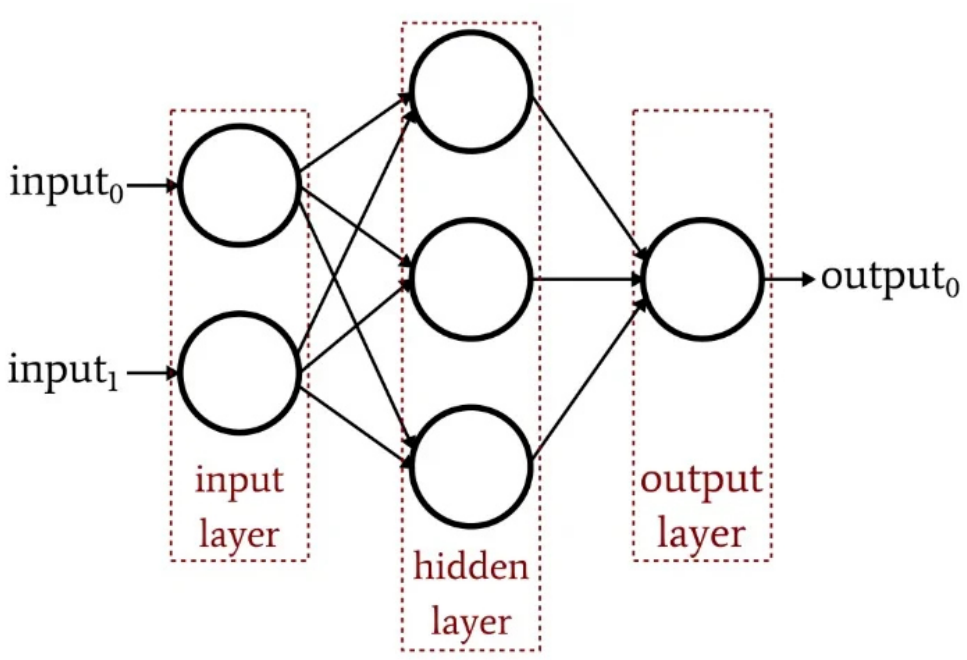


Figure 4.1: *Layout of a multilayer perceptron, with an input layer of size 2, one hidden layer and an output layer of size 1. From [58].*

4.1.3 Multilayer perceptron

As just mentioned, by stacking at least three layers on top of each other, as is shown in Figure 4.1, one can build a mulitlayer perceptron (MLP). The layers in a multilayer perceptron can be split into three groups of distinct purposes:

Input layer The first layer of the network is called the input layer. It serves as the entry point for the user, to provide data to the network. The provided data equates to this layer's output and no weighting or activation function gets applied.

Output layer The last layer of a neural network is called the output layer. It returns the network response to a given input and is used to enforce certain boundaries to the desired output. For example, if one would want to reconstruct a quantity, which is by definition positive like energy, this could be accomplished by applying the relu activation function (Equation 4.2). If no adjustment is needed, also a liner activation function $\phi(x) = x$ can be used in the output layer, as non-linearity is not required in this final step.

Hidden layer All layers in between the input and output layers are called hidden layers, as they cannot be accessed or 'seen' by the user in an easy way. They are the central part of the network, where it generalises, learns and memorises. Even though it can be shown, that a single hidden layer with an infinite or at least sufficiently high number of neurons, can approximate an arbitrary complex function perfectly [59], in most cases, it is beneficial to use more than one hidden layer. This allows the network to more easily grasp possible multi-level patterns present in the data, since it can now extract various relevant information in a first step and only in a second step combine these intermediate learnings into a final output.

4.2 Training

During the training of a neural net, the weights θ of its neurons are adjusted to optimally represent the input data given to the network. The term optimal is here highly dependent on the target of the user as well as the regime of deep learning applied. In contrast to other regimes like semi-supervised learning, reinforcement learning or active learning, this section will only cover the training process of supervised learning, as this is the type used in this thesis.
The goal of supervised learning is to approximate a true quantity $y_{i,\text{true}}$ of an input x_i, for a variety of events, as well as possible. In order to achieve this the network is presented with a large amount of data tuples $(x_i, y_{i,\text{true}})$, made up of the input data and the true desired output. By comparing the output of the network to the true value via a loss function, the weights of the network are adjusted gradually, until the average reconstruction of the target quantity is no longer improving. As this thesis is doing energy reconstruction of data for the KM3NeT/ORCA6 detector, the input data is the detector response produced by neutrino interactions for x_i together with the neutrino's corresponding energy for $y_{i,\text{true}}$. The reconstruction of a contiguous quantity, such as the energy, is called regression, while the reconstruction of a discrete quantity, like interaction type, is called classification.
During training the user has the option to specify various parameters, that influence how the training is executed. These parameters are called hyperparameters. In the following text, they will be highlighted by a dot over the respective symbol e.g. $\dot{a}$. Hyperparameters that were already mentioned in this chapter are the weights $\dot{\theta}$, activation function $\dot{\phi}$ and the number of layers in the network, together with any specifications of the layer type like the number of neurons $\dot{N}$ for dense layers.

4.2.1 Loss

In order to be able to adjust the network weights adequately, the loss function $\dot{\mathcal{L}}$ is used. It calculates the error between the true value $y_{i,\text{true}}$ of an event x_i and the network's prediction $y_{i,\text{pred}} = f(x_i, \theta)$. During the training, this function is sought to be minimised for the average of the training tuples.

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_{i,\text{true}}, f(x_i, \theta)) \tag{4.5}$$

Some options for the loss function of a regression problem are the mean squared error (MSE), mean absolute error (MAE), mean squared logarithmic error (MSLE), and log-normal loss. The choice of loss function should be done based on domain knowledge and be verified by performance tests. Domain knowledge describes the intuitive understanding of problems based on previous experience in any field, resulting in the ability to infer possibly beneficial adjustments to the network.

$$\mathcal{L}(y_{i,\text{true}}, y_{i,\text{pred}}) = \begin{cases} (y_{i,\text{true}} - y_{i,\text{pred}})^2 & \text{MSE} \\ |y_{i,\text{true}} - y_{i,\text{pred}}| & \text{MAE} \\ (\log(y_{i,\text{true}}) - \log(y_{i,\text{pred}}))^2 & \text{MSLE} \\ (\log(y_{i,\text{pred}}) - \log(y_{i,\text{true}}))^2 & \text{LogNormal} \end{cases} \tag{4.6}$$

4.2.2 Optimiser

The task of adjusting the network weights based on the loss function during the network training is performed by an optimiser. Since a neural network typically consists of an immense number of neurons, a complete search of the phase space for the best set of weights is most often not feasible. Alternatively, one can take an iterative approach, where a random setting start set of parameters is updated in several steps t, until solution is found to the problem. While most likely not finding the perfect solution, it still results in a relatively optimised solution, while requiring much less computation, than the full search.

Gradient descent A very simple type of iterative minimisation technique is called gradient descent. It uses the gradient of each weight with respect to the loss function $\nabla_\theta \mathcal{L}$, to calculate the optimal change of each weight i.e. in the direction of strongest decrease. Changing the weights once is called a weight update.
The calculation of the gradients is done today in a process called backpropagation, which utilises the repeated application of the chain rule to calculate the gradients with respect to the network output. By calculating the gradients for the individual neurons

backwards, meaning starting from the output neuron, the calculated gradients can be reused for the gradients further up the model. Before that, gradients were calculated starting from the inputs, where for every single neuron all consequent gradients had to be calculated. Already proposed by Rosenblatt in 1962 [52], backpropagation became widespread only after the analysis by David E. Rumelhart in 1986 [60].
Based on the weights and gradients of the current time step t, the next set of weights in timestep $t+1$ is calculated, with the update rule:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_{\theta_t}\mathcal{L}, \tag{4.7}$$

with $\dot{\eta}$ the learning rate, a hyperparameter used to scale the step size, meaning the absolute change in weight.

stochastic gradient descent (SGD) In order to get the most accurate weight update out of gradient descent, the calculation of the average gradient for the whole dataset is required. This however is not feasible, since the gradients for millions of parameters for millions of events are very memory intensive. Instead, a slight variation, named stochastic gradient descent is used. It replaces a single weight update over the whole dataset, with several weight updates over smaller chunks of the whole dataset, which approximate the full gradient. These chunks are called batches. The number of events included in one batch is called the batch size. The training on the whole dataset and therefore all available batches is called an epoch.

Adam The optimiser used throughout this thesis is adaptive moment estimation (adam). It is more nuanced than the SGD following the update rule shown in Equation 4.7, since it adjusts the needed learning rate during the training steps, based on moving averages of the first and second moments of the gradient. It tries to increase the step size for large plains of low gradients and decrease the step size in anticipation of regions of high gradients. Following the original paper [61], the update rule first calculates biased estimators for the first and second momentum m_t, v_t:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_{\theta_t}\mathcal{L} \tag{4.8}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_{\theta_t}\mathcal{L})^2, \tag{4.9}$$

with $\dot{\beta}_1$ and $\dot{\beta}_2$, determining the rate of decay for previous time steps. Since m_0 and v_0 are initialized as 0, the hereby introduced bias gets removed with:

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{4.10}$$

$$\widehat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{4.11}$$

resulting in a final update of

$$\theta_{t+1} = \theta_t - \eta \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon}, \tag{4.12}$$

with $\dot{\epsilon}$ a factor ensuring computational security and as for SGD $\dot{\eta}$, the learning rate.

4.2.3 Batch normalization

In an MLP the so-called 'internal covariant shift' can occur. It describes, when the average output of a layer for all events shifts on an absolute scale, requiring a re-optimisation of the weights in all consequent layers, that had been optimised to the old range of layer output values. In order to combat this batch normalisation, essentially whitening of the layer output was proposed [62]. Whitening describes the process of setting the mean μ and variance σ of a distribution to zero and one, respectively. For the not normalised outputs x_i the normalised outputs y_i are computed as:

$$y_i = \gamma \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{4.13}$$

In order to reduce computations, this is calculated for each batch and feature individually. The added ϵ ensures computational robustness, as was explained for the adam update rule in Equation 4.12. Furthermore, two additional learnable parameters γ and β are used, so that the layers do not suffer in their representation capabilities.

4.3 Datasets

During training and analysis of a neural network, one should use three separate datasets generally named training dataset, validation dataset and test, inference or analysis dataset. This thesis uses a split of 80:10:10 for the three datasets, applicable to the numbers given in Table 5.1.
The training dataset is used during the training, described in section 4.2. From it, the network learns and generalises. However, at some point in the training, the network will stop learning generalisations about the dataset but instead start to remember each event by some unique characteristic. This behaviour is called 'overfitting' and harms the performance on new data, since those unique quirks in the train data used to memorise, do not translate to new unseen events. In order to start memorising individual events or small groups of events the network assigns small groups of neurons to recognise this pattern. Networks tend to show this behaviour only when they have a high amount of neurons, relative to the complexity and variety of the data, as then reassigning a few neurons from generalisation to memorisation does not hurt the

network's overall capabilities much. In order to combat this behaviour, one can reduce the number of trainable parameters in the network, which forces the network to use more neurons for all events and hinders reassignment. This however can also harm the general potency of the network.
In order to estimate the severance of overfitting occurring in the training, the validation set is used. This dataset contains additional events that have not been used to adjust the weights during training. By calculating the performance of the network on this previously unseen data, an unbiased score can be given to the performance of the network, called the validation score or validation loss. This is typically done at the end of every epoch. First, the validation loss decreases alongside the training loss, benefiting from the generalisation learned at the beginning of the training. If the network then starts to memorise the individual events and gradually loses its generalisation, the validation loss starts to increase. With this, the training can be stopped, as the best possible performance on new data has been reached. One has to ensure, that the increase of validation loss is due to overfitting and not caused by statistical variation. Therefore the training should only be stopped if the validation loss increases over a longer time period. The network of the epoch with the best and therefore lowest validation score will then be used for any further analysis.
Depending on the complexity of the data and the number of trainable parameters, the training of a neural network can last for many epochs. As the optimisation of the network is increasing in smaller and smaller steps the longer it trains, a considerable number of epochs has a more or less equal degree of optimisation with respect to the actual data features. The exact epoch, within the about equally well-optimised epochs, which has the best validation score is mostly determined by statistical fluctuations fitting the validation data. If one were to use the validation dataset also for the actual analysis, the result would be biased by the good performance of the network on the validation data. To remove this bias, the actual analysis should always be done with a third dataset with new unseen events, the inference dataset.

4.4 Graph Neural Networks

From Equation 4.4 we can see that the number of trainable parameters in a dense layer scales quadratically with the number of neurons in the current and previous layers. This can become very computationally intensive very quickly. In order to reduce the computational impact, the exploitation of data inherent symmetries has proven to be a viable approach.

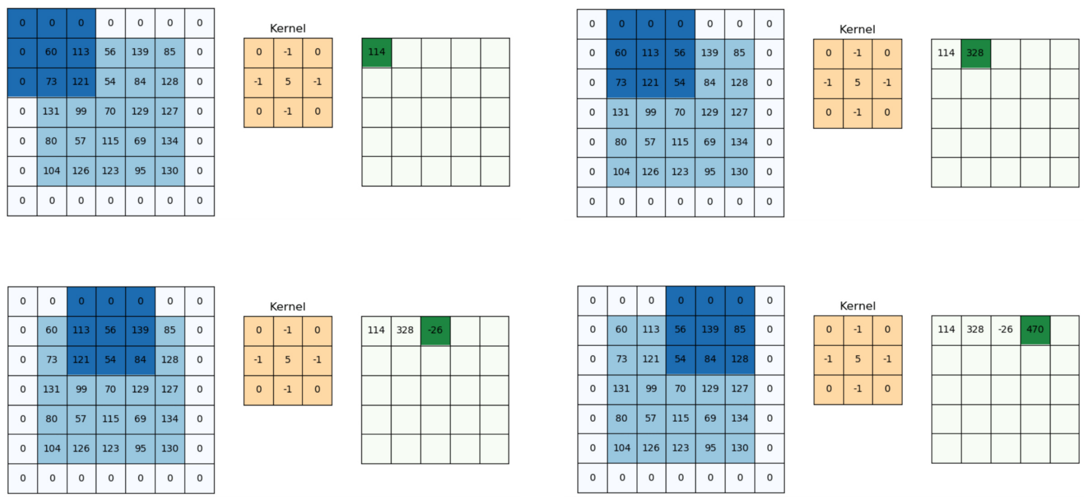


Figure 4.2: *Illustration for the calculation of the output of a convolutional layer, for the first 4 neurons. Nonexistent pixels outside the data frame are padded as 0. From [28]*

4.4.1 Convolutional layer

Utilising translational invariance in data is done by convolutional layers. In a convolutional layer, a neuron is only connected to neurons of the previous layer that are within a specified distance of the original neuron. Therefore, the number of weights is reduced by replacing the dependence on the number of neurons in the previous layer with a dependence on the specified distance of interest. Additionally, the weights used to connect to the previous layer are reused for every neuron in the current layer. This reduces the necessary memory drastically, as the amount of parameters in a layer is no longer dependent on its size. The remaining set of weights used to calculate the output of a neuron is called the kernel with size $\dot{k}$. The size of the kernel determines the range over which information can be exchanged in a single convolution. By using the same kernel for every data point, the same operations are performed, forcing the layer to capture (spatial) patterns, that are applicable throughout the whole input data. An illustration of a convolutional layer output calculation is shown in Figure 4.2.

In order for a convolutional layer to be able to accurately represent the data, the data has to be spatially equidistant. This is trivial for image data, as pixels make up an orthogonal grid, but is to some degree also satisfied by the DU and DOM locations in the KM3NeT detectors, being placed on a regular grid (see Figure 3.3) and a regular vertical placement.

However, this strict requirement of regular equidistant input data also causes problems. First and foremost, the grid of placed DOMs is not exactly regular, either since the floor did not allow the placement of line in a strictly regular grid, or as is the case

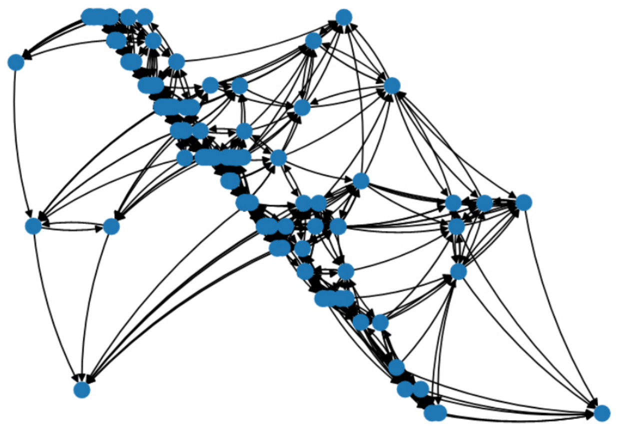


Figure 4.3: *Graph representation of an exemplary down downing ν_μ^{CC}, with location on the vertical axis and time on the horizontal axis. Blue dots are the nodes representing PMT hits. Black arrows show the 8 nearest neighbours of each node.*

of most water-based detectors, the position of the DOMs gets varied on an event by event basis, due to the sea current moving and turning the individual modules. The information on the exact position of the DOMs and PMTs would be lost when using a fixed kernel.
Additionally, the cascade in a neutrino interaction only triggers a small number of PMTs in the detector response. The convolutional layer however requires data for every pixel or DOM/PMT in the detector. This results in so-called sparse matrices as input data, where most entries are zero. In order to be able to connect a substantial amount of active neurons, large kernel sizes are required, connecting the neurons to a high amount of neurons in the previous layer. This negates the advantage of convolutional layers compared to dense layers of connecting only a small amount of neurons in each layer.

4.4.2 Graph convolutions

A good way of addressing the problems of convolutional layers is by extending the concept of kernel convolution from tabular data to graphs. An example of such a graph is shown in Figure 4.3, it shows the only two components of any graph: nodes (blue dots) corresponding to pixels in a classical convolution and edges (black arrows) defining the connections between nodes. Each node represents a PMT hit and the edges indicate the k nearest neighbours of each node, with respect to a given metric.
An implementation of such a graph convolution for KM3NeT has been done by Stefan

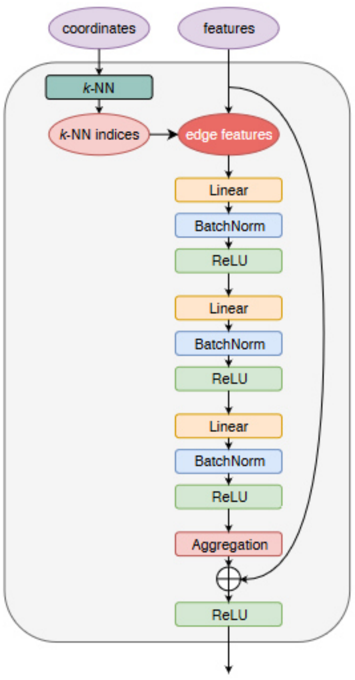


Figure 4.4: *Structure of the EdgeConv block. From [66].*

Reck during his PhD [63], named EdgeConv block [64]. It follows the idea proposed by Wang et al. [65], which has been adopted for particle physics in [66].

A sketch of the architecture of this EdgeConv block is given in Figure 4.4. The block is provided with a list of the features of every hit in an event, as well as a list of all the coordinates of the hits. The list of features contains the most recorded properties described in section 3.1, without the ToT, since it is implicitly contained in the density of hits in an event. The coordinates of the event are comprised of its position and the time of the hits.

To calculate the new features of each node n_i^F the following steps have to be performed. First, the k nearest neighbours for each node are calculated, based on the provided coordinates. The distance between nodes is defined by the Euclidean distance in 4D spacetime:

$$d_{ij} = \sqrt{(\mathrm{p}_{x,j} - \mathrm{p}_{x,i})^2 + (\mathrm{p}_{y,j} - \mathrm{p}_{y,i})^2 + (\mathrm{p}_{z,j} - \mathrm{p}_{z,i})^2 - c^2 \cdot (t_j - t_i)^2}. \tag{4.14}$$

Next, for each of the k nearest neighbours n_j^F the edge feature $e_{i,j}^F$ is defined as a tuple of the absolute value of the original node and the difference to its neighbour:

$$e_{i,j}^{2F} = \left(n_i^F, n_j^F - n_i^F\right). \tag{4.15}$$

Next, each of these edge feature vectors is fed separately to the same MLP, consisting of 3 dense layers with sizes C_1, C_2 and C_3. The output of the MLP for each edge $e_{i,j}^{C_3}$, then gets average into the updated new node features $n_i^{C_3}$.

$$n_i^{C_3} = \sum_{j=1}^{k} e_{i,j}^{C_3} \tag{4.16}$$

Note that the size of the updated node feature vector is no longer determined by the number of measured features as in the original data F, but instead determined by the number of neurons C_3 in the last layer of the MLP.
The last step in an EdgeConv block combines the original node features n_i^F with the updated features $n_i^{C_3}$ via a single dense layer, with C_3 features, retaining the size of the updated feature vector. This is called a shortcut connection.

4.5 Deep learning in KM3NeT

This section gives an overview of the software used within KM3NeT to handle the task of deep learning, as well as a description of the model architecture used throughout this thesis.

4.5.1 OrcaNet

OrcaNet is an internal framework of the KM3NeT collaboration handling various tasks needed for deep learning. It has been developed by Michael Moser and Stefan Reck and is available online under an open source license [67]. It handles a variety of tasks:

- Loading data for training and validation during the training process, as well as inference afterwards
- Saving the model after every completed epoch, together with metrics on learning rate and the loss for training and validation data
- Reading in the needed hyperparameters, model architecture and file paths from three separate toml files. Examples of **config.toml**, **model.toml** and **list.toml** are given in Appendix B.

4.5.2 Particle Net

Together with the EdgeConv block discussed in subsection 4.4.2, the ParticleNet architecture was proposed. It utilises the EdgeConv blocks to build a neural network, that works with the event graphs, a Graph Neural Network (GNN). A sketch showing

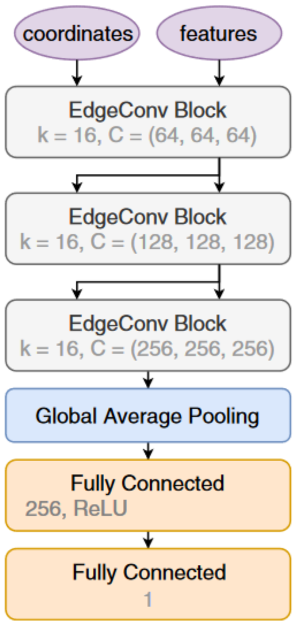


Figure 4.5: *Model architecture used for the networks in this thesis inspired by ParticleNet. Adjusted from [66].*

the model used throughout this thesis, slightly modified from the originally proposed model architecture, is displayed in Figure 4.5. It has been chosen as the model architecture, as it has proven to perform very well in previous works of KM3NeT [68, 69] and other experiments like IceCube [70, 71]. Another promising network architecture are graph based transformers [72], as has been shown by the winning solutions of the kaggle competition issued by IceCube for direction reconstruction in early 2023 [73].

The ParticleNet architecture consists of 3 consecutive EdgeConv blocks followed by a 'Global average pooling' layer and a couple of dense layers designed to handle feature aggregation and output formatting. The second and third EdgeConv blocks receive as input for the feature and coordinate list the output of the previous block $n_i^{C_3}$. This means that the k nearest neighbours algorithm in the second and third block can no longer determine the distance of nodes as described in Equation 4.14, but instead uses the more general form of:

$$d_{i,j} = \sqrt{\langle \delta n_{i,j}^{C_3}, \delta n_{i,j}^{C_3} \rangle} \tag{4.17}$$

with $\delta n_{i,j}^{C_3}$ the element-wise difference of the feature vectors of two nodes n_i and n_j. The size of the internal dense layers stays the same within each block, but doubles for every next block, going from 64 at the first block to 256 in the last.

After the edge convolutions, 'global average pooling' is performed, meaning every feature in the final graph is averaged over all nodes, resulting in a vector of length 256. These neurons are then processed by one hidden dense layer for further calculation until the information gets combined at the output neuron. In contrast to the original paper, the intermediate dense layer, does not have 'dropout' enabled.

5 Data preparation

This chapter will first discuss the production of simulation data for KM3NeT/ORCA. Then, the training process of neural networks, explained in section 4.2, will be expanded on by introducing the concept of weighted training as implemented by Lukas Hennig (LH) for his master thesis [69]. Next, the weight calculation used for this weighted training is explained. The last possible drawbacks of the current weighted training implementation are discussed, together with possible solutions.

5.1 Monte Carlo Data production

Simulating the detector response to incoming particles of various types in different detector conditions is very important. It allows to matching of the raw data recorded by the detector, which by itself is for the most part meaningless, with similar-looking particle cascades of known quantities like energy, arrival direction and flavour. By comparing the recorded events to simulations one can design analysis tools for reconstruction and classification tasks.

The simulations of neutrino in KM3NeT follow the simulation chain illustrated in Figure 5.1. First, 'gSeaGen' is generating a number of neutrinos and if, when and where an interaction with the detector medium occurs [74]. Depending on the energy of the generated neutrino, either 'km3sim' [75] ($E < 100$ GeV) or 'JSirene' [76] ($E \geq 100$ GeV) then calculates the secondary particles produced in the cascade following the interaction, as well as the emitted Cherenkov light by those secondaries and their arrival at the various PMTs in the detector. Next, 'JTriggerEfficiency' [76] samples noise based on the average underwater conditions for any given 'run'. A run describes a time interval of typically several hours, in order to match it to the recorded detector conditions like noise level (section 3.3). Then, 'JTriggerEfficiency' simulates the different trigger responses of the detector to the incoming signals originating from the

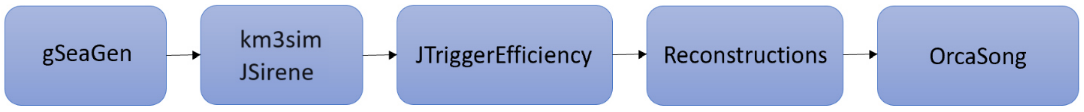


Figure 5.1: *Simulation chain in KM3NeT for GeV neutrinos. From [69].*

neutrino and noise and saves the events as root files.
As for the real recorded data, now various analysis tools are run on the individual events, reconstructing for example energy or direction, and adding them to the saved root files. In order to be able to use them with Graph Neural Networks (section 4.4), they need to be converted into a suitable tabular format, representing events as Graphs, instead of the tree-like structure of the root. This is done by applying h5extract2 of the km3pipe module [77] and the for the particle type adequate extractors provided by the python module OrcaSong [78].

Furthermore, whole analyses are optimized for large sets of simulations, before running the final analysis on previously unseen real data. The process of using new unseen data is called 'unblinding' and ensures, that analysis does not infer biases into the tested hypothesis, based and the statistical fluctuations of the real dataset. This optimisation however needs very accurate simulations, so the analysis gets optimised for the actual data features and not simulation bugs. In order to confirm the quality of the simulations and how well they match the real data, a Data/Monte Carlo ratio (DMCR) can be calculated. For this, one needs to run the analysis on matching amounts of simulations and real events and then compare the distributions of reconstructions to each other. If the distributions match, the simulations are good, if not, this should cause investigations, as to how this discrepancy between the simulation to the real data is caused. Importantly, even unoptimised analysis should result in matching statistics, as they should reconstruct the real and simulation data equally wrong.

The number of simulated events used in this thesis, separated for interaction type and dataset is given in Table 5.1. Interaction type here specifies, what flavour the neutrino has, whether the neutrino is a particle or antiparticle and which gauge bosons mediated the interaction (see chapter 2). The dataset specifies whether the events were simulated in either a slight variation in the production of these types of events, either run-by-run (rbr) or single-run (sr) production chain. A energy resolved visualization is attached in Appendix A.
Run-by-run here refers to the default simulation procedure, where simulations are produced in smaller quantities for every detector condition measured and therefore all runs, containing events with varying noise levels. In addition to the rbr events, Lukas Hennig produced an additional dataset more suited for the tau classification done in his thesis [69]. This dataset is produced entirely with the noise level of run 8022, showing a low level of noise. The events of this dataset have an equal share of tau to non-tau events and a flat energy spectrum, which removes the possibility, that a network predicts the interaction type, based on event statistics and not features in the event graphs. Non-tau events refer to any events that are not a tau-CC interaction.

Interaction type		Dataset	
Particle	Channel	run-by-run	single-run
ν_e	CC	359020	240005
$\overline{\nu}_e$	CC	434528	240655
ν_μ	CC	255258	239517
$\overline{\nu}_\mu$	CC	320625	239894
ν_τ	CC	202070	712323
$\overline{\nu}_\tau$	CC	203154	709168
ν_μ	NC	426877	235060
$\overline{\nu}_\mu$	NC	254701	231898

Table 5.1: *Overview of the number of simulation events used, separated for interaction type and dataset. For visualisation see Appendix A.*

Even though antineutrinos are simulated separately from normal neutrinos, neutrino telescopes such as KM3NeT are not capable of differentiating between and can only differentiate between electromagnetic showers, hadronic jets and muon tracks. For this reason, the eight different interaction types listed in Table 5.1 can be sensibly grouped into five different datasets. Three datasets, containing only ν_e^{CC}, ν_μ^{CC} or ν_x^{NC} events. Furthermore, a dataset including ν_e^{CC} and ν_x^{NC} events is created, typically called the shower dataset, as both topologies look to the detector shower-like. The fifth dataset additionally includes the track-like ν_μ^{CC} neutrinos and the ν_τ^{CC} of all decay channels. Unfortunately, ν_τ^{CC} are not tagged for their decay process during simulation, which makes it impossible to include them in any of the datasets, except the one including all events.

5.2 Weighted training

Using the standard method of training explained in section 4.2 only allows to train on the native statistics given by the training dataset. In order to train on a dataset with different statistics, the dataset would need to be cut off from certain events or filled with duplicates. In the first option, the network would lose statistics for its training and become less representative of the underlying data, while the second option would balloon the file size of the training files. An alternative to this is given by weighted training, which assigns a weight to each event, effectively resulting in a network trained on different statistics.

The weighted training performed during this thesis was introduced to the training, by adjusting the loss calculation. As described in subsection 4.2.1 the training tries to minimize the average loss $\mathcal{L}$ over the events $(x_i, y_{i,\text{true}})$ with respect to its weights θ:

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_{i,\text{true}}, f(x_i, \theta)) \tag{5.1}$$

To include an individual weight w_i for each event $(x_i, y_{i,\text{true}})$, the expression for the minimisation problem can be adjusted to:

$$\min_\theta \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \cdot \mathcal{L}(y_{i,\text{true}}, f(x_i, \theta)) \tag{5.2}$$

With this weighted loss, each event is now only contributing to the final average loss proportional to its weight. Additionally, the normalisation has been adjusted to account for the fact, that the total contribution of N events is no longer N, but the total weight of all contribution events.
This way of implementing weighted training has been chosen since it was already implemented by LH for his master's thesis. Possible problems caused by this implementation, as well as other ways of implementing weighted training, will be discussed in section 5.5.

5.3 Physical simulation weight

In order to be able to compare the simulations to real data, the simulated events need to be weighted as they are expected to be measured by the detector. This weighting is called physical, as it represents the flux one would expect based on the current physical knowledge. The calculation of these physical simulation weights $w_i^{(\text{sim})}$ for a simulated neutrino with index i depends on several factors, namely the expected flux at the neutrino site calculated theoretically and various other factors like the interaction cross section and the detector response to the neutrino, that are determined during the simulation of a given run. In the 510 days, the ORCA detector had 6 DUs, and the run_ids 7224 to 11293 have been issued. The exact formula for the calculation of the physical simulation weights $w_i^{(\text{sim})}$ for neutrinos simulated by KM3NeT is given by

$$w_i^{(\text{sim})} = \varphi_i^{Dec} \cdot \frac{t_i}{n_i^{gen}} \cdot w2_i. \tag{5.3}$$

As mentioned in section 2.3, KM3NeT assumes the Honda atmospheric neutrino flux model for its purposes. The exact values of the flux are binned for energy, azimuth and zenith of the arriving neutrinos and averaged over the summer of 2014. Other neutrino sources than our atmosphere can be neglected as they do not play a significant role in the relevant GeV energies. In order to get the neutrino flux at the detector φ_i^{Dec}, the

oscillation probabilities of the atmospheric neutrinos have to be calculated and applied to the atmospheric flux. The probabilities dependent on the L/E of the neutrino are calculated by the OscProb module [79]. In order to determine the travel distance of the neutrino, the zenith angle can be used in combination with the diameter of the earth.
The simulation part of the weight calculation consists of three values. t_i, the length of the run in seconds, called its 'lifetime' for which the event has been simulated. n_i^{gen}, the number of neutrino interactions inside the detector volume simulated for the respective run. Together these two normalisation factors match the number of generated events with the duration of the run, allowing for the generation of an arbitrary number of events. This is for example used in the additionally generated single-run dataset by Lukas Hennig [69]. At last $w2_i$ combines various factors of the detector response for the simulated events, like for example the "average interaction cross-section per nucleon along the neutrino path through the Earth". A detailed description of the various contributing factors can be found in [74].

5.4 Sample weight options

In order to calculate the individual weights for the simulations, I programmed a Python script that enables the user to set various parameters in the weight calculation for the three independently calculated categories of source datasets, interaction type ratios and spectrum (energy, azimuth and zenith). In order to be able to adjust the options independently the respective weights $w_i^{(\mathrm{ds/it/sp})}$ are normalised and at the end multiplied together to form the final event weight $w_i^{(\mathrm{f})}$. The formulas provided in the following do not necessarily match the exact way the script works but are instead presented in such a way that the effective calculation scheme can be explained more easily. Each sample weight option will now be explained separately, while simultaneously a notation convention for the weight options of a given dataset and the network trained with it will be established.

5.4.1 Source datasets

The source data consists of the two datasets mentioned in Table 5.1:

- run-by-run
- single-run

A ratio of $x \in [0, 1]$ describes the total weight of sr events in the data of the respective trained network. The share of sr events in the respective dataset will be noted on the top left of the dataset: ${}^{0.8}$dataset. Alternatively, if the dataset contains only one dataset, the name of it will be given instead of the ratio: ${}^{\mathrm{rbr}}$dataset.

5.4.2 Interaction types

As discussed in subsection 2.4.2, the available neutrino data consists of neutrinos with the following interaction types:

- ν_e^{CC} (electromagnetic shower)
- ν_μ^{CC} (track)
- ν_x^{NC} (hadronic jet)
- ν_τ^{CC} (varied)

Since neutrino telescopes in general and therefore also KM3NeT/ORCA cannot determine, if an interaction has been induced by a antineutrino or not, both particles are treated together.
The ratio of events belonging to the different interaction types can be changed from a state of flat weight to a state of physical weight (section 5.3). The state of flat weight $w_i^{\text{(it, flat)}}$, where each interaction type contributes equally, is calculated as

$$w_i^{\text{(it, flat)}} = \frac{1}{n_I} \cdot \frac{1}{N_I}, \tag{5.4}$$

with N_I the total number of events with interaction type I and n_I the number of different interaction types in the data. For the full dataset this is 4. This normalises the total weight over all events to 1.
The state of physical weight $w_i^{\text{(it, phys)}}$ is calculated as

$$w_i^{\text{(it, phys)}} = \frac{1}{N_I \cdot \sum_i w_i^{\text{(sim)}}} \cdot \sum_{i \in I}^{N_I} w_i^{\text{(sim)}}, \tag{5.5}$$

with $w_i^{\text{(sim)}}$ the physical simulation weight discussed in Equation 5.3. By dividing the events by the total simulation weight over all events and the number of events with the respective interaction type N_I, the total weight is again normalised to 1.
Combining both states of weight into a single final interaction type weight $w_i^{\text{(it)}}$ is given by the superposition:

$$w_i^{\text{(it)}} = w_i^{\text{(it, phys)}} \cdot r + w_i^{\text{(it, flat)}} \cdot (1 - r), \tag{5.6}$$

with $r \in [0, 1]$ describing the Physical weight share (PWS). r will be displayed on the top right of the dataset. ${}^{0.8}$dataset${}^{0.4}$ for example has 80% sr events and a PWS for interaction types of 40%. Alternatively, as for the source datasets, if the dataset

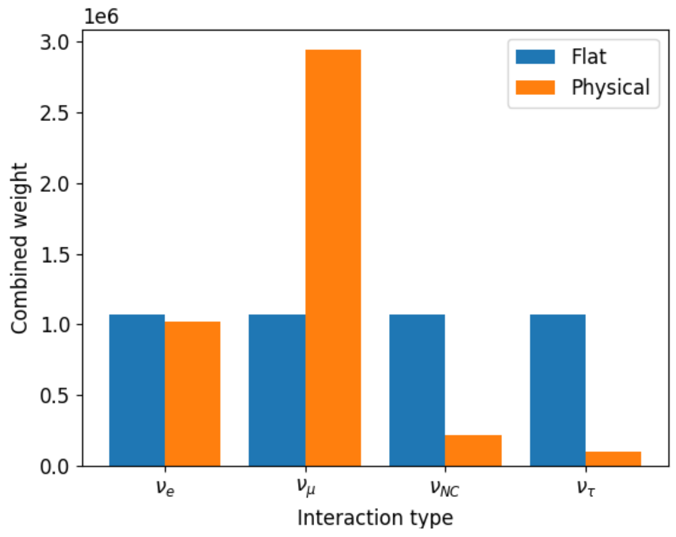


Figure 5.2: *Combined weight for interaction types in the case of physical weighting (orange) and flat weighting (blue). Notation follows subsection 2.4.3.*

contains only one interaction type, the name of it will be given instead of the ratio: ${}^{0.8}\text{dataset}^{\nu_e}$.

For the extreme cases of complete physical weight or complete flat weight, the total weight of the interaction types during training is shown in Figure 5.2. The physical weighting shows a high content of ν_μ^{CC} and a low content of ν_x^{NC} and ν_τ^{CC}.

5.4.3 Spectra

As a third option, the spectra weight $w_i^{(\mathrm{sp})}$ of energy, zenith, and azimuth can, similar to the interaction type weight, be changed from a flat weight to a physical weight. For now, the three spectra can only be tuned together by a single value. However, if desired a possibility for independent adjustment can be implemented. As the simulation weights $w_i^{(\mathrm{sim})}$ are already calculated for the energy and angle of the individual events, the translating to the physical state in the script requires only normalisation:

$$w_i^{(\mathrm{sp,\ phys})} = \frac{1}{\sum_i w_i^{(\mathrm{sim})}} \cdot w_i^{(\mathrm{sim})}. \tag{5.7}$$

In order to calculate the flat weights, the phase space of the events is binned for energy zenith and azimuth and each bin B calculates its weights as:

$$w_i^{(\mathrm{sp,\ flat})} = \frac{1}{n_B} \cdot \frac{1}{N_B}, \tag{5.8}$$

with, comparable to the flat interaction type weights, N_B the number of events in the respective bin B and n_B the number of bins. They are then combined in the final weight by

$$w_i^{(\text{sp})} = w_i^{(\text{sp, phys})} \cdot r + w_i^{(\text{sp, flat})} \cdot (1 - r), \tag{5.9}$$

with the PWS $r \in [0, 1]$. It will be displayed in the bottom right corner of the dataset: ${}^{0.8}\text{dataset}^{\nu_e}_{0.6}$ for example has a PWS of 60% with the remaining 40% as flat weight. The Physical weight share (PWS), is used for both, interaction types and spectra weights in the next section, however it will be clear from context, which of both is meant.

5.5 Implementation drawbacks

The chosen implementation of the weighted training, discussed in section 5.2, has primarily been chosen by Lukas Hennig as it was already somewhat implemented during the OrcaNet development. Due to this in discussions with colleges at ECAP and the KM3Net Oscillation working group, potential problems and solutions have been identified. However, one has to note, that none of the following issues have been tested and it is unclear as to how much these problems reduce the general performance of the neural networks, due to the concrete choice of implementation.

5.5.1 Dominated batches

Figure 5.3 shows the distribution of sample weights for the five different datasets. All show a range of sample weights of at least 10^4. With the batch sizes of 32 or 64 in the trained networks, this range of weights could cause the weight update within a batch to be dominated by a very small number of events in that batch removing the averaging and therefore increasing the possibility of adjusting in a counterproductive way. This varies the effective batch size during the training on a batch-by-batch basis, resulting in a very jagged optimisation.

Increase batch size The most direct approach to tackle this problem would be to increase batch size. A larger batch size would then enable more averaging in general, reducing the amount a single batch gets dominated by a single event. Independent of this concrete problem setting the batch size as large as possible, up to the limit of the VRAM of the GPUs is recommended by a group of Google engineers [80]. On the other hand, a very large batch size is generally associated with less generalisation ability of a neural network.

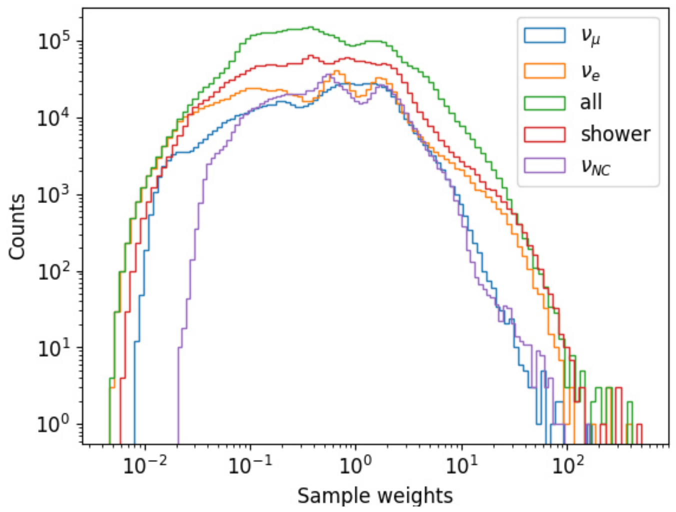


Figure 5.3: *Sample weight distribution for all 5 interaction type combinations datasets tested in section 7.2. Notation follows subsection 2.4.3.*

Sample batch according to weight A more fundamental approach would be to replace the implementation of weighted training completely, by removing the weighted loss function and instead applying the weighting during the sampling process of the batches. By sampling in a probabilistic way, events with higher weight would be trained on more often, but each batch would still weigh all events within it equally.

Fixed batch size By changing to a true probabilistic sampling method, the standard definition of an epoch falls short, requiring that every event in the dataset be trained at once, as that would result in extremely long epochs. Even if one would enforce a loaded sampling keeping pure randomness in check it would require the higher weighted events to be sampled hundreds of times, until every event is trained on. Instead, one would need to redefine an epoch to finish at a certain number of events or batches. This could be as simple as the total number of events in the training set. However, as this number does not influence the training itself, establishing a value independent of the dataset, it would present an option to establish a standard throughout KM3NeT to ensure better comparability of different neural networks, independent of the dataset sizes. This concept, of evaluating after a fixed amount of training samples, is already standard in the training of LLM transformers [56] like GPT [57], where training is done on an enormous amount of text tokens in various combinations.

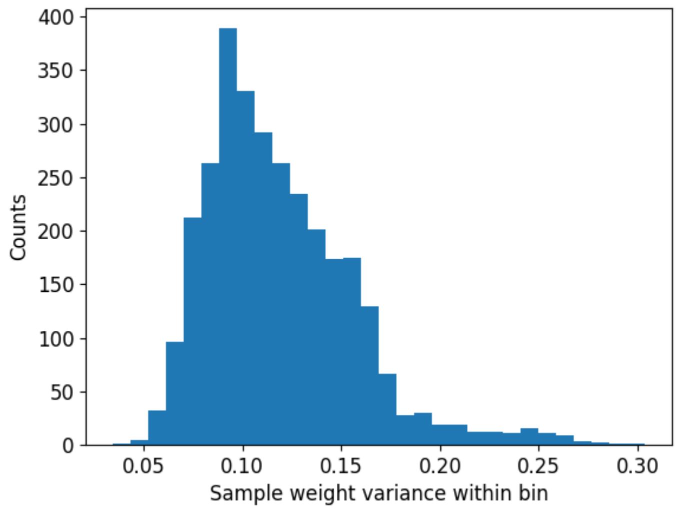


Figure 5.4: *Variance of sample weights with in a phase space bin of log10(energy* , coszenith and interaction type. Variance in all events is 0.55.

5.5.2 Reduced topology variance

A problem independent of the current implementation is caused by the physical simulation weights explained in section 5.3. The simulation weights do include a variety of factors, which only are required in order to adjust the actual training statistics. This results in some events being weighted less, than others, even though they are located in the same place in the phase space. If the weights with similar attributes vary, the network will try to improve on only the most influential events, since the others do not contribute much to its loss. This is essentially removing particle cascade variety from the training dataset, reducing the network's incentive to generalise and effectively train it on a smaller dataset. Weights that vary due to the desired rebalancing of dataset statistics for example energy are necessary and not in question here.

To show the amount of unnecessary weight differences, all available events are binned into 20 logarithmic energy bins from 1 GeV to 100 GeV, 20 linear coszenith bins, the 8 interaction types given in section 5.1. Azimuth and run_id are not considered in the binning, as they are already produced in about equal amounts and are therefore not as important to reach the desired data statistics. Including these features in the binning, would reduce the number of events in the actual bins, increasing statistical fluctuations. Figure 5.4 shows the variance measured in each of the 3200 bins. For comparison the variance in the dataset of all events shown in Figure 5.3 is 0.55. Even though the

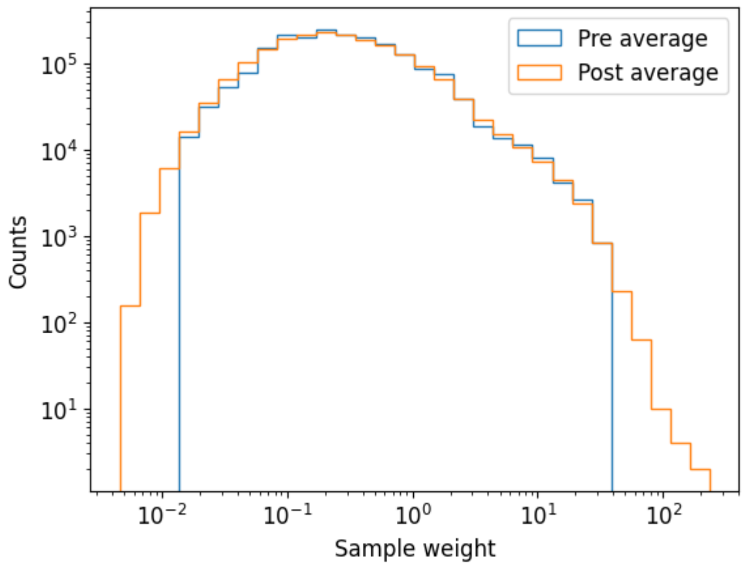


Figure 5.5: *Sample weight distribution for all events before (blue) and after (orange) proposed phase space bin averaging.*

average variance within a bin is only about 20% of the total dataset variance, it still makes up a significant amount of it.

A possible approach to this would be to average the weights within a given phase space bin after the initial weights were calculated. This would keep the weights of the target distribution, but eliminate all other weight differences. For this to work properly one would need to think about, which parameters can be averaged over and which can not. Additionally, a good procedure for finding the boundaries of each phase space bin would be needed. Using the phase space binning described earlier the sample weight distribution of the full dataset before and after averaging can be seen in Figure 5.5. While the distribution is relatively consistent with and without averaged weights, the most extreme values are eliminated, reducing the range of weights over one decade.

5.5.3 Test setup

In order to test for these problems, I propose two options: Implement a class for my sample weight script, that would allow for the definition of arbitrary spectra (energy, zenith, azimuth), after applying the original weights. This would allow one to create a dataset with the same distribution, as the unweighted dataset, while keeping the sample weight problem caused by the the physical weights calculations. Alternatively one could apply random weights, similar to the weighted distribution, to the unweighted

dataset.
Both options would result in two datasets of identical target distribution, with one of them having assigned weights. Comparing networks trained with these two datasets should give a good handle on the influence the large weight range has since the benefits of the weighted training are being removed.

6 Manual hyperparameter optimisation

As this thesis investigates the effects of weighted training within the general deep learning efforts of KM3NeT, first a general agreement for the performance and behaviour of my networks compared to the previous results of Daniel Guderian (DG) [68] and Lukas Hennig (LH) [69] are established. This ensures that my results are also true for other KM3NeT deep-learning applications. The comparison is especially important since I use slightly different versions for the simulation data as DG and do energy regression instead of tau classification (LH).
In order to achieve the just mentioned, this chapter first discusses the way networks will be scored during this thesis. Next, an estimate of the error of networks will be calculated, followed by a discussion on the results of the manual hyperparameter optimisation of various datasets. All networks that are discussed in this chapter have been trained without weighting and all datasets from Table 5.1.

6.1 Network score

As already established in section 4.3, for the actual evaluation of the networks the test dataset is used. In order to be able to easily compare networks, the calculated network score should be a single number. How to calculate the score from a set of individual events, depends on two factors. First, how to calculate the individual event errors e_i from the event's true reconstruction quantity $\mathrm{y}_{i,true}$ and its corresponding prediction by the network $\mathrm{y}_{i,pred}$. Second, as to how these event errors are combined into the final network score. Both factors will be discussed in the following.

6.1.1 Event error

A network score is typically calculated as the median or mean of the distribution of its absolute errors $|e_i|$. One should however only use the absolute errors, if the distribution's shape of both positive and negative errors is similar. This is not the case when evaluating networks trained with the OrcaNet framework for energy reconstruction and evaluating them on the standard event error used for likelihood and deep learning-based analyses in KM3NeT. By using the relative fractional error

(RFE):

$$e_{i,RFE} = \frac{y_{i,true} - y_{i,pred}}{y_{i,true}}, \tag{6.1}$$

on networks trained on the $log10(energy)$, an asymmetric error distribution emerges as shown in Figure 6.1, evaluating under and overestimation on different scales. On the one hand in the network training a mean squared error of the $log10(energy)$ is used, limiting the valid energy range to]0, ∞[. This is already the case for the real physical neutrinos, but only gets enforced to the network by the logarithmic conversion. Additionally, the training on the $log10(energy)$ is necessary, in order to preserve sensitivity to lower energies, while simultaneously training on multiple decades of energy. On the other hand, the RFE assumes a valid number range of]-∞, ∞[, resulting in the cut-off at -1, since the network can only guess positive values for the energy, which is 0+ as the lowest value and therefore underestimation by 100%, hence an RFE of -1.

By taking the absolute error values and thereby superimposing the errors of this asymmetric distribution, the impact of underestimation on the final score is reduced compared to overestimation since it does not contain high values. In fact, about 10% of the errors are greater than 1, and no error is smaller than -1. A possible solution to this is to use the same type of error for the evaluation as during the network training, a logarithmic error (LE):

$$e_{i,LE} = log10\left(\frac{y_{i,true}}{GeV}\right) - log10\left(\frac{y_{i,pred}}{GeV}\right) = log10\left(\frac{y_{i,true}}{y_{i,pred}}\right). \tag{6.2}$$

The error distribution resulting from the LE on the same exemplary network as used for the RFE in Figure 6.1 can be seen in Figure 6.2. It results in a much more symmetrical error distribution, allowing for the reasonable use of the absolute error distribution to calculate the network score.

However, since the LE is not used in any other KM3NeT analysis, and results in different absolute values for the network score than the RFE, it cannot be used for comparisons of this analysis with other energy reconstructions. For that reason, internal comparisons of results within the thesis will be using the LE, while comparisons to other results will be done with the RFE.

6.1.2 Network score

The process of reducing a set of event errors into a single network score is defined by two aspects, the summary statistic like mean or median, and whether the events are weighted with a flat spectrum, giving each bin in the phase space equal contribution or weighted physically, resembling the anticipated neutrino spectrum measured in the

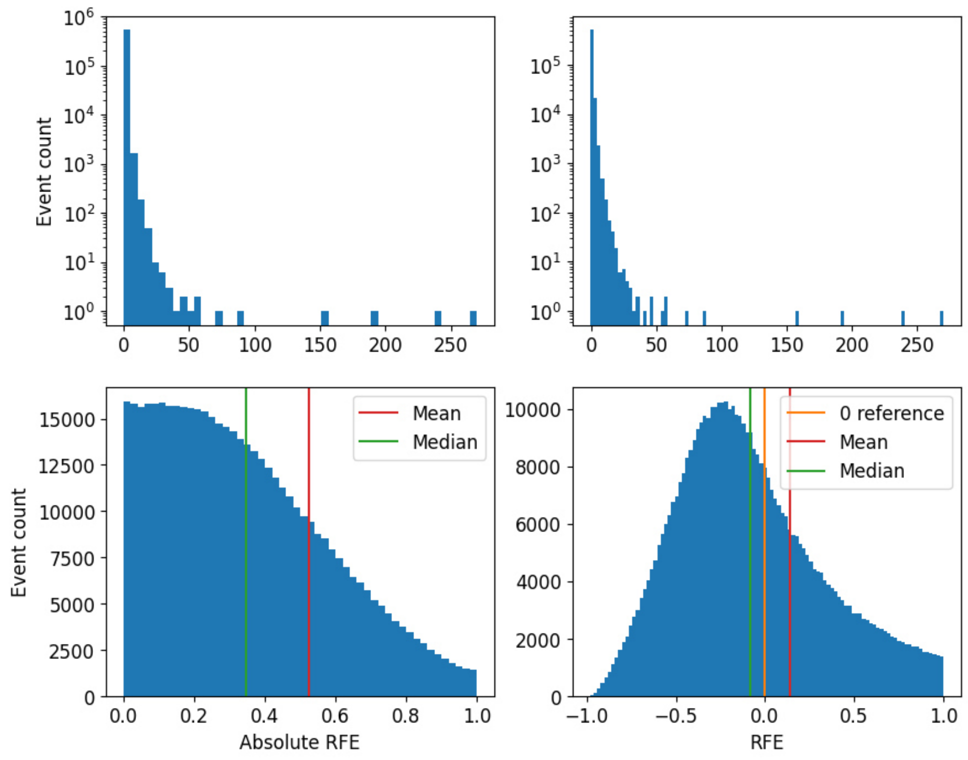


Figure 6.1: *Relative fractional error distribution of* $^{0.5}\text{network}^{0.5}_{0.5}$. *The error is of unit 1 (see. Equation 6.1). Top: Histogram of whole test dataset. Bottom: Zoom of top row, with a cutoff at an absolute error of 1. Mean and median are calculated for the whole dataset, not only the plotted part. Bin width is equal for left and right in the bottom row. Left: Absolute error of Equation 6.1. Right: Error as in Equation 6.1.*

detector as explained in section 5.3.
Even though the mean is the used summary statistics to describe the average of a distribution, it misrepresents most unsymmetrical and skewed distributions, as is the case for the absolute errors shown in the lower left of Figure 6.1 and 6.2. A much better choice is the median, which is generally more resistant to single outliers, as well as the current standard for analyses in KM3NeT.
The question of weights unfortunately is much more complicated. At first glance using the expected neutrino flux in the detector seems obvious since what we in the end want to perform well on is real data. However, there are two major issues with physical weighting.
Independent of which physical weighting we apply to the events, we initially have to choose a atmospheric flux model to use for our weighting and as explained in section 2.3, there are still open questions regarding the flux of atmospheric neutrino. By weighting

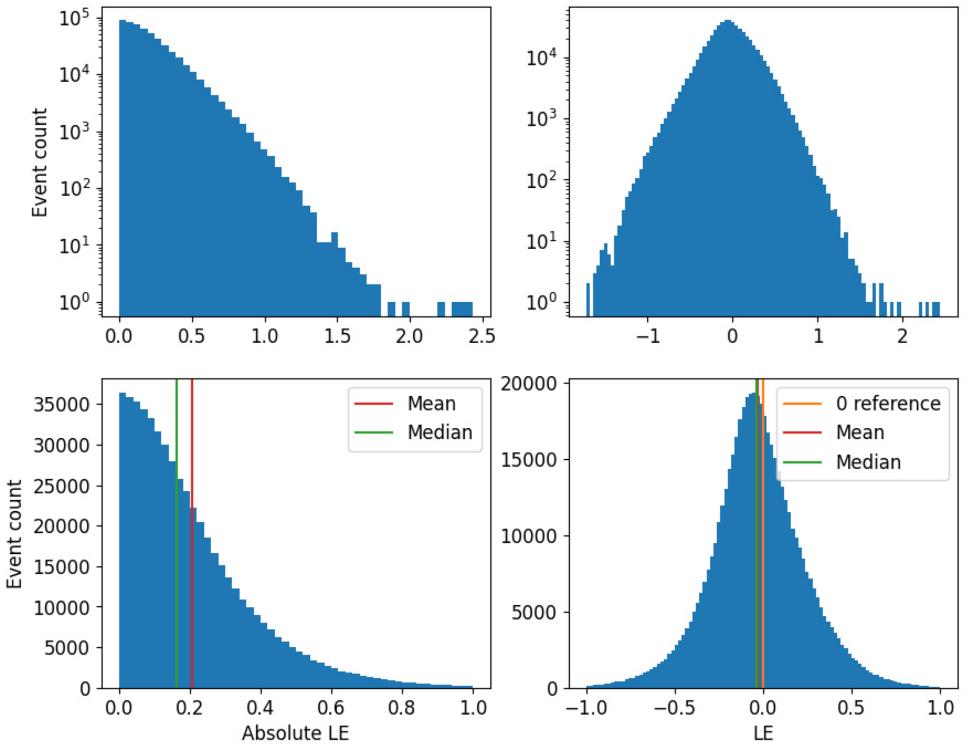


Figure 6.2: *Log10 error distribution of $^{0.5}\text{network}^{0.5}_{0.5}$. The error is of unit 1 (see. Equation 6.2). Top: Histogram of whole test dataset. Bottom: Zoom of top row, with a cutoff at an absolute error of 1. Mean and median are calculated for the whole dataset, not only the plotted part. Bin width is equal for left and right in the bottom row. Left: Absolute error of Equation 6.2. Right: Error as in Equation 6.2.*

with a specific model, the uncertainties of the model would be transferred to the error of the network.

The next problem is caused by different neutrino flavours occurring in the data for given energies. ν_τ^{CC} interactions can only occur at energies over approximately 3 GeV, due to the high rest mass of the τ lepton of approximately $1.78\,\text{GeV}\cdot\text{c}^{-2}$. Therefore, the influence on the network score by events with energies below 3 GeV, is only made of a combination of ν_e^{CC},ν_μ^{CC} and ν_x^{NC}, as is shown in Figure 6.3. How this can lead to unintuitive results is shown in Figure 6.4. Here the performance of ν_x^{NC} is on average better, than the other interaction types, while simultaneously performing worse than the other types at every data point, where data for both interaction types is available. The average ν_τ^{CC} performance is not influenced by the general bad energy reconstruction below 3 GeV, which drags down the average performance of the other

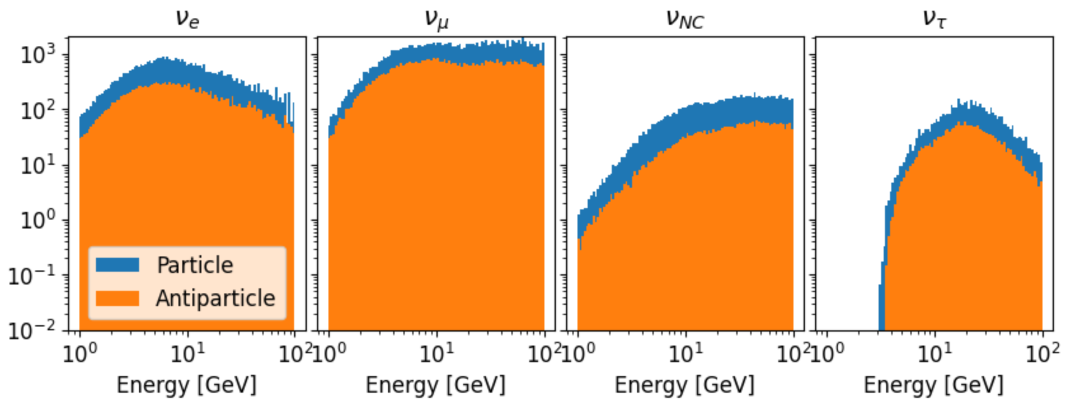


Figure 6.3: *Event occurrence for rbr events, separated by interactions types and weighted physically as discussed in section 5.3. Notation follows subsection 2.4.3.*

interaction types.

For these reasons, I will use a flat weighting for the calculation of network scores in this thesis, providing base score with hopefully little bias. Since a physical weighting still can provide additional information, it will be shown in addition to the flat weighted network score.

6.2 Estimate for network uncertainty

In this section, the statistical variance in network performance is estimated. To test this, seven neural networks have been trained with an identical set of hyperparameters given in Table 6.1 and the model architecture discussed in subsection 4.5.2. It deviates from the best hyperparameter set found by DG for the energy reconstruction of his PhD thesis only for the number of k-nearest neighbours, which has been reduced to 16 in order to save on computing power. This is reasonable since the goal of the MHPO in this chapter is not to have a detailed analysis or find the absolute best hyperparameter configuration, but only to establish a general agreement with the previous works.

The flat and physical weighted network score for all seven networks is plotted in Figure 6.5. For the flat-weighted LE a standard deviation of 0.007 is calculated, which corresponds to an error of 5%. The physical weighted LE has a lower relative deviation of 2% with a standard deviation of 0.002. The lower spread for the physical weighted score makes sense since the selection of the final epoch is based on the validation loss, which itself is calculated with physical weighting. The hereby determined standard

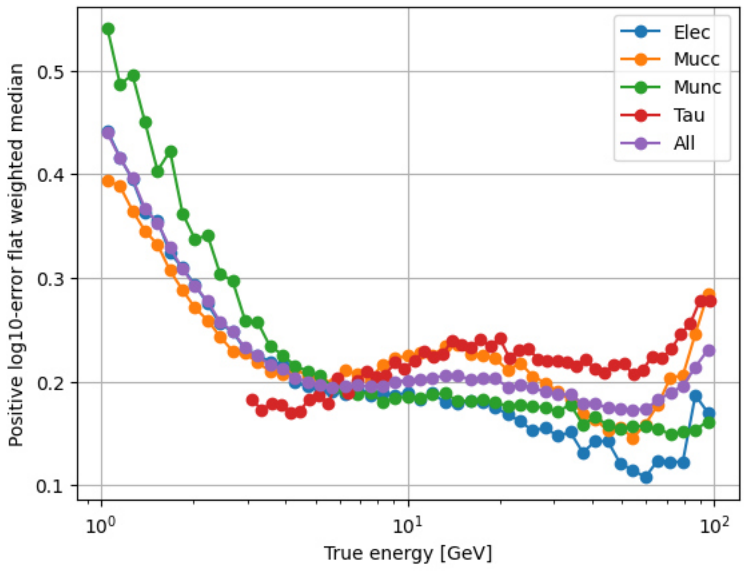


Figure 6.4: *Plots shows the bad performance of low energy events, separated for the different interaction types. Each point is the median of the absolute errors weighted flat for events with its respective interaction type and energy bin. The event errors are calculated according to Equation 6.2.*

Hyperparameter	Value
Learning rate	0.005
batch size	32
k-nn	40
Activation function	Relu
Loss	LogNormal
Optimiser	Adam
ϵ (Adam)	0.1

Table 6.1: *Default hyperparameter values for MHPO*

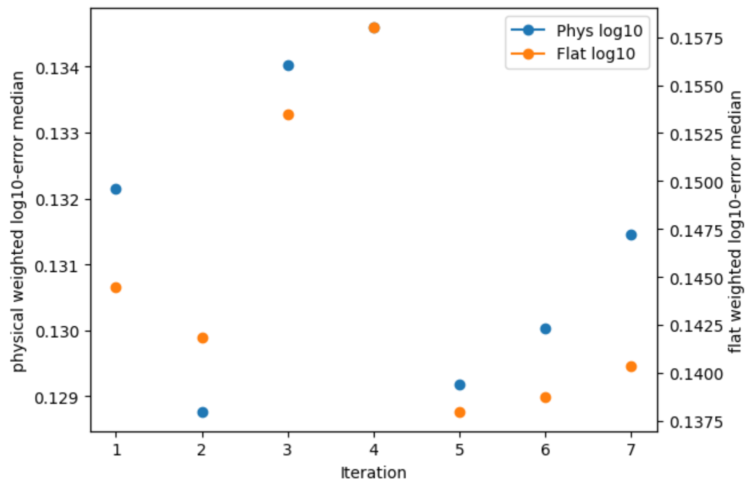


Figure 6.5: *Physical and flat weighted LE network scores for 7 networks trained with identical configuration given in Table 6.1.*

deviations will be used as the approximation for the statistical error in the network evaluation from now on.

6.3 Basic hyperparameter optimisation

This section presents the results of the initial MHPO. The networks are all trained with the same hyperparameter as in section 6.2. Since the different tests were run in parallel, the optimised parameters are not applied in the subsequent subsections. The networks have only been trained with and scored on ν_e^{CC} data, to ensure a converged network within 24 hours. This limitation is chosen for the same reasons as the reduction of k discussed in section 6.2.

Adam optimizer The first hyperparameter to be discussed is ϵ, the factor responsible for numerical stability in the adam optimiser as explained in paragraph 4.2.2. Its performance over the range from 1e-8 to 0.5 is shown in Figure 6.6. The best results are produced around a value of 0.1. This is the same value as in the initial parameter set from Table 6.1 and the default parameter in OrcaNet. It has to be noted that this value differs drastically from the recommendation of 1e-8 given by the authors in the original paper [61].

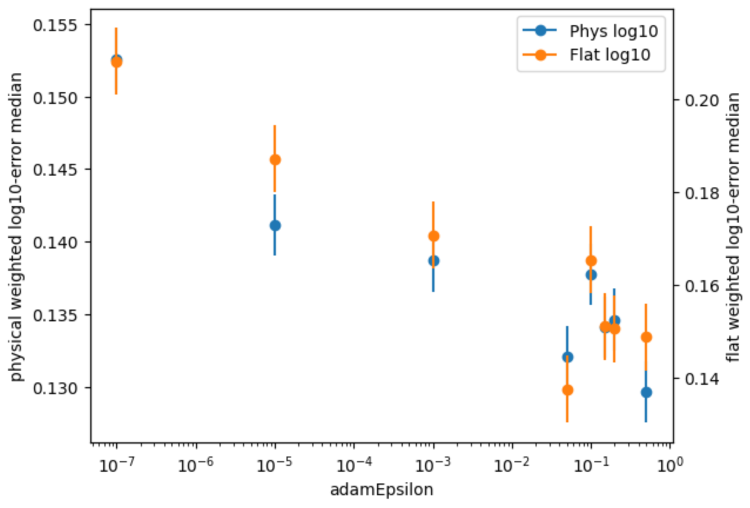


Figure 6.6: *Physical and flat weighted LE network scores for the adam optimiser parameter ϵ for values between* 1×10^{-8} and 0.5

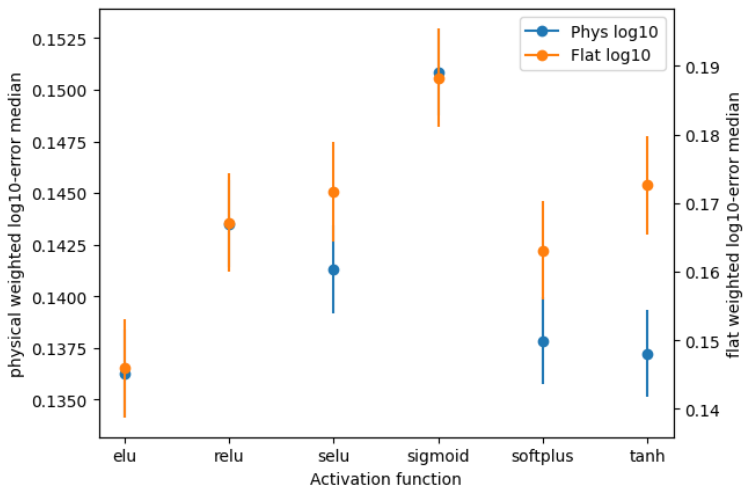


Figure 6.7: *Physical and flat weighted LE network scores for 6 different activation functions of the EdgeConv blocks.*

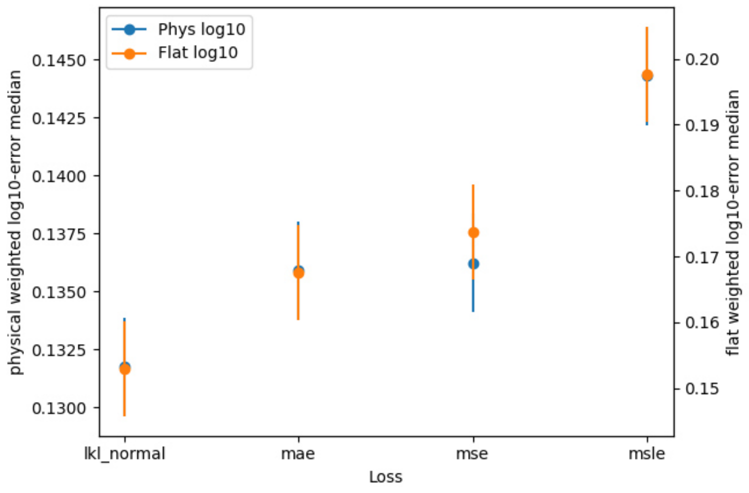


Figure 6.8: *Physical and flat weighted LE network scores for 4 different losses.*

Activation function In addition to the 2 activation functions defined in subsection 4.1.1, elu, selu, tanh, and softplus have been tested as the activation functions used within the EdgeConv blocks of the neural network. The 4 additional activation functions are given by:

$$\text{Elu: } \phi(x) = \begin{cases} x, & \text{if } x \geq 0, \\ \alpha \cdot (e^x - 1), & \text{otherwise,} \end{cases} \tag{6.3}$$

$$\text{Selu: } \phi(x) = \lambda \cdot \begin{cases} x, & \text{if } x \geq 0, \\ \alpha \cdot (e^x - 1), & \text{otherwise,} \end{cases} \tag{6.4}$$

$$\text{Tanh: } \phi(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{6.5}$$

$$\text{Softplus: } \phi(x) = \ln(1 + e^x). \tag{6.6}$$

From the comparison presented in Figure 6.7, we can see that the best performance is achieved by 'elu'. Even though its performance is significantly better than the default relu activation function, no change will be done for further analysis, in order to preserve comparability with DG in the analysis presented in chapter 7.

Loss Furthermore, the losses defined in Equation 4.6 have been tested and are shown in Figure 6.8. The best performing loss has been found to be the reconfigured log_normal loss. However, also the mean squared and mean absolute error show a good performance.

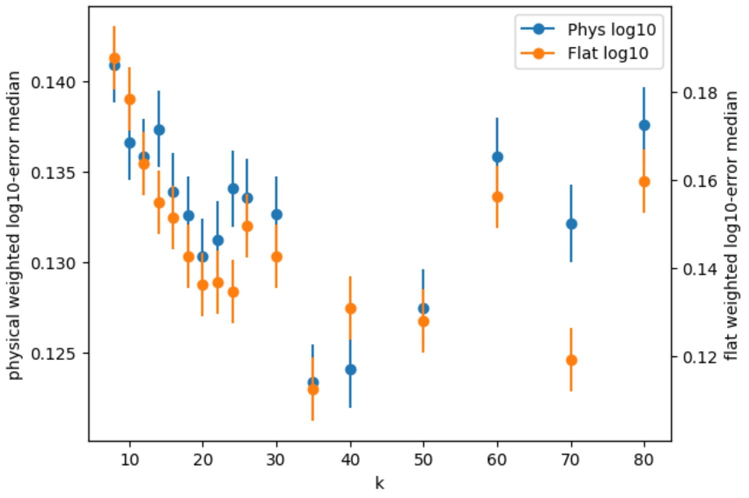


Figure 6.9: *Physical and flat weighted LE network scores for k nearest neighbours from 8 to 80.*

K nearest neighbours Increasing the number of nearest neighbours k, results in more information gain of the network per epoch, as already found by DG, since the network can exchange more information over its edges simultaneously and more importantly with more distant nodes. However, it also increases the needed computing time per epoch, since more edges have to be calculated in every convolution. The limiting factor in the training of neural networks is not the number of epochs, but the time needed to gain this performance. As for the other hyperparameters, the performance after 24 hours of training is evaluated in Figure 6.9. It can be seen that increasing k results in a better performance, with the best results around $k = 40$. After that, the network performance decreases, due to the increased time spent in the nearest neighbour algorithm and consequently fewer total events used in the training. $k = 40$ might be the optimal for the limited training time of 24 hours, while a higher k might perform better after more training time. As a higher k would require much more time in the further analysis, the time restriction seems reasonable.

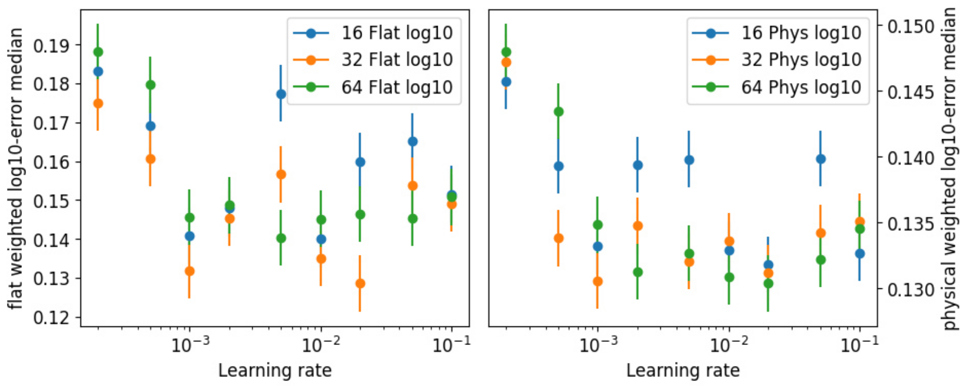
6.4 Learning rate optimisation

In contrast to the hyperparameters discussed in section 6.3, the learning rate η has been optimised for datasets of various interaction type combinations (see section 5.1), as it is typically the most significant hyperparameter with the largest impact on the final network performance. In addition to this generally being the case, it has also been observed by LH with his automatic hyperparameter optimisation for his tau

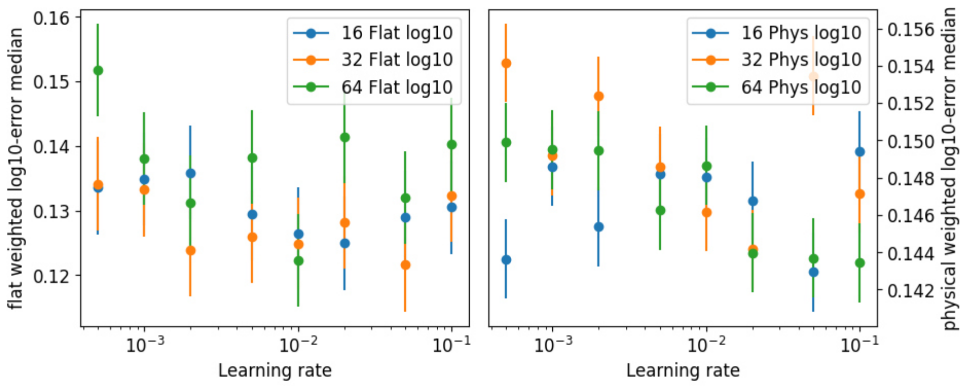
classification task. For this reason, the learning rate is tested for each dataset used in the further analysis individually, in contrast to the previously discussed hyperparameters tested only on ν_e^{CC}, where similar results for the other datasets are expected.
Since the learning rate of neural networks is itself highly dependent on matching batch size, these two parameters have been varied together. The learning has been varied in a range from 5×10^{-4} to 1×10^{-1} around the optimal learning rate found by DG and LH. Each learning rate has been tested with the batch sizes of 16, 32 and 64. The batch sizes are chosen as a power of 2, as this optimises the memory usage of the GPUs.
In total five different datasets have been tested. These include the datasets containing only a single interaction type of either ν_e^{CC}, ν_μ^{CC} or ν_x^{NC}. ν_τ^{CC} events have not been trained on, as these networks aim to test the capabilities of the neural networks when specialising their reconstruction for a single event topology. Since the τ lepton produced in the ν_τ^{CC} interaction, can decay in 3 different ways, which are not tagged during Monte Carlo production and therefore not differentiable, no single event topology can be gained from ν_τ^{CC}. The results for the three single type networks are shown in Figure 6.10.

Additionally, a network has been trained with a dataset containing ν_e^{CC} and ν_x^{NC}, generally called the shower network. The last of the 5 networks was using a dataset containing all interaction types, including ν_τ^{CC}. The results for both multi-type networks are shown in Figure 6.11.

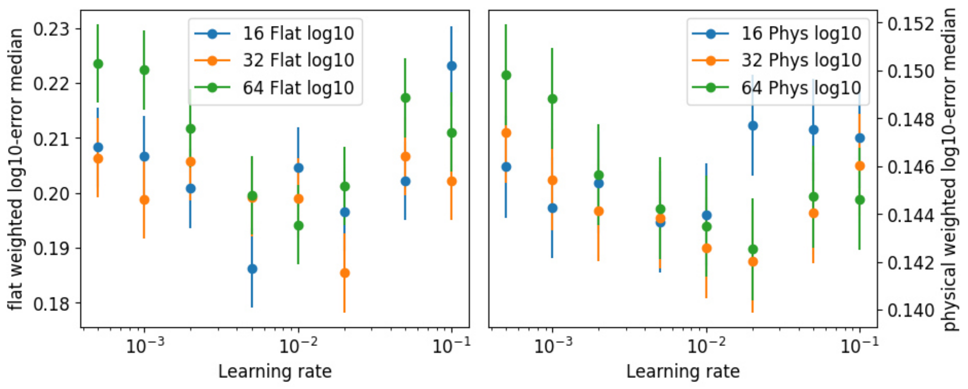
For all five networks a learning rate of 1×10^{-2} seems to be optimal. This is a bit larger than the default value found by DG of 5×10^{-3} and on the higher end of the optimal range found by LH. Due to the time restraint of 24 hours of training, however, lower learning rates are at a disadvantage, since low learning rates adjust the weights in smaller increments and therefore take longer to reach their optimal state. To counter this, a learning rate of one step lower than the best 1×10^{-2} has been chosen to be used in further analysis.
The single-interaction type networks show the best performance for a batch size of 32, while the multi-interaction type networks prefer a batch size of 64. This may be caused by a higher variety of event topologies in the multi-type networks, requiring a higher batch size in order to calculate an accurate average gradient. Additionally, a higher batch size enables training on more events in total, since calculating more events simultaneously in each batch is faster. This is more important for the multi-interaction type events, as they are trained on more total events, with more features to understand.



(a) *Trained with and evaluated on ν_e^{CC}.*

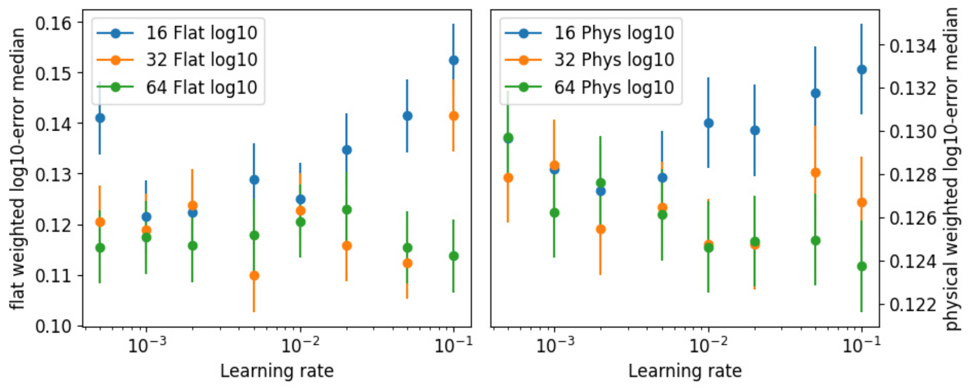


(b) *Trained with and evaluated on ν_x^{NC}.*

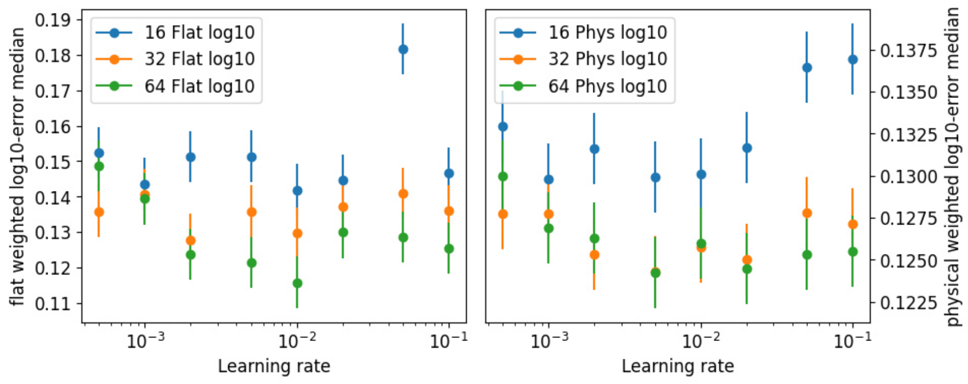


(c) *Trained with and evaluated on ν_μ^{CC}.*

Figure 6.10: *Flat and physical network scores for various learning rates in combination with batch sizes of 16, 32 and 64.*



(a) *Trained with and evaluated on $\nu_e^{CC} + \nu_x^{NC}$. (Shower network)*



(b) *Trained with and evaluated on all interaction types.*

Figure 6.11: *Flat and physical LE network scores for various learning rates in combination with batch sizes of 16, 32 and 64.*

6.5 Summary

In section 6.1 the use of a logarithmic error to evaluate and score the networks in this thesis was explained, as well as why the test events will be weighted flat and physical in the following chapters. Additionally, the statistical error of trained networks has been estimated. Then a hyperparameter optimisation has been performed, comparing to the works of DG and LH. This has been done mainly on networks trained with only ν_e^{CC}, but the learning rate and batch size have been optimised for each of the five different datasets, that will be used in the following analysis. A general agreement with the initial settings taken from DG and the automatic hyperparameter optimisation by LH has been found for all tested hyperparameters.

7 Sample weight testing

This chapter constitutes the main analysis of this thesis. It will first investigate the effects of the different sample weight options presented in section 5.4 and conclude on an overall good setting. This setting will be used to train five networks with the datasets presented at the end of section 5.1, which is discussed in more detail and compare to the previous works of Lukas Hennig and Daniel Guderian. In contrast to chapter 6, the networks in this chapter have been trained for up to four days, until convergence had been reached.

7.1 Sample weight tests

This section will present the effects the 3 sample weight options have on the performance of neural networks. In order to isolate the effects of each option as well as possible, only the tested option will be varied from the standard ${}^{0.5}\text{dataset}^{0.5}_{0.5}$, while the other two are fixed. The ${}^{0.5}\text{dataset}^{0.5}_{0.5}$ is used as the starting ground, as the moderate settings of 0.5 ideally reduce the influence of effects arising for the extreme settings with values of 0 or 1. By displaying an 'x' instead of the value for a sample weight option in the ${}^{0.5}\text{dataset}^{\nu_e}_{\text{x}}$, it is implied that this option is currently being varied in the analysis.
This section is not designed to find the best solution for a specific analysis conducted at the end, but to generally investigated the effects the tested sample weight options have on the network predictions. Therefore the individual effects presented, will not linked by a clear unifying thread, but stand to some degree alone, for the reader to apply to their specific application. The overarching conclusion for each option will be highlighted as **bold**.

7.1.1 Datasets

The results of the variation of sr-percentage in the ${}^{\text{x}}\text{training data}^{0.5}_{0.5}$ can be seen in Figure 7.1. The most striking feature that can be observed is the network behaviour at either 0 or 100% sr-percentage, where most of the time a drastically worse performance, indicated by a high error, can be observed. This is expected, since these sr-percentages set the weight of either the rbr or sr events to 0, reducing the total number of events in the training data considerably. Generally, this effect is more significant at 100% sr events, indicating a better training quality for the rbr-events. The effect is reduced by evaluating on the dataset which the network has been trained on. An exception can be seen at the flat error 0% sr-percentage, which however is still within errors. For this reason, the following discussion will not take these extreme settings into account.

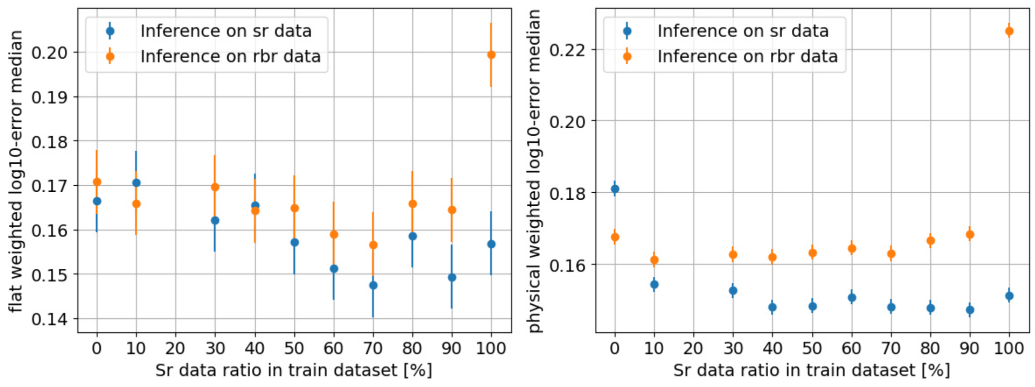


Figure 7.1: *Plot shows weighted LE network score of ^{x}networks$_{0.5}^{0.5}$ on test data separated for rbr and sr events. X axis shows sr ratio in %. Left: Flat weighted. Right: Physically weighted.*

Looking at the flat error on the left, we can see that with increasing sr-percentage the network performance increases, until it reaches its best performance at a sr-percentage of 70%. This makes sense, as the flat error puts more importance on interaction types and energies, with a low physical flux. As the sr dataset boasts more individual events and therefore a higher variety in those regions compared to rbr dataset (see Appendix A), a network with a higher sr-percentage achieves a lower error overall. It has to be explicitly stated, that adding a higher percentage of sr weight only increases the variety at higher energies, while the total weight for higher energies stays the same. While true for both the ${}^{\mathrm{rbr}}$test data and ${}^{\mathrm{sr}}$test data, evaluating on the ${}^{\mathrm{sr}}$test data profits more from a high sr-percentage, reducing its error by 0.2 instead of 0.1 for the ${}^{\mathrm{rbr}}$test data. The difference in error for both networks can be attributed to the lower noise level in the sr data, which especially at high sr percentage reduces the network's capabilities to handle the noisy ${}^{\mathrm{rbr}}$test data.
While the physical score on the right of Figure 7.1 shows the same noise-related error spread in the evaluation on sr and rbr test data as the flat error, it does not change on average. Only evaluating the test datasets separately, reveals a preference of the physical score for a low sr-percentage of 10% for the generally more realistic ${}^{\mathrm{rbr}}$test data. The stronger reaction to a varied sr-percentage of the flat score could be due to the different weighting of scoring metric (flat) and network optimisation, i.e. validation with physically weighted ${}^{\mathrm{rbr}}$validation data$_1^1$. This on average unchanged score for

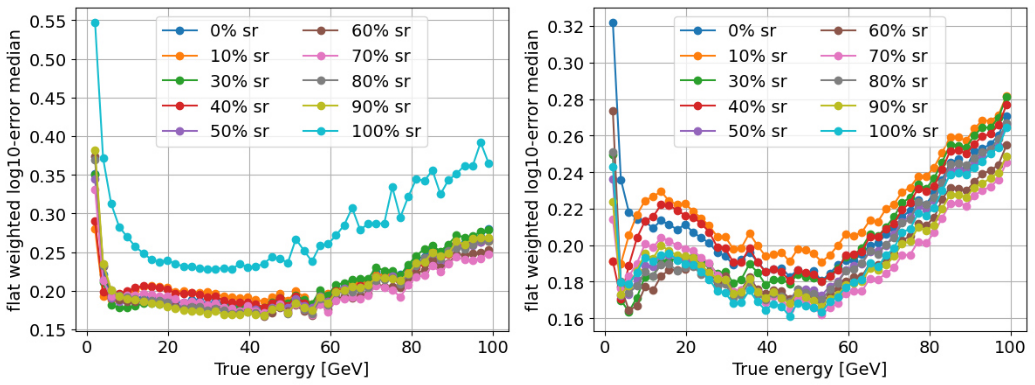


Figure 7.2: *Plots show the energy resolved* ${}^{x}\text{network}_{0.5}^{0.5}$ *flat weighted LE on rbr (left) and sr (right) test data. Legend gives the ratio of sr data in the train dataset.*

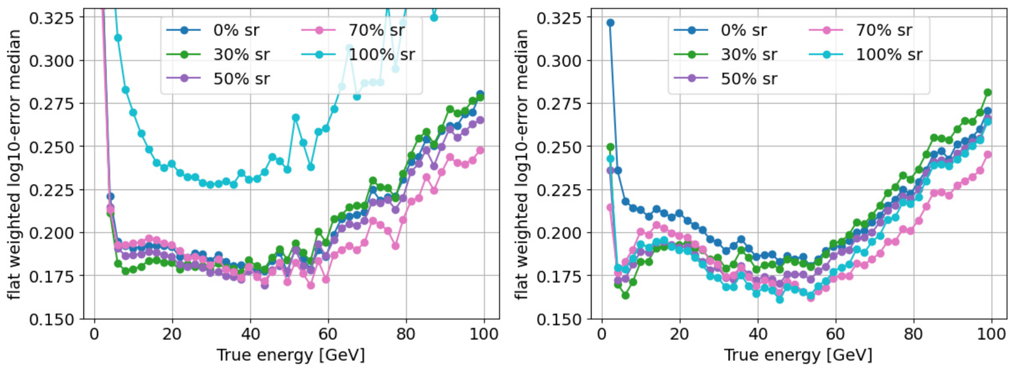


Figure 7.3: *Same plot as Figure 7.2, but with equal y axis and only selected* ${}^{x}\text{networks}_{0.5}^{0.5}$ *for better visibility.*

the physical error also gives a less clear picture of what sr-percentage is optimal. A higher sr-percentage now only seems preferable, when evaluating on the ${}^{\text{sr}}$test data. Evaluating the performance on ${}^{\text{rbr}}$test data instead shows the best performance with a 10% sr-percentage, indicating that the best network for the rbr test data is dependent on the evaluation metric used.

To confirm the hypothesis, that a higher sr-percentage increases performance for high energies, valued more by the flat score, the error evaluated for both datasets and resolved for energy is presented in Figure 7.2 and 7.3. For the ${}^{\text{rbr}}$test data a lower error can be observed for events with energy over 40 GeV, while for ${}^{\text{rbr}}$test data the lower error is already preferable starting at 20 GeV.
From Figure 7.2 we can see, that for both test datasets the best performance is reached around 50 GeV. In addition to this, a range of low error can be found from 3 GeV

to 10 GeV in the sr test data. To a much smaller degree, it can also be observed for the rbr test data. In the sr evaluation, this drop in error can already be observed with a very low sr-percentage, in contrast to the rbr evaluation, where it only occurs for a sr-percentage of less than 80%. This might be caused by some structural difference between sr and rbr events in the training data, which gets exploited by the network.

From the just discussed I conclude, that one should aim for a setting that ensures high statistics and event variety in the regions of interest for the desired analysis, while training the network on enough noisy rbr events, to learn to deal with noise in the real data. For low-energy applications, this is ensured by a low sr ratio and for high-energy applications a higher sr ratio seems more adequate.

7.1.2 Interaction types

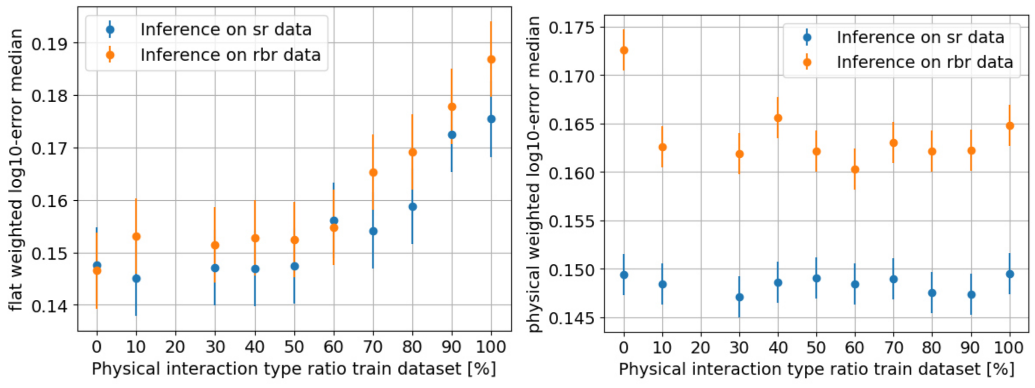


Figure 7.4: *Plot shows weighted LE network score of ${}^{0.5}\text{networks}^{x}_{0.5}$ on test data separated for rbr and sr events. X axis shows the varied interaction type PWS in %.*

The change of ${}^{0.5}\text{network}^{x}_{0.5}$ performance, for flat and physical weighted score, is shown in Figure 7.4. In both cases, the reconstruction of the rbr events is consistently harder than the reconstruction of sr events. Using a flat weighted score results in a steep decrease in performance if the share of physical weights increases over 50%. For the physical weighted score only the ${}^{0.5}\text{network}^{0}_{0.5}$ performs notably worse.
Even though the absolute values of the error distributions are different for sr and rbr test data, their shape is almost identical. For this reason, further analysis will be done only on the rbr test data, which is most representative of real events. In Figure 7.5 the ${}^{0.5}\text{network}^{x}_{0.5}$ performance is shown, for the individual interaction types. The left plot shows a flat weighted score, the right shows a physical weighted score.

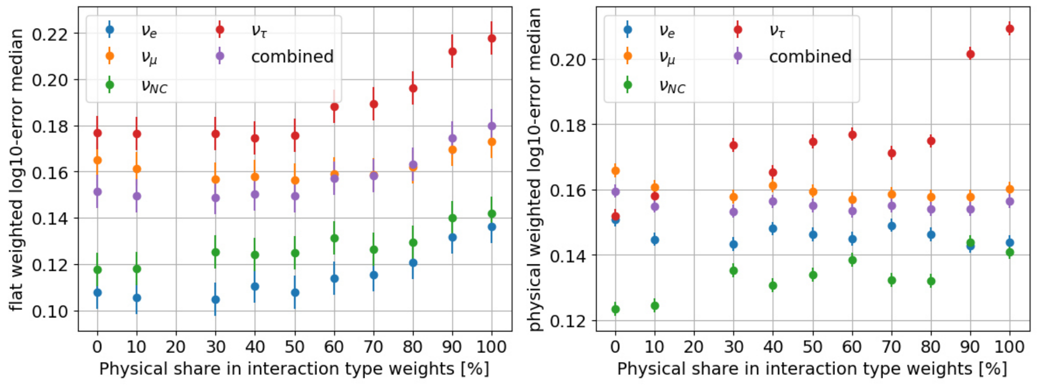


Figure 7.5: *Plots show the energy resolved ${}^{0.5}$network${}^{x}_{0.5}$ weighted LE on ^{rbr}test data${}^{0}_{0}$ (left) and ^{rbr}test data${}^{1}_{1}$ (right). X axis shows the interaction type PWS in %.*

For the flat score the performance of all interaction types decreases with increasing PWS, for the physical score this decrease can only be observed for ν_τ^{CC} and ν_x^{NC}. This is expected, as only these two interaction types lose statistics during training for an increasing PWS. **This shows, that the interaction PWS can be used to adjust the under representation of ν_τ^{CC} and ν_x^{NC}, ensuring even performance over the various interaction types.**

Additionally, this is a very good example, of why a physical weighting in the error can be misleading. To show this, Figure 7.6 plots the energy-resolved performance for the separate interaction types for a low PWS network (left) and a high PWS network (right). We can see, that the low PWS network performs better on nearly every data point, across all interaction types. The only exceptions to this are in the energy range from 2 GeV to 20 GeV especially for ν_μ^{CC}. This is expected, as the 2 GeV to 20 GeV ν_μ^{CC} events alone have about 35% of the total physical weight. Everywhere else the performance is either getting worse or the data points fluctuate too hard to make a confident statement about it. Since this immense concentration on a 'small' group of particles, outweighing everything else, as is done by the physical score (Figure 7.4), can be unexpected and therefore misleading.

Next for varying PWS in training the evolution in energy-resolved performance for combined interaction types is shown in Figure 7.7. Here we can see, that the performance in high energy ranges decreases above 60% PWS, while lower energies (2 GeV to 10 GeV) benefit from a higher PWS. While the spectrum weights stay the same for all interaction types individually for all ${}^{0.5}$networks${}^{x}_{0.5}$, the ν_μ^{CC} neutrino flux falls off faster at higher energies, than the ν_τ^{CC} flux. Resulting in more low energy statistics with a higher interaction type PWS, due to the increase of ν_μ^{CC} statistics overall.
Also with higher PWS the increased performance in the energy range from 2 GeV to

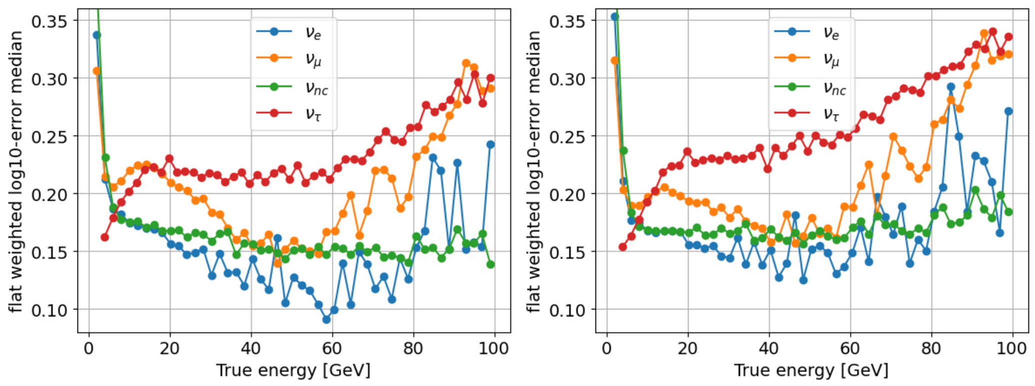


Figure 7.6: *Performance of the* ${}^{0.5}$network$^{0.1}_{0.5}$ *(left) and* ${}^{0.5}$network$^{0.8}_{0.5}$ *(right), resolved by energy and interaction type. Notation as in subsection 2.4.3.*

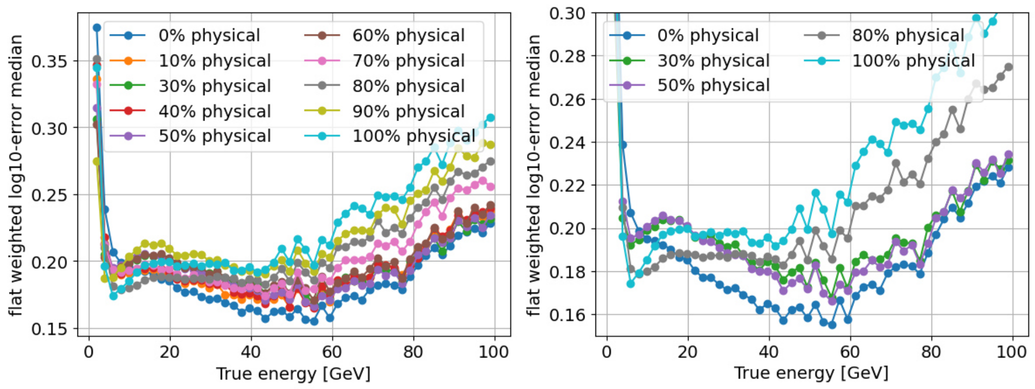


Figure 7.7: *Performance of* ${}^{0.5}$networks$^{x}_{0.5}$*, resolved by energy. Left: All trained networks. Right: Only selected networks for better visibility.*

10 GeV gets more pronounced. In contrast to Figure 7.2, which has varied the dataset ratio statistics, the effect can now also be observed in the ${}^{\text{sr}}$test data$^{\text{x}}_{\text{x}}$. This subsection however only shows performance on rbr test data.

7.1.3 Spectra

In this subsection, the ${}^{0.5}$network$^{\nu_e}_{\text{x}}$ performance is discussed. As for the previous two subsections the analysis starts with the performance for the different datasets shown in Figure 7.8.

We can see, that as before the rbr data is harder to reconstruct than the sr data and the development of the curves is similar for the different datasets. For the flat

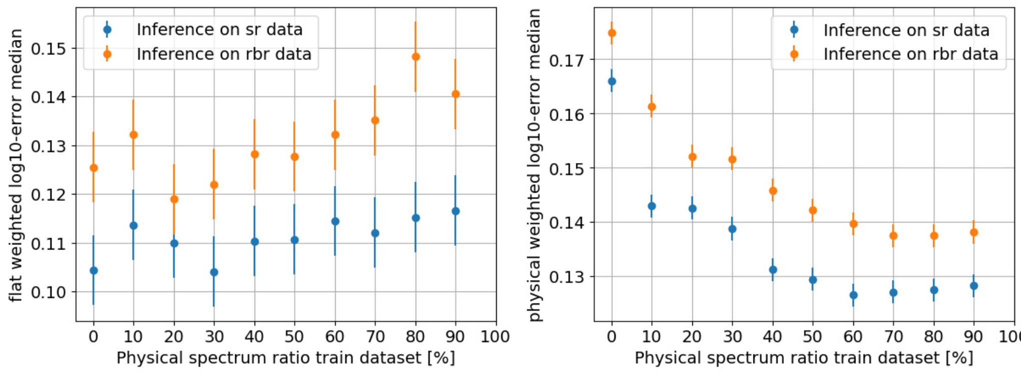


Figure 7.8: *Plots show flat weighted LE network score of* $^{0.5}\text{networks}_x^{\nu_e}$ *on test data separated for rbr and sr events. X axis shows the varied spectrum PWS in %. Left: Flat weighted. Right: Physically weighted.*

weighted error (left) we observe a decrease in performance if trained with high PWS. This finding is consistent with the energy-resolved spectrum in Figure 7.9. The full spectrum is plotted with a logarithmic x-scale since the statistics for ν_e^{CC} with high energies (above 50 GeV) are very low. Still, the higher performance for high PWS is observable for energies of 2 GeV to 20 GeV. Above 20 GeV a lower PWS is preferable.

This also shows for the $^{\text{rbr}}\text{test data}_1^{\nu_e}$, weighing lower energies with more importance. For a PWS above 70% no increase in physically weighted performance can be observed. **I would advise using a PWS of 20 - 70%, depending on the goal of the analysis, as outside of this range no clear improvement for either flat or physical error is noticeable.** The most equal error over the whole energy range can be seen with a PWS of 40%.

7.2 Dataset comparison

The just presented results on the performance of the sample weight options for dataset ratio, interaction type composition and spectrum composition will now be applied in the training of the three single and two multi-interaction type datasets discussed in section 5.1. Without explicit requirements for a concrete analysis, a configuration with a good performance overall is aimed for. From section 7.1 I conclude that such a sample weight configuration can be achieved best with $^{0.7}\text{train data}_{0.4}^{0.5}$. For the general comparison of these five networks, the LE is used, as was discussed in subsection 6.1.1, while for completeness identical plots using the RFE are attached in Appendix C.

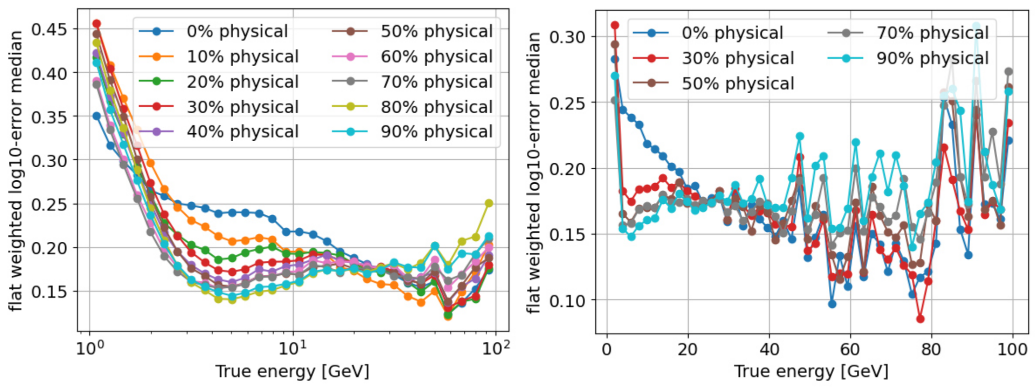


Figure 7.9: *Plots show the energy resolved $^{0.5}$networks$_x^{\nu_e}$ flat weighted LE on $^{\text{rbr}}$test data$_0^{\nu_e}$. Left: All spectrum PWS on logarithmic x-scale. Right: Selected spectrum PWS to show low statistics at high energies.*

The performances of the three single interaction type networks are shown in Figure 7.10. The networks are evaluated on all three self-contained interaction types $^{\text{rbr}}$test datasets$^{\nu_e/\nu_{NC}/\nu_\mu}$. By looking at the flat and physical network scores in each column, it can be seen that every interaction type is best reconstructed by the network trained on events of the same interaction type shown on the diagonal. This indicates that the graphs of every interaction type do hold different information, that can be recognised by the networks.
Additionally one can observe a lower error for $^{0.7}$shower networks$_{0.4}^{\nu_e+\nu_{NC}}$ compared to the $^{0.7}$track network$_{0.4}^{\nu_\mu}$. This shows that track events are inherently much harder to reconstruct than both other types. This is expected as in contrast to shower events, track events generally leave the detector volume before depositing all their energy. It can also be seen when only looking at the scores of the track network in the bottom row of plots. While not trained on shower events, the track network gives a lower error of the reconstruction of shower events than the trained on track events. In order for this to be possible, the information decoded in the event graphs has to be very similar, which allows the network to translate its learnings from track events to the easier-to-reconstruct shower events.

Looking at the performance of the two multi-type networks shown in Figure 7.11 we can see, that except for the shower network evaluated on ν_x^{NC}, both networks are performing equivalently or even better than the respective specialised single-type network. This is surprising, as I would have expected a slight, but definitely worse performance compared to the specialised networks. This finding should be taken with caution, as it stands in contrast to the general experience of KM3NeT works, which agrees with my expectations. The reasoning for my expectation goes as follows.
Since the different event topologies do to some degree show different patterns in the

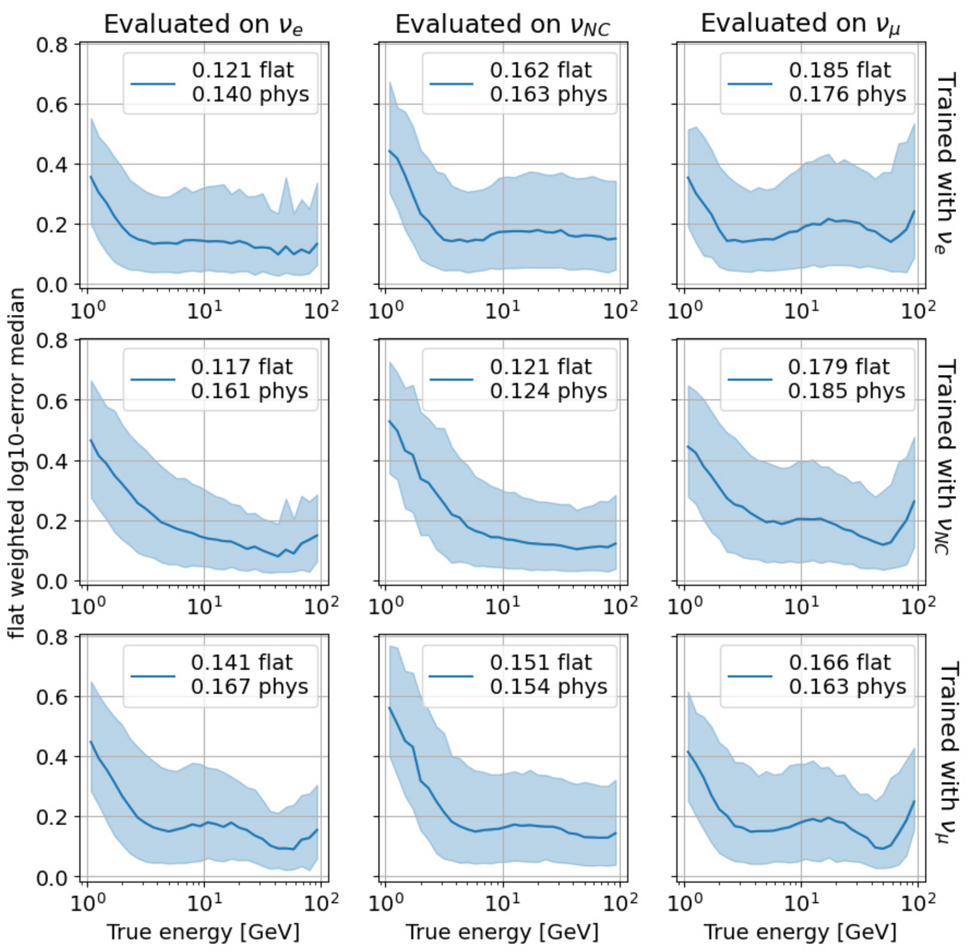


Figure 7.10: *Single interaction networks trained and evaluated on ν_e^{CC},ν_x^{NC} and ν_μ. Plots show the median, 16 and 84 percentile of the flat weighted LE resolved over true energy. Legend shows flat and physical LE score. Notation follows subsection 2.4.3.*

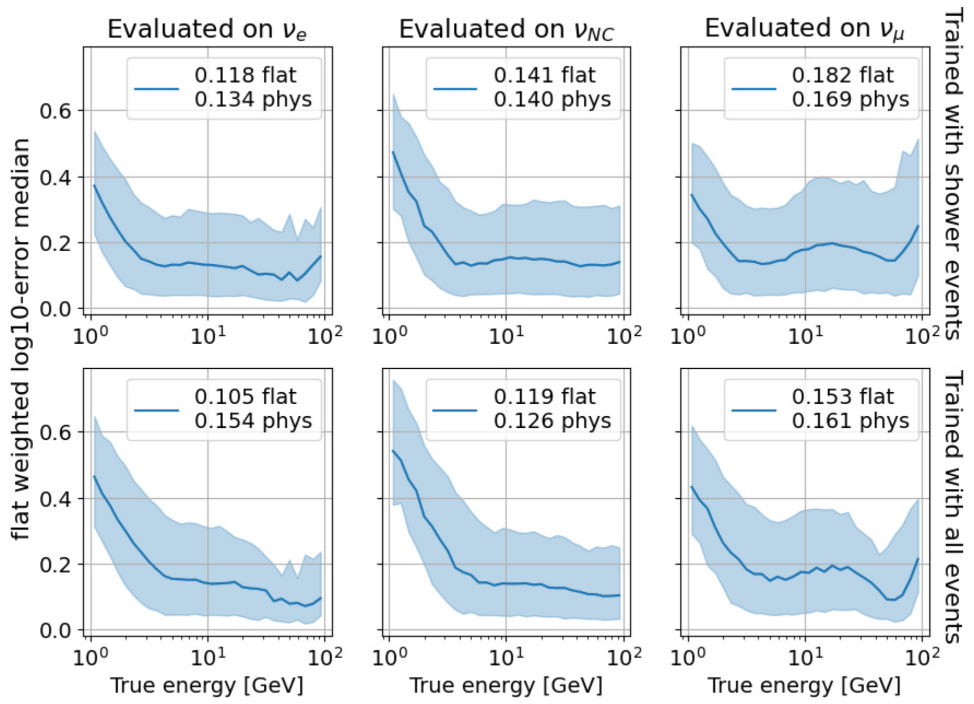


Figure 7.11: *Multi interaction networks trained on shower and all interaction types. Evaluated on ν_e^{CC},ν_x^{NC} and ν_μ. Plots show the median, 16 and 84 percentile of the flat weighted LE resolved over true energy. Legend shows flat and physical LE score. Notation follows subsection 2.4.3.*

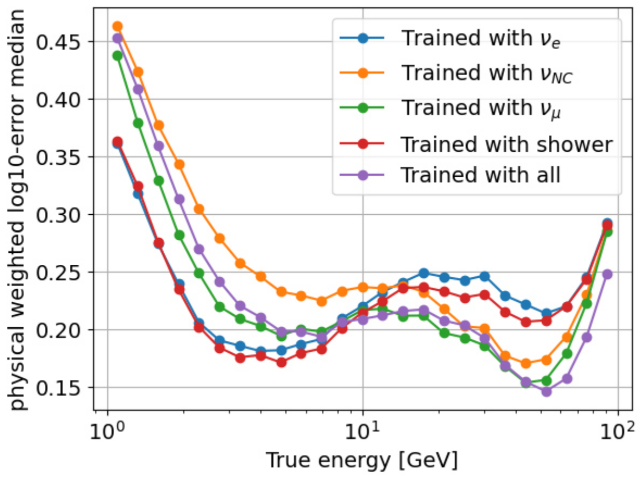


Figure 7.12: *Plot shows the energy resolved physical LE score for all 5 networks discussed. The legend shows which interaction types were used in the networks training. Notation follows subsection 2.4.3.*

graphs, that need to be analysed. When trained on multiple event topologies, the network needs to implicitly decide, which topology the event has and reconstruct with the according pattern. This reduces the network performance as it cannot distinguish perfectly between the different interaction types as was done for the single-type networks. However, by combining the interaction types, other unknown factors must be at play, which I cannot explain. The question would be if these other factors are somewhat based on the physical features of the event graphs, a higher amount of different graphs or positive effects of the combined events statistics.

Furthermore, we can observe, that especially in the evaluation on ${}^{\nu_\mu}$test data$_,$ the low error at 2 GeV to 10 GeV is related to the amount of ν_e^{CC} in the training data. Figure 7.12 shows the performance of all five networks evaluated on ${}^{\nu_\mu}$test data$_.$ The strongest dip can be observed for pure ν_e^{CC} and combined shower events, both boasting a high amount of ν_e^{CC}. The last dip is found with ν_x^{NC}. Unfortunately, I cannot explain this behaviour, even though it is probably caused by an important systematic, judging from its severeness.

7.3 Standard networks

Next, other reconstruction methods are compared to the weighted ${}^{0.7}$networks$^{0.5}_{0.4}$. The reconstruction methods in question mostly distinguish between different event topologies, as long as a well-performing classification can be established. This classification

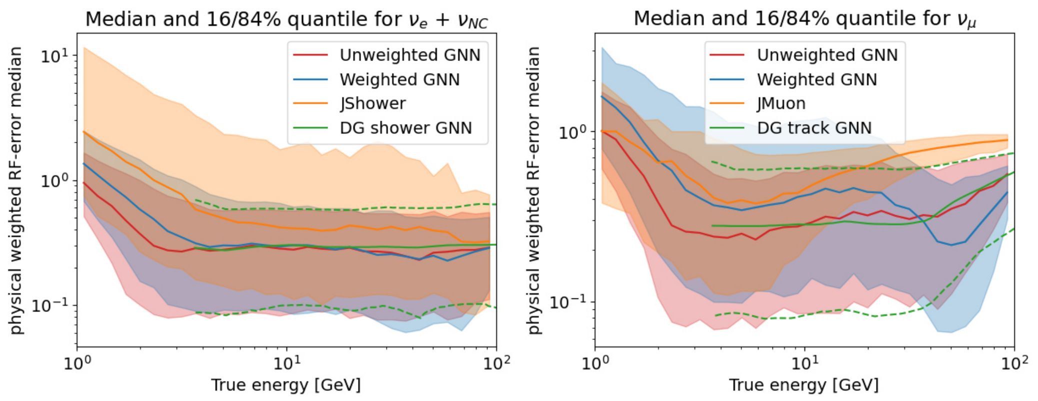


Figure 7.13: *Comparison of the weighted GNN, unweighted GNN, standard reconstruction and DG GNN for shower (left) and track (right) network. Plots show the median, 16 and 84 percentile of the flat weighted RFE resolved over true energy. Notation follows subsection 2.4.3.*

can be done between track and shower events, but not between electromagnetic and hadronic showers. For this reason, the track and shower network will be compared to the other reconstruction algorithms.

7.3.1 Comparison with other reconstructions

First a comparison of the median, 16th and 84th percentile of the LE for the two networks evaluated on $^{\mathrm{rbr}}$test data^0 is shown in Figure 7.13. It compares the performance of the standard reconstruction algorithms, JShower (shower events) and JMuon (track events), with the original energy reconstruction by DG, the unweighted network introduced in chapter 6 with the sample weight optimised networks. The comparison is done using the neural networks disadvantageous RFE since the data for the energy reconstruction by DG has been taken from plots in his thesis and could not be reproduced from data in order to calculate the LE.

For the shower network, a clear outperformance of the standard algorithm can be observed for all neural network-based reconstructions. Compared to DG, both new networks performed slightly better above 20 GeV. All neural networks show an equal error distribution down to 3 GeV. The results from DG unfortunately are not available below this threshold. The unweighted network even preserves this behaviour down to 2 GeV. Below this energy threshold, all reconstruction algorithms show an increasingly bad energy reconstruction. The good performance down to a lower energy of the unweighted network compared to the weighted network, is probably due to the applied weighting. As discussed in subsection 6.1.2, the anticipated neutrino flux

actually tapers off below 3 GeV. In the distribution of generated events, however, this is not apparent, thereby inducing more incentive for the network to guess very low energies. This is balanced by better performance of the weighted network in the range of 50-80 GeV, even though this is not as significant.
For the track network, the picture is not as clear. Here the network of DG and the unweighted network show agreement in their performance. This makes sense, since they are trained on essentially the same dataset. Both of them outperform the standard reconstruction considerably as was the case for the shower network. The weighted network on the other hand is performing worse, than even the standard reconstruction below energies of 20 GeV, with a slight performance increase down to 3 GeV. On the other hand for energies above 40 GeV the weighted network is performing dramatically better than even the other neural networks. Even though this uneven performance of the network is most likely not optimal, it shows the power weighted training can have over the final network performance. In order to achieve a more even response, could probably gained with a ${}^{0.7}\text{network}^{\nu_\mu}_{>0.4}$, using a higher PWS for the energy spectrum. This higher PWS can then counteract the smaller amount of low energetic neutrinos in the flux of ν_μ^{CC} compared to ν_e^{CC} (see Figure 6.3). At energies below 2-3 GeV the same decrease in performance as for the shower network can be observed.
The bad performance of the weighted ${}^{0.7}\text{network}^{\nu_\mu}_{0.4}$ compared to the standard reconstruction changes, when no longer using the RFE, but instead the LE. Looking at Figure 7.14, we can see, that for the LE the weighted network now outperforms JMuon for every energy. The performance relative to the other neural network does not change considerably. This shows the importance of evaluating an algorithm on the actual target application, instead of a proxy metric, where drastically different results can occur.

7.3.2 2D energy histogram

Further explanations of the findings from subsection 7.3.1 can be given, when analysing a 2D histogram of the true and predicted values of the test data, shown in Figure 7.15. For both networks tails diverging from the ideal ratio of $true/pred = 1$ at the ends of the training energies can be observed. This behaviour can be observed in all networks trained on a confined reconstruction quantity. In the reconstruction of an event, a network analyses the input and calculates its prediction. However, as networks cannot reconstruct with perfect precision, the network to some degree also bases its prediction on the training statistics. This guessing shows in the spread of the reconstructed values. At the edges of the possible values of the reconstructed quantity, however, guessing is only possible in one direction, which shifts the spread towards the centre of the distribution and focuses it into this tail.
At low energies, this occurs equally strongly for both networks, while at high energies

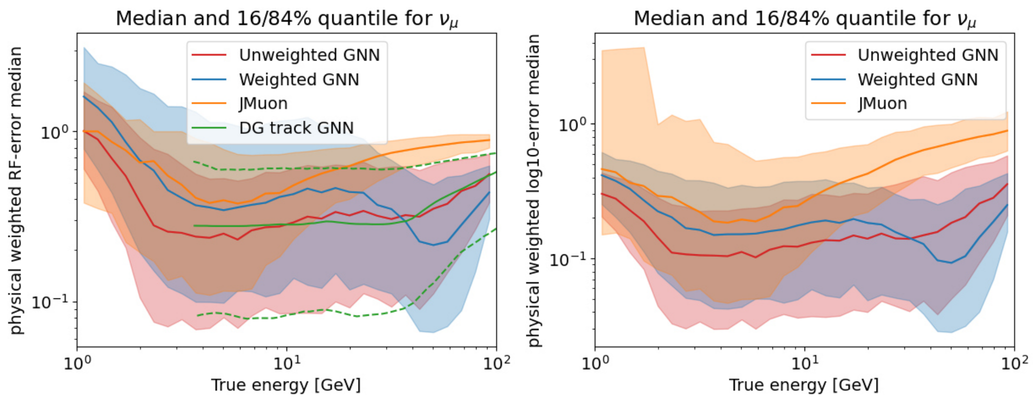


Figure 7.14: *Comparison of the physical weighted RFE (left) and LE (right) of the track network for weighted GNN, unweighted GNN and standard reconstruction JMuon. Plots show the median, 16 and 84 percentile of the flat weighted RFE resolved over true energy. Notation follows subsection 2.4.3.*

the deviation of the shower network is not as pronounced as for the track network. This makes sense, as the reconstruction of low energy events is due to their lower average hit count harder and therefore promotes more statistical guessing of the network. Additionally, one can see the track network reconstructing events in the range of 8-40 GeV with an offset to the true energy instead of a central scattering around the target value. Relating these things to Figure 7.13, we can explain the steep decrease

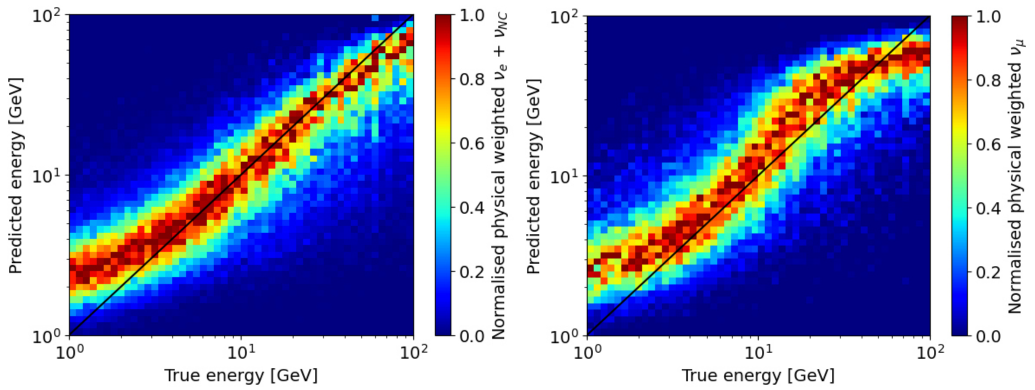


Figure 7.15: *2D histograms of predicted over true energy of shower (left) and track (right) network. Black line gives reference for ideal reconstruction. Bins are normalised within column. Notation follows subsection 2.4.3.*

in performance by the low energy tails. In order to combat this, the incentive of the network to guess also towards its energy edges, instead of the centre has to be

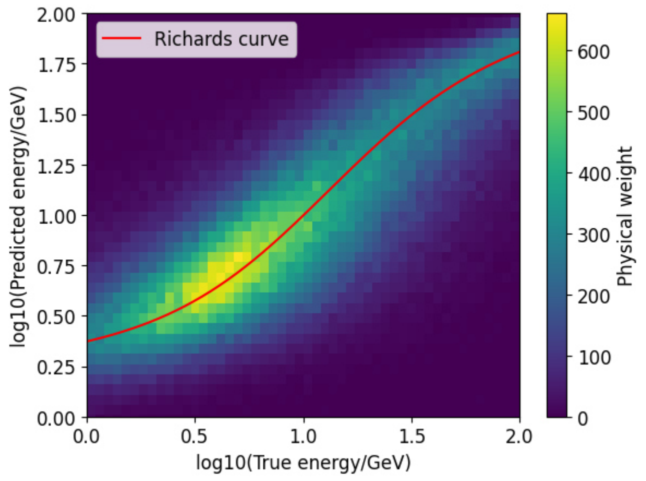


Figure 7.16: *Physical weighted 2D histogram of track network for log10 of predicted over true energy. Red curve shows manually optimised modified Richards curve (Equation 7.1,Table 7.1) to trace the reconstruction center.*

increased. This could be done, by training the network with more weight on the lowest energies, than is expected in the flux. Ironically this is being done, when training on unweighted data, explaining the good performance down to 2 instead of 3 GeV.
The bad performance of the track network between 3 and 40 GeV is explained by the offset in reconstructed energy. This seems to be necessary, to enable the very accurate reconstruction at 50 GeV, which coincides with the location, where the network crosses the ideal reconstruction line.

7.3.3 Energy correction

In order to combat the effect of the tails and offset, described in subsection 7.3.2, an energy correction has been tested. The idea for this correction is to correct the network prediction y_{pred} by the difference of this energy to where it is on average belonging $\widehat{y_{true}}$. This difference is given by a function δ tracing the output ratio of predicted to true energy. Figure 7.16 shows the 2D histogram of the true and predicted energies of the shower network, as well as the difference function δ relating the predicted energy y_{pred} with its potential optimal position $\widehat{y_{true}}$. In order to receive a smooth output it is important that the relation of these two values is given by a single function and cannot be calculated in bins or with linear interpolations of bins. The difference function is described by a modified logistic curve, called Richards curve [81], with an added slope, described in Equation 7.1 and the parameters given in Table 7.1.

Variable	A	K	C	Q	B	ν	m
Value	0.32	1.75	1	13	2.5	0.8	0.1

Table 7.1: *Parameters of modified Richards curve from Equation 7.1 in Figure 7.16.*

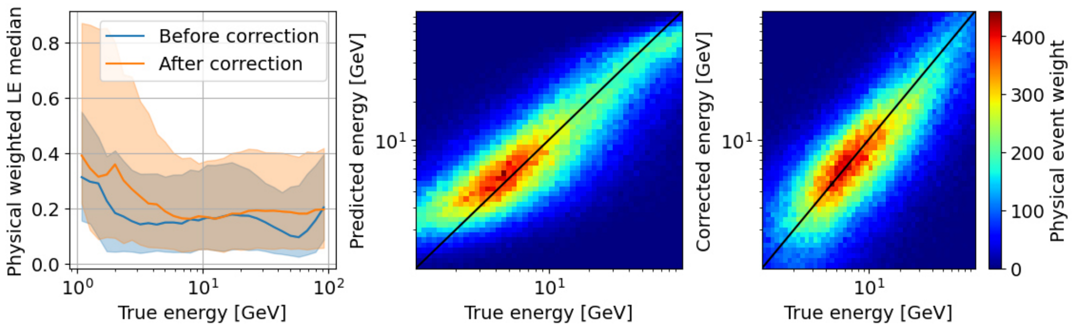


Figure 7.17: *Comparison of track network before and after energy correction. Left: Physical LE median, 16 and 84 percentile resolved over true energy. Middle: Physical weighted 2D histogram of predicted over true energy. Right: Physical weighted 2D histogram of corrected over true energy.*

$$\delta(\widehat{y_{true}}) = A + \frac{K - A}{(C + Q \cdot \exp(-B \cdot \widehat{y_{true}}))^{1/\nu}} + m \cdot \widehat{y_{true}} \tag{7.1}$$

This function was chosen to approximate the difference since the basic logistic curve already resembles the s-shape of the network output. The modified logistic curve used here allows for change of the lower and upper asymptote and creates the possibility for one of the bellies to become larger, by removing the pointsymmetric nature of the logistic curve. At last, the added slope allows the function to tilt. The parameters have not been fitted but varied manually until an adequate tracing had been achieved. A comparison of before and after the correction is shown in Figure 7.17. It can be seen in the 2D histograms that the tails have been removed and the corrected energies are now spread symmetric around the ideal prediction, showing the intended effect of the correction.

However looking at the median and quantiles of the LE, we can see, that the performance is not increasing on the edges, but instead decreasing in the centre of the distribution. The latter is happening, since by adjusting all values with a certain predicted energy, which mostly means pulling the distribution apart, the more or less badly reconstructed events with medium true energy values are being moved away from the ideal reconstruction instead of towards it, as is the case for the tails. The lack of intended improvement at the tails is understandable when seeing, that by pulling the centre of the tails on the ideal reconstruction line, the spread of the events becomes

so large, that it is cancelling out the benefits of the center adjustment.
Due to these results, it was decided to not use an energy correction. However, one could definitely investigate, how the change from an offset-dominated error to a spread-dominated error influences actual analysis. Furthermore, no effects concerning cuts on predicted uncertainty or the low quality of the events have been taken into account. Investigating these things, however, is out of the scope of this thesis.

7.3.4 Data/Monte Carlo ratio

At last, the Data/Monte Carlo ratio (DMCR) for both networks is investigated, where real data recorded by KM3NeT/ORCA6 is compared to physically weighted MC simulations. The MC data imitate the real data by including neutrinos and atmospheric neutrinos, but differ by not including simulated pure noise events. In order to ensure equivalence in the distributions, the noise events in the real data have to be vetoed. This is done by applying the cuts discussed in section 3.3.

- Number of triggered hits: nTriggerHits ≥ 15
- Reconstructed as upgoing: JGandalf direction > 0
- Signal-noise classifier: JGandalf likelihood ≥ 40

By applying these cuts, also most down-going atmospheric muons are removed. Additionally to event targeting cuts, several runs were removed completely from the dataset. The cuts and removed runs are the same as in [82].

In both plots shown in Figure 7.18, the green curve shows the physically weighted true MC flux. The stark increase at about 70 GeV shows the beginning of the atmospheric muon flux, since, as explained in section 3.3, they need to have at least this much energy, in order to penetrate the water above the detector and not lose the energy before reaching the detector. The energy reconstruction of the MC simulations and real data are plotted for the GNN and standard reconstruction algorithm. By dividing the histogram bin values one can calculate the DMCR, which gives an indication of the quality of the MC simulations, i.e. how good it replicates the real physics.
One can immediately see, that reconstructed values of the standard reconstruction are distributed much wider, than for the GNN reconstruction. This is mainly due to the fact, that the network is only trained on events from 1 GeV to 100 GeV, while the standard algorithm is based on our understanding of the underlying physical processes, which do not limit the valid energy ranges. The network has therefore learned that returning a value outside the training data values is always wrong, thereby hindering its ability to extrapolate its learnings to higher energies. This is shown also by the much higher number of events reconstructed in the 3 GeV to 30 GeV region, both data and MC events, which have been 'pulled down' from the bulk of events up to 500 TeV. However, some events are still reconstructed at energies over 100 GeV, while this

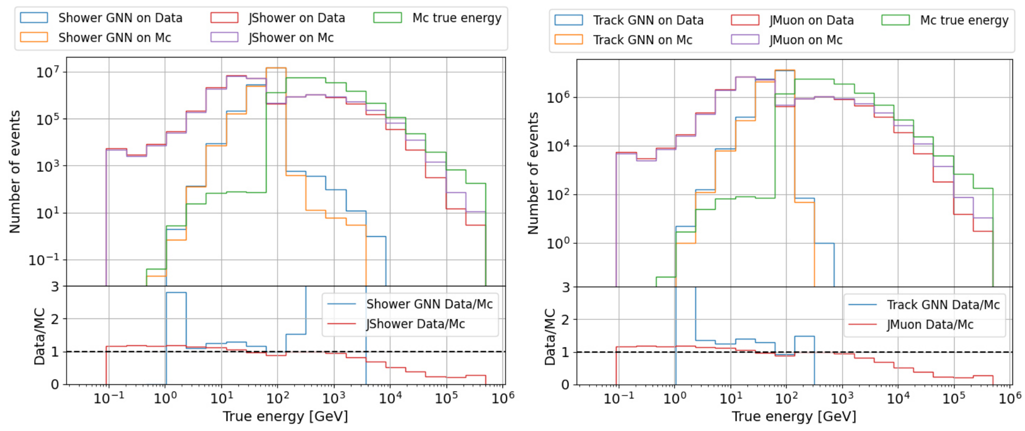


Figure 7.18: *DMCR for shower (left) and track (right) network. Upper axis show the MC true energy (green) GNN reconstruction on data (blue) and MC (orange), as well as standard reconstruction on data (red) and MC (purple). Lower axis shows ratio of data reconstruction to MC reconstruction for GNN (blue) and standard reconstruction (red). Standard reconstruction for shower is JShower and track is JMuon.*

extrapolation is more prominent for the shower network than the track network. This is in agreement with the 2D histograms discussed in subsection 7.3.2, where the upper limit tail is much more flat for the track network. It might be useful to investigate the extrapolation capabilities of networks when an energy correction as discussed in subsection 7.3.3 is used.

Below 3 GeV the DMCR for the GNN shows a stark disagreement between simulation and real data. This may be the case since at low energies the events are typically of bad quality and only have a few hits. This enables the neural network to find patterns, which are not caused by the underlying physics behind the simulation, but instead by the low complexity of the graphs. As the standard reconstructions are written based on the expected physics, they are much less prone to finding patterns that are not actually there.

Even when only looking at the range of 3 GeV to 100 GeV, where the GNN performs best, the standard algorithms show better agreement in the reconstruction of real data and simulations. While the GNN has an error of about 20% in this energy range, the standard reconstruction has a DMCR error of only 10% up until 1 TeV. Above 1 TeV the standard algorithm agreement declines. This increasing deviation confirms that some problems exist within the KM3NeT simulations. These problems are known and under investigation. The higher error of the GNN overall might be caused by the low and high energy event problems discussed just now, which are consequently inferring their DMCR disagreement to the central energy region in between, increasing its DMCR error.

7.4 Conclusion

By testing the three options of sample weights, a configuration for good overall performance of shower events was found and consequently used to compare to the standard reconstruction algorithms of KM3NeT and the GNN energy reconstruction by DG. With these investigations, it was shown, that the use of sample weights in the training of neural networks, holds power over their performance. While no configuration of sample weights was found, that improved the network performance over the entirety of the trained energy range, sample weights allow the user to influence at what energies the network will achieve a better or worse performance. This allows the network to be optimised for energies more relevant to the concrete analysis one wants to run. However, the benefits of optimising the network in this way can only be tested directly with the given analysis. Doing this, however, is computationally expensive and very impractical if one would want to implement it as the loss function during training.

8 Summary and Outlook

This thesis has tested the effects and possible benefits of applying sample weights during the training of GNNs. For this three different options to calculate the weights were evaluated and the best found configuration has been compared to likelihood-based reconstruction algorithms, as well as the previous GNN energy reconstruction by Daniel Guderian.

First, the relevant theory of neutrino physics has been presented in chapter 2, including the phenomenon of neutrino oscillations, the production of atmospheric neutrinos and the different typologies of neutrinos induced particle cascades produced dependent on their flavour and mediating gauge boson. Next, in chapter 3, the detection of neutrinos by the water Cherenkov detector KM3NeT/ORCA has been explained.

Furthermore, in chapter 4, the principle of neural networks has been explained. It starts with the basic building blocks of neural networks and the standard training procedure used in supervised learning. And leads up to the working principle of Graph neural networks (GNNs) and in particular the ParticleNeT architecture, which is used for the energy reconstruction tested in this thesis.

Next, the Monte Carlo simulation data used for training and evaluation of these networks was explained in chapter 5. This chapter also extended the basic training of neural networks to weighted training, which enables training on different statistics, than naturally given by the data. The 3 options for calculating the sample weights, for dataset, interaction type and spectrum weights have been explained. In the end, potential problems with the current implementation of weighted training, like varying batch size, have been highlighted and potential alternative implementations for weighted training were discussed.

In order to ensure the validity of results in the greater context of the KM3NeT deep learning efforts a manual hyperparameter optimisation has been performed in chapter 6. It showed good agreement with the first GNN energy reconstruction in KM3NeT by Daniel Guderian. However, it was decided to use a different scoring metric than before in order to better match to optimisation of the trained neural networks.

The main analysis, testing the effects of the 3 sample weight options and comparison of a chosen configuration to other reconstruction algorithms used in KM3NeT is presented in chapter 7. It was found that including more data samples is beneficial to the network performance, as long as the noise level in the data was kept similar

to real detector conditions. The best setting was found to be around 70% single-run data. Additionally, it was shown, that specifying weights for interaction type ratio and spectra, an amplification of regions with low statistical relevance is beneficial. Beyond a certain level of presence in the data, however, only a shift in the network optimisation could be observed. This lead to a significant decrease in performance in other now less weighted regions. In order to produce an overall well performing network, a configuration of 70% single run data, 50% physical interaction types and 40% physical spectrum weights was found optimal. Further testing revealed this configuration resulted in an overall good performance only for shower events.
By comparing the sample weight optimised network to the likelihood-based reconstruction algorithms of JShower and JMuon and the GNN energy reconstruction by Daniel Guderian it was found, that the use of sample weights resulted in a similar reconstruction error distribution as the unweighted GNNs and definitely better performance, than the likelihood-based approaches. Additionally, an energy connection of the neural network has been tried, in order to correct for the tails produced by the neural networks, which however did not improve the overall network performance. As this correction has only been tried out very briefly, it could still be useful, to test if quality cuts could extend the range of accurate network predictions into lower energies.

Bibliography

[1] Cush. *Wikipedia Standard Model of particle physics*. https://en.wikipedia.org/w/index.php?lang=en&title=File%3AStandard_Model_of_Elementary_Particles.svg. Accessed: 2024-02-28.

[2] Laurie M. Brown. 'The idea of the neutrino'. In: *Physics Today* 31.9 (Sept. 1978), pp. 23–28. ISSN: 0031-9228. DOI: 10.1063/1.2995181. eprint: https://pubs.aip.org/physicstoday/article-pdf/31/9/23/8285438/23_1_online.pdf. URL: https://doi.org/10.1063/1.2995181.

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

[4] Frederick Reines and Clyde L. Cowan. 'The neutrino'. In: *Nature* 178 (1956), pp. 446–449. DOI: 10.1038/178446a0.

[5] G. Danby et al. 'Observation of High-Energy Neutrino Reactions and the Existence of Two Kinds of Neutrinos'. In: *Phys. Rev. Lett.* 9 (1 July 1962), pp. 36–44. DOI: 10.1103/PhysRevLett.9.36. URL: https://link.aps.org/doi/10.1103/PhysRevLett.9.36.

[6] *Nobelprize Physics 1988*. https://www.nobelprize.org/prizes/physics/1988/summary/. Accessed: 2024-02-11.

[7] *Nobelprize Physics 1995*. https://www.nobelprize.org/prizes/physics/1995/summary/. Accessed: 2024-02-11.

[8] K. Kodama et al. 'Observation of tau neutrino interactions'. In: *Phys. Lett. B* 504 (2001), pp. 218–224. DOI: 10.1016/S0370-2693(01)00307-0. arXiv: hep-ex/0012035.

[9] ALEPH Collaboration et al. 'Precision electroweak measurements on the Z resonance'. In: *physrep* 427.5-6 (May 2006), pp. 257–454. DOI: 10.1016/j.physrep.2005.12.006.

[10] Y. Fukuda et al. 'Evidence for Oscillation of Atmospheric Neutrinos'. In: *Physical Review Letters* 81.8 (Aug. 1998), pp. 1562–1567. ISSN: 1079-7114. DOI: 10.1103/physrevlett.81.1562. URL: http://dx.doi.org/10.1103/PhysRevLett.81.1562.

[11] Q. R. Ahmad et al. 'Direct Evidence for Neutrino Flavor Transformation from Neutral-Current Interactions in the Sudbury Neutrino Observatory'. In: *Phys.*

Rev. Lett. 89 (1 June 2002), p. 011301. DOI: 10.1103/PhysRevLett.89.011301. URL: https://link.aps.org/doi/10.1103/PhysRevLett.89.011301.

[12] *Nobelprize Physics 2015*. https://www.nobelprize.org/prizes/physics/2015/summary/. Accessed: 2024-02-11.

[13] B. Pontecorvo. 'Neutrino Experiments and the Problem of Conservation of Leptonic Charge'. In: *Zh. Eksp. Teor. Fiz.* 53 (1967), pp. 1717–1725.

[14] Ziro Maki, Masami Nakagawa and Shoichi Sakata. 'Remarks on the unified model of elementary particles'. In: *Prog. Theor. Phys.* 28 (1962), pp. 870–880. DOI: 10.1143/PTP.28.870.

[15] Ivan Esteban et al. 'The fate of hints: updated global analysis of three-flavor neutrino oscillations'. In: *Journal of High Energy Physics* 2020.9 (Sept. 2020). ISSN: 1029-8479. DOI: 10.1007/jhep09(2020)178. URL: http://dx.doi.org/10.1007/JHEP09(2020)178.

[16] *Neutrino mass ordering*. https://www.mdpi.com/psf/psf-08-00007/article_deploy/html/images/psf-08-00007-g001.png. Accessed: 2024-02-28.

[17] Ara Ioannisian and Stefan Pokorski. 'Three neutrino oscillations in matter'. In: *Physics Letters B* 782 (2018), pp. 641–645. ISSN: 0370-2693. DOI: https://doi.org/10.1016/j.physletb.2018.06.001. URL: https://www.sciencedirect.com/science/article/pii/S0370269318304428.

[18] S Adrián-Martínez et al. 'Letter of intent for KM3NeT 2.0'. In: *Journal of Physics G: Nuclear and Particle Physics* 43.8 (June 2016), p. 084001. DOI: 10.1088/0954-3899/43/8/084001. URL: https://dx.doi.org/10.1088/0954-3899/43/8/084001.

[19] *Fluxes of different terrestrial and cosmic neutrino sources*. https://www.researchgate.net/figure/Fluxes-of-different-terrestrial-and-cosmic-neutrino-sources-Figure-taken-from-11_fig1_331048316. Accessed: 2024-02-28.

[20] Takaaki Kajita. 'ATMOSPHERIC NEUTRINOS AND DISCOVERY OF NEUTRINO OSCILLATIONS'. In: *Proc. Japan Acad. B* 86 (2010), pp. 303–321. DOI: 10.2183/pjab.86.303.

[21] M. Honda et al. 'Reduction of the uncertainty in the atmospheric neutrino flux prediction below 1 GeV using accurately measured atmospheric muon flux'. In: *Phys. Rev. D* 100 (12 Dec. 2019), p. 123022. DOI: 10.1103/PhysRevD.100.123022. URL: https://link.aps.org/doi/10.1103/PhysRevD.100.123022.

[22] G. D. Barr et al. 'Uncertainties in atmospheric neutrino fluxes'. In: *Physical Review D* 74.9 (Nov. 2006). ISSN: 1550-2368. DOI: 10.1103/physrevd.74.094009. URL: http://dx.doi.org/10.1103/PhysRevD.74.094009.

[23] G. Battistoni et al. 'The FLUKA atmospheric neutrino flux calculation'. In: *Astroparticle Physics* 19.2 (May 2003), pp. 269–290. ISSN: 0927-6505. DOI: 10.

1016/s0927-6505(02)00246-3. URL: http://dx.doi.org/10.1016/S0927-6505(02)00246-3.

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

[25] J.A. Aguilar et al. 'Design and sensitivity of the Radio Neutrino Observatory in Greenland (RNO-G)'. In: *Journal of Instrumentation* 16.03 (Mar. 2021), P03025. DOI: 10.1088/1748-0221/16/03/P03025. URL: https://dx.doi.org/10.1088/1748-0221/16/03/P03025.

[26] P. A. Cherenkov. 'Visible luminescence of pure liquids under the influence of γ-radiation'. In: *Dokl. Akad. Nauk SSSR* 2.8 (1934), pp. 451–454. DOI: 10.3367/UFNr.0093.196710n.0385.

[27] Mathieu de Naurois and Daniel Mazin. 'Ground-based detectors in very-high-energy gamma-ray astronomy'. In: *Comptes Rendus. Physique* 16.6–7 (Aug. 2015), pp. 610–627. ISSN: 1878-1535. DOI: 10.1016/j.crhy.2015.08.011. URL: http://dx.doi.org/10.1016/j.crhy.2015.08.011.

[28] Geiselbrecht Nicole. 'Event Classification and Energy Reconstruction for ANTARES using Convolutional Neural Networks'. MA thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2021.

[29] Jannik Hofestädt and on behalf of theKM3NeT Collaboration. 'Measuring the neutrino mass hierarchy with KM3NeT/ORCA'. In: *Journal of Physics: Conference Series* 1342.1 (Jan. 2020), p. 012028. DOI: 10.1088/1742-6596/1342/1/012028. URL: https://dx.doi.org/10.1088/1742-6596/1342/1/012028.

[30] Particle Data Group et al. 'Review of Particle Physics'. In: *Progress of Theoretical and Experimental Physics* 2020.8 (Aug. 2020), p. 083C01. ISSN: 2050-3911. DOI: 10.1093/ptep/ptaa104. eprint: https://academic.oup.com/ptep/article-pdf/2020/8/083C01/34673722/ptaa104.pdf. URL: https://doi.org/10.1093/ptep/ptaa104.

[31] *Cherenkov threshold values.* https://www.sheffield.ac.uk/physics/research/particle/neutrino/water-cherenkov. Accessed: 2024-02-28.

[32] M. Ageron et al. 'ANTARES: The first undersea neutrino telescope'. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 656.1 (2011), pp. 11–38. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2011.06.103. URL: https://www.sciencedirect.com/science/article/pii/S0168900211013994.

[33] Paul de Jong. 'Legacy results of the ANTARES Neutrino Telescope'. In: *PoS* EPS-HEP2023 (2023), p. 061. DOI: 10.22323/1.449.0061.

[34] *KM3NET Members and detector sites.* https://www.km3net.org/about-km3net/collaboration/members/. Accessed: 2024-02-29.

[35] F.P. An et al. ‘The detector system of the Daya Bay reactor neutrino experiment’. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 811 (2016), pp. 133–161. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2015.11.144. URL: https://www.sciencedirect.com/science/article/pii/S0168900215015636.

[36] A.A. Aguilar-Arevalo et al. ‘The MiniBooNE detector’. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 599.1 (2009), pp. 28–46. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2008.10.028. URL: https://www.sciencedirect.com/science/article/pii/S0168900208015404.

[37] S. Fukuda et al. ‘The Super-Kamiokande detector’. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 501.2 (2003), pp. 418–462. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(03)00425-X. URL: https://www.sciencedirect.com/science/article/pii/S016890020300425X.

[38] KM3NeT collaboration. *KM3NET Members and detector sites.* https://www.km3net.org/wp-content/uploads/2016/02/KM3NeT-Telescope-1.jpg. Accessed: 2024-02-28.

[39] *KM3NeT Detection Unit design.* https://www.km3net.org/research/detector/sensors/. Accessed: 2024-02-12.

[40] *KM3NeT Cabeling.* https://www.km3net.org/research/detector/deep-sea-network/. Accessed: 2024-02-12.

[41] Yury Malyshkin. ‘Baikal-GVD neutrino telescope: Design reference 2022’. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1050 (2023), p. 168117. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2023.168117. URL: https://www.sciencedirect.com/science/article/pii/S0168900223001079.

[42] Matteo Agostini et al. ‘The Pacific Ocean Neutrino Experiment’. In: *Nature Astronomy* 4.10 (July 2020), pp. 913–915. ISSN: 2397-3366. DOI: 10.1038/s41550-020-1182-4. URL: http://dx.doi.org/10.1038/s41550-020-1182-4.

[43] Z. Ye et al. ‘Proposal for a neutrino telescope in South China Sea’. In: (2022). DOI: 10.48550/arXiv.2207.04519.

[44] V. Antonelli et al. *Solar Neutrinos.* 2012. arXiv: 1208.1356 [hep-ex].

[45] Adam Burrows and James M. Lattimer. ‘Neutrinos from SN 1987A’. In: *Astrophys. J. Lett.* 318 (1987), pp. L63–L68. DOI: 10.1086/184938.

[46] Mark Aartsen et al. ‘Neutrino emission from the direction of the blazar TXS 0506+056 prior to the IceCube-170922A alert’. In: *Science* 361.6398 (July 2018), pp. 147–151. ISSN: 1095-9203. DOI: 10.1126/science.aat2890. URL: http://dx.doi.org/10.1126/science.aat2890.

[47] IceCube Collaboration*† et al. ‘Evidence for neutrino emission from the nearby active galaxy NGC 1068’. In: *Science* 378.6619 (2022), pp. 538–543. DOI: 10.1126/science.abg3395. eprint: https://www.science.org/doi/pdf/10.1126/science.abg3395. URL: https://www.science.org/doi/abs/10.1126/science.abg3395.

[48] M. Ageron et al. ‘Dependence of atmospheric muon flux on seawater depth measured with the first KM3NeT detection units: The KM3NeT Collaboration’. In: *The European Physical Journal C* 80.2 (Feb. 2020). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-020-7629-z. URL: http://dx.doi.org/10.1140/epjc/s10052-020-7629-z.

[49] Steven Haddock, Mark Moline and James Case. ‘Bioluminescence in the Sea’. In: *Annual review of marine science* 2 (Jan. 2010), pp. 443–93. DOI: 10.1146/annurev-marine-120308-081028.

[50] N. Reeb et al. ‘Studying Bioluminescence Flashes with the ANTARES Deep Sea Neutrino Telescope’. In: *Limnology and Oceanography: Methods* 21.11 (2023), pp. 734–760. DOI: 10.1002/lom3.10578. arXiv: 2107.08063 [physics.ao-ph].

[51] Bjoern Herold. ‘Study of 40K-induced rates for a KM3NeT design option with multi-PMT optical modules’. In: *Nuclear Instruments 'I&' Methods in Physics Research Section A-accelerators Spectrometers Detectors and Associated Equipment - NUCL INSTRUM METH PHYS RES A* 626 (Jan. 2011). DOI: 10.1016/j.nima.2010.04.137.

[52] C. Van Der Malsburg. ‘Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms’. In: *Brain Theory*. Ed. by Günther Palm and Ad Aertsen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 245–248. ISBN: 978-3-642-70911-1.

[53] John Jumper et al. ‘Highly accurate protein structure prediction with AlphaFold’. In: *Nature* 596 (July 2021), pp. 1–11. DOI: 10.1038/s41586-021-03819-2.

[54] Mehmet Akdel et al. ‘A structural biology community assessment of AlphaFold2 applications’. In: *Nature Structural I&' Molecular Biology* 29 (Nov. 2022), pp. 1–12. DOI: 10.1038/s41594-022-00849-w.

[55] *AlphaGo beats World Champion*. https://www.bbc.com/news/technology-35785875. Accessed: 2024-02-17.

[56] Muhammad Usman Hadi et al. ‘Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects’. In: (July 2023). DOI: 10.36227/techrxiv.23589741.

[57] Yiheng Liu et al. ‘Summary of ChatGPT-Related research and perspective towards the future of large language models’. In: *Meta-Radiology* 1.2 (Sept. 2023), p. 100017. ISSN: 2950-1628. DOI: 10.1016/j.metrad.2023.100017. URL: http://dx.doi.org/10.1016/j.metrad.2023.100017.

[58] *Layout of a multi layer perceptron.* https://www.allaboutcircuits.com/uploads/articles/an-introduction-to-training-theory-for-neural-networks_rk_aac_image2.jpg. Accessed: 2024-02-28.

[59] Namig J. Guliyev and Vugar E. Ismailov. 'On the approximation by single hidden layer feedforward neural networks with fixed weights'. In: *Neural Networks* 98 (2018), pp. 296–304. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2017.12.007. URL: https://www.sciencedirect.com/science/article/pii/S0893608017302927.

[60] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams. 'Learning representations by back-propagating errors'. In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[61] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2017. arXiv: 1412.6980 [cs.LG].

[62] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. arXiv: 1502.03167 [cs.LG].

[63] Stefan Reck. 'Cosmic ray composition measurement using Graph Neural Networks for KM3NeT/ORCA'. PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2023. DOI: 10.25593/open-fau-135.

[64] *EdgeConv block package.* https://pypi.org/project/medgeconv/. Accessed: 2024-02-17.

[65] Yue Wang et al. 'Dynamic Graph CNN for Learning on Point Clouds'. In: *ACM Trans. Graph.* 38.5 (Oct. 2019). ISSN: 0730-0301. DOI: 10.1145/3326362. URL: https://doi.org/10.1145/3326362.

[66] Huilin Qu and Loukas Gouskos. 'Jet tagging via particle clouds'. In: *Phys. Rev. D* 101 (5 Mar. 2020), p. 056019. DOI: 10.1103/PhysRevD.101.056019. URL: https://link.aps.org/doi/10.1103/PhysRevD.101.056019.

[67] *OrcaNet by KM3NeT.* https://git.km3net.de/ml/OrcaNet. Accessed: 2024-02-17.

[68] D. Guderian. 'Development of detector calibration and graph neural network-based selection and reconstruction algorithms for the measurement of oscillation parameters with KM3NeT/ORCA'. PhD thesis. U. Munster, 2022.

[69] Hennig Lukas. 'Tau neutrino identification with Graph Neural Networks in KM3NeT/ORCA'. MA thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2023.

[70] R. Abbasi et al. 'Graph Neural Networks for low-energy event classification 'I&'amp; reconstruction in IceCube'. In: *Journal of Instrumentation* 17.11 (Nov. 2022), P11003. ISSN: 1748-0221. DOI: 10.1088/1748-0221/17/11/p11003. URL: http://dx.doi.org/10.1088/1748-0221/17/11/P11003.

[71] Andreas Søgaard et al. *GraphNeT: Graph neural networks for neutrino telescope event reconstruction*. 2022. arXiv: 2210.12194 [astro-ph.IM].

[72] Seongjun Yun et al. *Graph Transformer Networks*. 2020. arXiv: 1911.06455 [cs.LG].

[73] Habib Bukhari et al. *IceCube – Neutrinos in Deep Ice The Top 3 Solutions from the Public Kaggle Competition*. 2023. arXiv: 2310.15674 [astro-ph.HE].

[74] S. Aiello et al. 'gSeaGen: The KM3NeT GENIE-based code for neutrino telescopes'. In: *Computer Physics Communications* 256 (2020), p. 107477. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2020.107477. URL: https://www.sciencedirect.com/science/article/pii/S0010465520302241.

[75] *KM3NeT simulation software km3sim*. https://git.km3net.de/simulation/km3sim. Accessed: 2024-02-12.

[76] *KM3NeT software jpp incl. JSirene and JTriggerEffiency*. https://git.km3net.de/common/jpp. Accessed: 2024-02-12.

[77] *KM3NeT analysis software km3pipe*. https://git.km3net.de/km3py/km3pipe. Accessed: 2024-02-28.

[78] *KM3NeT software OrcaSong*. https://git.km3net.de/ml/OrcaSong. Accessed: 2024-02-12.

[79] *OscProb*. https://github.com/joaoabcoelho/OscProb. Accessed: 2024-02-21.

[80] Varun Godbole et al. *Deep Learning Tuning Playbook*. Version 1. 2023. URL: https://github.com/google-research/tuning_playbook.

[81] AN Tsoularis and James Wallace. 'Analysis of Logistic Growth Models'. In: *Mathematical biosciences* 179 (July 2002), pp. 21–55. DOI: 10.1016/S0025-5564(02)00096-2.

[82] Victor Carretero. 'Measuring atmospheric neutrino oscillation with KM3NeT/ORCA6'. In: *PoS* ICRC2023 (2023), p. 996. DOI: 10.22323/1.444.0996.

A Event numbers

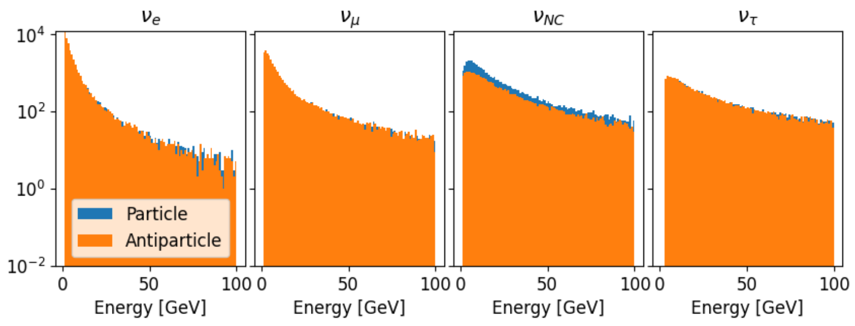


Figure A.1: *Number of events in the run-by-run dataset. Resolved for energy and separated by interaction type. Notation follows subsection 2.4.3.*

Run-by-run dataset

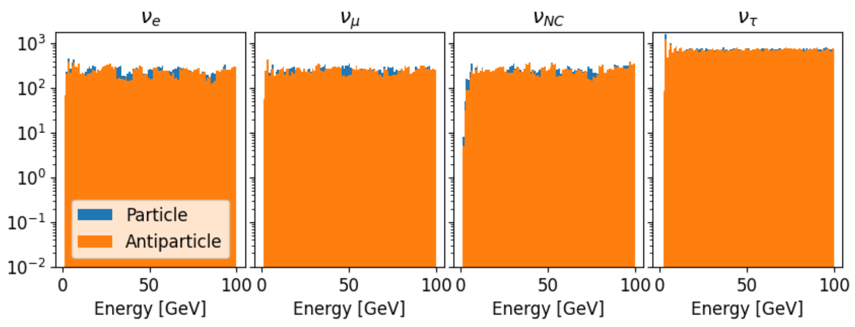


Figure A.2: *Number of events in the single-run dataset. Resolved for energy and separated by interaction type. Notation follows subsection 2.4.3.*

Single run dataset

B OrcaNet tomls

```
[model]
type = "DisjointEdgeConvBlock"
next_neighbors = 16
shortcut = true
activation = 'relu'

#particle net
blocks = [
    {units=[64, 64, 64], batchnorm_for_nodes=true},
    {units=[128, 128, 128]},
    {units=[256, 256, 256], pooling=true},
    {type='OutputRegNormal', output_name='energy',output_neurons=1}
]


[compile]
optimizer = "adam"
#optimizer = "sgd"

[compile.losses]
energy = {function='lkl_normal', metrics=["msle","mse","mae",]}
```

```
[config]
    learning_rate = [0.01, 0.004]
    train_logger_display = 200
    train_logger_flush = -1
    verbose_train = 1
    batchsize = 64
    shuffle_train = true
    use_custom_sample_weights = true
    validate_interval = 1
    sample_modifier = {
        name = 'GraphEdgeConv',
        column_names = ['pos_x','pos_y','pos_z',
            'time','dir_x','dir_y','dir_z'],
        knn = 16
    }
    label_modifier = {
        name='RegressionLabels',
        columns=['visible_energy'],
        model_output='energy',
        stacks=2,
        log10=true
    }



[points]

train_files = [
"/home/wecapstor3/capn/mppi132h/GNNs/data/sample_weight_files/
output/train/itype_ratio/05.h5",
]
validation_files = [
"/home/wecapstor3/capn/mppi132h/GNNs/data/sample_weight_files/
output/val/allIncTau_physical_standard.h5",
]
inference_files = []
```

C Relative fractional error for analysis

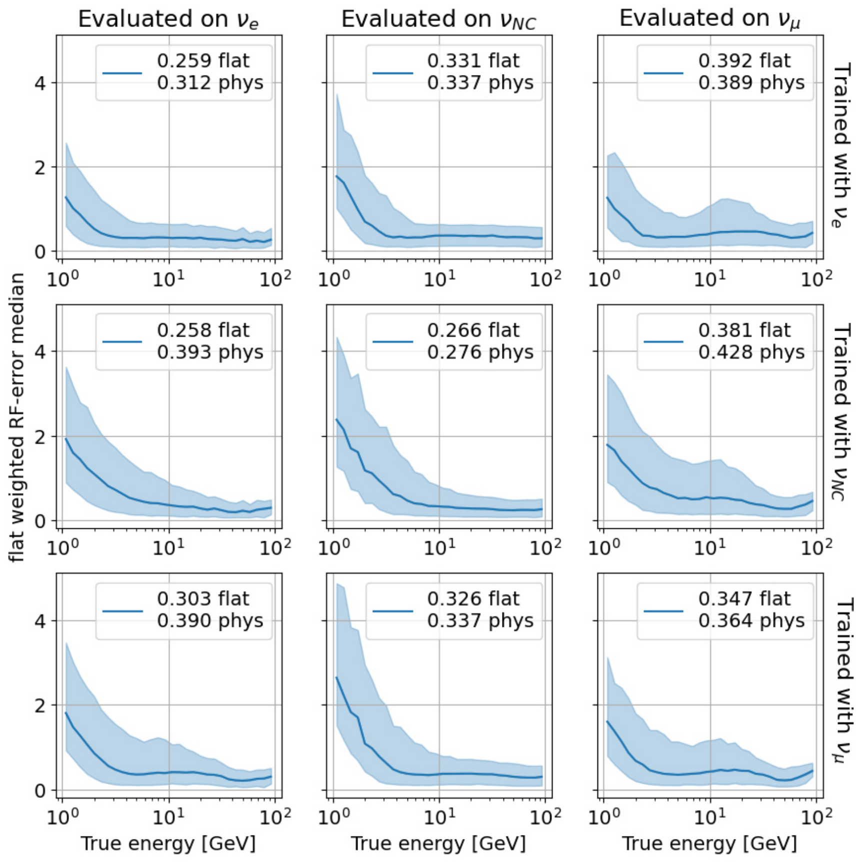


Figure C.1: *Single interaction networks trained and evaluated on* ν_e^{CC},ν_x^{NC} *and* ν_μ. *Plots show the median, 16 and 84 percentile of the flat weighted RFE resolved over true energy. Legend shows flat and physical RFE score. Notation follows subsection 2.4.3.*

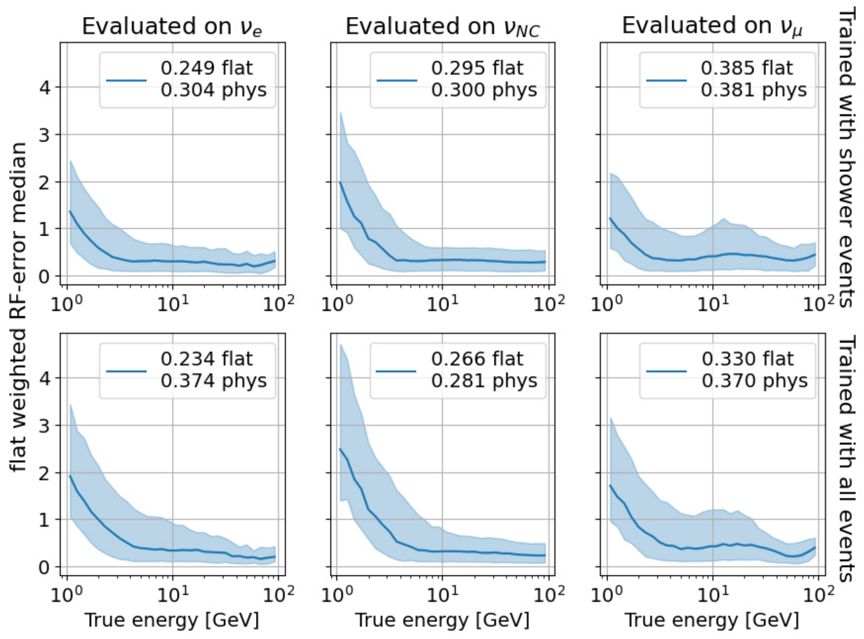


Figure C.2: *Multi interaction networks trained on shower and all interaction types. Evaluated on ν_e^{CC},ν_x^{NC} and ν_μ. Plots show the median, 16 and 84 percentile of the flat weighted RFE resolved over true energy. Legend shows flat and physical RFE score. Notation follows subsection 2.4.3.*

Eigenständigkeitserklärung

Hiermit versichere ich, Bastian Setter 22408546, die vorgelegte Arbeit selbstständig und ohne unzulässige Hilfe Dritter sowie ohne die Hinzuziehung nicht offengelegter und insbesondere nicht zugelassener Hilfsmittel angefertigt zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen und wurde auch von keiner anderen Prüfungsbehörde bereits als Teil einer Prüfung angenommen.
Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.
Mir ist insbesondere bewusst, dass die Nutzung künstlicher Intelligenz verboten ist, sofern diese nicht ausdrücklich als Hilfsmittel von dem Prüfungsleiter bzw. der Prüfungsleiterin zugelassen wurde. Dies gilt insbesondere für Chatbots (insbesondere ChatGPT) bzw. allgemein solche Programme, die anstelle meiner Person die Aufgabenstellung der Prüfung bzw. Teile derselben bearbeiten könnten.
Des Weiteren ist mir bekannt, dass die gemeinsame Bearbeitung der Aufgabenstellung mit anderen Personen in einem Raum oder mithilfe sozialer Medien eine unzulässige Hilfe Dritter im o.g. Sinne darstellt, wenn nicht ausdrücklich Gruppenarbeit vorgesehen ist. Jeder Austausch mit anderen Personen mit Ausnahme von Prüfenden und Aufsichtführenden während der Prüfungszeit über Aufbau oder Inhalte der Prüfung oder Informationen (z.B. Quellen) ist unzulässig. Gleiches gilt für den Versuch der jeweiligen Handlung.
Verstöße gegen die o.g. Regeln sind als Täuschung bzw. Täuschungsversuch zu qualifizieren und führen zu einer Bewertung der Prüfung mit „nicht bestanden“.

Erlangen, den 01.03.2024 **Eigenhändige Unterschrift**