# Low-Energy Neutrino Reconstructions using Recurrent Neural Networks

Bjørn H. Mølvig

SUPERVISORS:
Troels C. Petersen
Oswin Krause

Master's Thesis (cand.scient.)
The Niels Bohr Institute
Faculty of Science
University of Copenhagen

# ABSTRACT

The IceCube Neutrino Observatory is a physics experiment located at Antarctica. The experiment is capable of detecting neutrinos using light-detecting digital optical modules (DOMs) sitting in an array deep below the surface in the South Pole ice sheet. This thesis presents a versatile Deep Learning-based approach to low-energy neutrino reconstruction and -identification at the IceCube Experiment. The algorithms developed in this work attempt to alleviate some of the weaknesses of the current methods: The currently used state-of-the-art reconstruction algorithm, Retro Reconstruction, is a table-based maximum likelihood estimator able to reconstruct neutrinos at a rate of $O(10^{-2})$ reconstructions per second.

By employing recurrent neural networks (RNNs), the information available at a lower level in the form of Digital Optical Module readouts is fed directly to the reconstruction model, removing the need for high-level feature generation and parametrization of a likelihood-function. The developed models have been gathered in an ensemble of models capable of reconstructing neutrino energies, directions and interaction vertices atleast 10 % better across most of the $1 - 1000$ GeV range. In addition to the increased performance, the reconstruction speed is increased tremendously: The developed RNNs are capable of reconstrucing 5000+ events per second on an old consumer-grade GPU.

The developed architecture is general and is therefore capable of solving other tasks than regression. By applying the model to the task of classifying cascade- and track-like events, a classifier has been developed showing promising results: Across the GeV energy range, areas of 59.2 % - 97.7% under the receiver operator characteristic curves (ROC AUCs) are achieved.

# ACKNOWLEDGEMENTS

For the past year, I have been working on neutrino reconstruction in the Icecube experiment. The time spent would not have been so joyous, had it not been for the people surrounding me. First, I would like to thank my main supervisor Troels C. Petersen for always being enthusiastic and curious and for always aiding in finding the next goal to pursue. I would also like to thank my co-supervisor Oswin Krause. Without his almost wizard-like machine learning abilities, this thesis would have looked quite different.

Besides my two formal supervisors, I want to thank Tom Stuttard for serving as something very similar. Without your deep understanding of the detector, of neutrinos and the generation of data, it would not have been possible to generate this work.

A huge thanks to my partner in crime and office mate Mads Ehrhorn. You have been a tremendous contributor with regards to keeping spirits high. Less importantly, I could not have created this thesis without your expert knowledge and database management.

In addition to Mads, the rest of the office crew of Daniel Nielsen, Christian Michelsen, Helle Leerberg and Lau Mortensen aided in creating a relaxed, funny and enjoyable atmosphere. Thank you all for numerous physics- and machine learning-related (and many other!) discussions.

Lastly, I want to thank my significant other, Amanda Frisk; for keeping me reminded that there are other things in life than neutrinos.

# INTRODUCTION

The IceCube Neutrino Observatory is the largest neutrino experiment ever created. Encompassing a cubic kilometer of ice, the experiment battles the elusive nature of neutrinos by using volume making the experiment detect roughly 100.000 neutrinos per year.

When charged particle traverse the ice at superluminal speeds, they emit Cerenkov radiation. This radiation is detected by the 5160 digital optical modules (DOMs) placed on strings in a hexagonal array deep within the South Pole ice.

Converting the individual DOM detections to meaningful physics is a daunting task. Not only are the DOMs incredibly noisy, but the vast majority of events are background muons, making the need for robust, high performing reconstruction algorithms necessary. The current, state-of-the-art algorithm is a table-based maximum likelihood-approach capable of reconstructing neutrinos at a rate of $O(10^{-2})$ requiring vast amounts of memory.

The goal of this work is therefore simple: Use a deep learning approach, specifically recurrent neural network building blocks (the Long Short-Term Memory architecture and the Gated Recurrent Unit architecture), and more modern Attention-based architectures, to explore whether machine learning algorithms can be developed that are faster and better than the current method.

This work is therefore neither a high-energy particle physics project nor a theoretical machine learning project, but something in between; namely, an application of machine learning to a particle physics experiment - and hopefully readers from both fields will be able to learn something new.

This thesis is structured as follows: In Chapter 1, an introduction to particle physics phenomenology is given. Hereafter neutrinos and how they are detected in the IceCube Neutrino Observatory is introduced in Chapter 2. In Chapter 3 we switch gears and expand on the machine learning concepts used in this work. Then the used data is described in Chapter 4 along with how data was selected and prepared for modelling. In Chapter 5, the process of developing a core deep learning model is described, before finally evaluating and fine-tuning the developed models in Chapter 6. In the short Chapter 7, the entire work is evaluated and suggestions for future work is given.

# CONTENTS

# 1

# NEUTRINOS & THE STANDARD MODEL

The Standard Model of particle physics is the current best description of the particles that make up the world as we know it. In this chapter, a brief history of the neutrino will be given before introducing the Standard Model (SM) and the fascinating properties of neutrinos.

## 1.1 THE DISCOVERY OF THE NEUTRINO

The history of the neutrino takes its humble beginnings in 1896, where the French scientist Henri Becquerel discovered radioactivity [1]. During the exploration of this newly discovered phenomenon, a problem arose which proved difficult to solve: The electron energy-spectrum of the beta-decay. At the time it was believed that beta-decay was a process, where one particle decays into two, i.e. a process of the form

$$p_1 \to p_2 + p_3, \tag{1}$$

where $p_1$, $p_2$ and $p_3$ denote the respective particles involved. Such a process would require the decay-products to have discrete energies to ensure energy- and momentum conservation, but this was not observed. Instead, the energy of the electron was found to be distributed according to the continuous spectrum sketched in Figure 1.

*Interestingly, the cut-off of the spectrum can be used to determine the mass of $\nu_e$.*

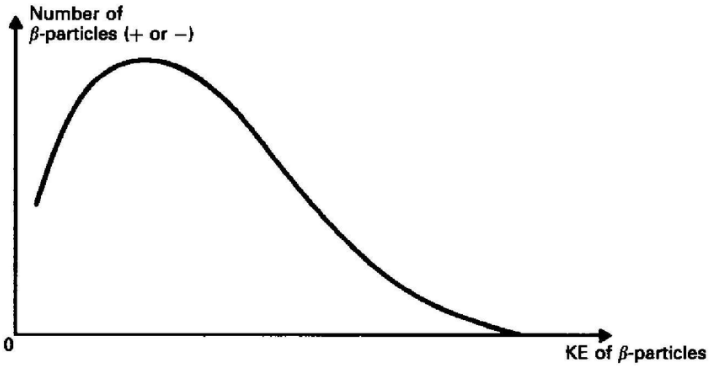



Figure 1: Illustration of the energy-spectrum of the electron in a $\beta$-decay with arbitrary units. Figure from [2].

The spectrum is consistent with the emission of a third particle, which led Pauli to postulate the existence of a neutral particle [3].[1]

1 Pauli coined the particle *neutron*("neutral one") in Italian - a few years later, the neutron as we know it was discovered by Chadwick [4]. Hence, the name of the third particle

The particle was confirmed to be the neutrino in 1956 by Reines and Cowan; a feat for which Reines was awarded the Nobel Prize in 1995 [5, 6].

## 1.2 THE STANDARD MODEL

The neutrino comes in 3 different flavors: the electron neutrino $\nu_e$, the muon neutrino $\nu_\mu$ and the tau neutrino $\nu_\tau$. They join the SM consisting of 17 fundamental particles shown in Figure 2.

The neutrinos are *neutral fermions* and along with the quarks (the *up, down, charm, strange, top* and *bottom*) and the *charged leptons* (the *electron, muon* and *tau*), they complete the family of *leptons* in the SM.

The fermions are half-integer spin fundamental particles described by the Dirac equation

$$\left(i\gamma^\mu \partial_\mu - m\right)\psi = 0, \tag{2}$$

where $\gamma^\mu$ are the gamma-matrices and the Einstein summation convention is employed along with natural units ($c = \hbar = 1$). Furthermore, each fermion has an antiparticle denoted by adding a bar over the particle symbol, e.g. $\bar{e}$ with the same mass and opposite charge.

*Notation such as $\bar{e}$ and $e^+$ for antiparticles will be used interchangeably.*

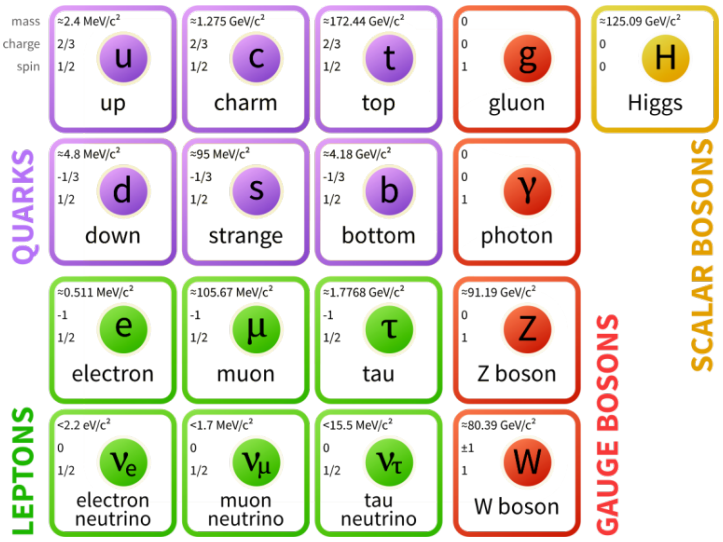



Figure 2: The particles of the Standard Model. Figure from [7].

The forces of the SM stem from local gauge symmetries. The fermions couple to the electromagnetic and weak forces (depending on their electric charge and weak hyperspin) through the electroweak (EWK) interaction with U(1)×SU(2)-symmetry mediated by the photon, the W-bosons and the Z-boson (denoted $\gamma, W^\pm$ and $Z$ in Figure 2). The

of the $\beta$-decay was changed to *neutrino* ("little neutral one" in Italian) as proposed by Fermi.

quarks couple to an additional force, the strong nuclear force, through the strong interaction described by quantum chromodynamics (QCD) with SU(3)-symmetry mediated by the gluons.

The interactions of the SM particles can be described by Feynman diagrams. For instance, the annihilation-process

$$e^+ + e^- \rightarrow \gamma + \gamma \tag{3}$$

creating a photon-pair is described by the Feynman diagram shown in Figure 3. In the Feynman diagram, time runs from left to right, and antiparticles are denoted with arrows pointing backwards in time. Each vertex of the diagram is associated with an expression for the interaction, which is used to calculate decay rates and cross sections.

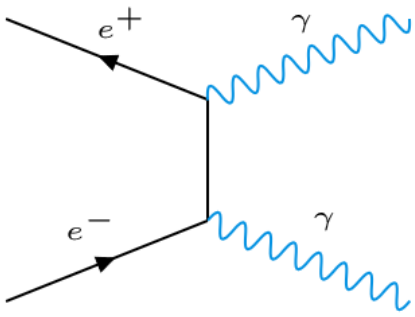



Figure 3: The Feynman diagram describing an electron-positron annihilation process.

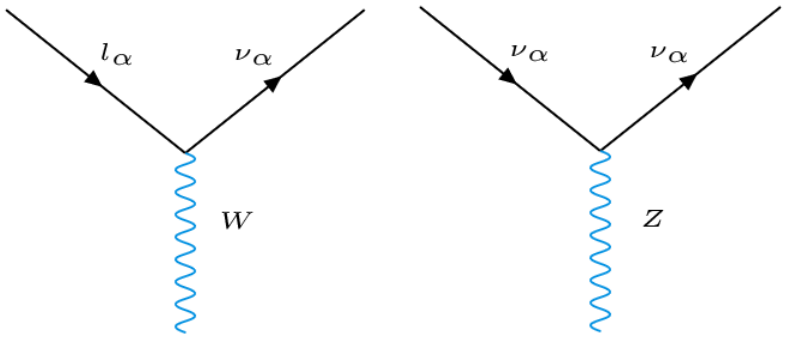
Except for the neutrinos, the massive particles of the SM acquire their masses through their interactions with the Higgs field, whose excitations manifest as Higgs-particles - the newest confirmed members of the SM [8]. Until recently, the neutrinos were believed to be massless, but with the discovery of neutrino oscillations, neutrinos were confirmed to be massive particles. Interestingly, there is currently no agreed upon mechanism for how the neutrinos acquire their masses.

### 1.2.1 *Neutrino Interactions*

As previously mentioned, neutrinos couple to the EWK through the weak interaction only, since the neutrinos have neutral electric charge. In 1957, Wu and collaborators observed that the weak interaction is not invariant with respect to parity-transformations [9]. The interaction is therefore described by a vector minus axial vector interaction (or *V-A*) of the form $\gamma^\mu(1-\gamma^5)$, where $\gamma^5 = i\gamma^0\gamma^1\gamma^2\gamma^3$. Specifically, the weak interaction vertex is given by

$$\frac{-ig_W}{\sqrt{2}}\frac{1}{2}\gamma^\mu(1-\gamma^5), \tag{4}$$

where $g_W$ is the weak coupling constant and the operator $\frac{1}{2}\gamma^\mu(1-\gamma^5)$ is identified as the left chiral projection operator $P_L$. This means that only left-handed (LH) neutrinos and right-handed (RH) anti-neutrinos can undergo weak interactions.




(a) (b)

Figure 4: Interaction vertices of interest in experimental neutrino physics. (a): Interaction vertex of a t-channel charged current weak interaction, where $\alpha$ denotes lepton flavour. (b): Interaction vertex of a t-channel neutral current weak interaction, where $\alpha$ denotes lepton flavour.

Generally, the neutrinos interact in two different ways: As charged current (CC) weak interactions and neutral current (NC) weak interactions. The interactions of biggest interest in experimental neutrino physics are depicted in Figure 4. In Figure 4a, a charged W-boson is exchanged in a CC process, where a charged lepton is transformed into a neutrino of the same flavour. The reverse process is also allowed, where a neutrino transforms into a charged lepton of the same flavour.

In Figure 4b, the NC interaction vertex is depicted. This could for instance be a neutrino of flavour $\alpha$, which interacts with a valence electron through scattering - a process used by the SNO-experiment in Canada to differentiate between neutrino flavours [10]. In the IceCube experiment, it is not possible to distinguish between neutrino flavours, when they interact via a neutral current; for all neutrino flavours, the final-state neutrino leaves undetected and the $Z$-boson deposits its energy.

Generally, the neutrinos in the IceCube detector are only indirectly observed through their subsequent interactions such as ionization and hadronisation of the final-state particles.

## 1.3 NEUTRINO OSCILLATIONS

The phenomenon of neutrino oscillations first received scientific interest in the 1950s. Theoretical models of the neutrino flux from the sun were not consistent with experimental evidence: The solar neutrino problem.

The problem will be discussed in the following section before turning our attention to its solution.

### 1.3.1 *The Solar Neutrino Problem*

In 1964, Bahcall and Davis published papers describing their search for solar neutrinos [11, 12]. Their experiment used a radiochemical technique for neutrino detection: The sun generates energy through a fusion process known as the proton-proton cycle. A product of this process is a neutrino. Such neutrinos interacted with 615 tons of $C_2Cl_4$ contained in the Homestake Mine in South Dakota, USA through the process

$$\nu_e + {}^{37}_{17}\mathrm{Cl} \rightarrow e^- + {}^{37}_{18}\mathrm{Ar}. \tag{5}$$

Subsequently, the interactions were counted by measuring the number of radioactive decays of Ar atoms. The experiment observed $0.48 \pm 0.04$ neutrinos per day, which was in disagreement with the theoretical prediction of 1.7 neutrinos per day [13].

The aforementioned SNO-experiment was able to resolve the problem. The experiment utilized a detector designed to measure all three flavors of neutrinos, and they reported a total solar neutrino flux consistent with solar models, hereby eproviding results consistent with neutrino oscillations [14].

### 1.3.2 *The PMNS-matrix*

A mechanism to explain why all three flavors of neutrinos were detected from the sun was given by Pontecorvo in 1957: Neutrino oscillations [15]. A model for neutrino oscillations was further developed by Maki, Nakagawa and Sakata in 1962 in which the weak eigenstates of the neutrinos are linear combinations of mass-states whose propagation obey the Dirac Hamiltonian given by Eq. (2) [16]. The weak eigenstates can therefore be related to the mass-states by a $3 \times 3$ unitary matrix

$$\begin{bmatrix} \nu_e \\ \nu_\mu \\ \nu_\tau \end{bmatrix} = \begin{bmatrix} U_{e1} & U_{e2} & U_{e3} \\ U_{\mu 1} & U_{\mu 2} & U_{\mu 3} \\ U_{\tau 1} & U_{\tau 2} & U_{\tau 3} \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} \tag{6}$$

named the PMNS-matrix. In general, a $3 \times 3$ complex matrix has 18 free parameters. The unitarity condition $U^\dagger U = I$ reduces this number to 3 mixing angles and 6 complex phases. 5 of those phases can be absorbed into the lepton states leaving 4 free parameters [17]. The PMNS-matrix is therefore often parameterized with the three mixing angles $\theta_{ij}$ and one complex phase $\delta_{CP}$ with CP-violating effects as

*The PMNS-matrix is also known as the MNSP-matrix*

$$\begin{bmatrix} U_{e1} & U_{e2} & U_{e3} \\ U_{\mu 1} & U_{\mu 2} & U_{\mu 3} \\ U_{\tau 1} & U_{\tau 2} & U_{\tau 3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{23} & s_{23} \\ 0 & -s_{23} & c_{23} \end{bmatrix} \begin{bmatrix} c_{13} & 0 & s_{13}e^{-i\delta_{CP}} \\ 0 & 1 & 0 \\ -s_{13}e^{i\delta_{CP}} & 0 & c_{13} \end{bmatrix} \begin{bmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where $c_{ij} = \cos(\theta_{ij})$ and $s_{ij} = \sin(\theta_{ij})$.
The solutions of Eq. (2) have the form

$$|\psi(t)\rangle = \exp(-iEt)\,|\psi(0)\rangle\,, \tag{8}$$

where $\psi$ is the quantum state and $E$ is the energy of the particle. The evolution of a neutrino state $|\psi(t)\rangle$ can therefore be written as

$$|\psi(t)\rangle = \sum_{i=1}^{3} U_{\alpha i}\,|\nu_i(t)\rangle\,, \tag{9}$$

where $\alpha$ denotes flavour and the time evolution is given by Eq. (8). To obtain oscillation probabilities, the neutrino quantum state given in Eq. (9) has to be expanded in the weak eigenbasis. Denoting the unitary matrix in Eq. (6) as $U$ and inverting it using the unitarity condition, the $i$'th mass-state expansion in the weak eigenbasis is

$$|\nu_i\rangle = \sum_{\alpha=e,\nu,\tau} U^*_{\alpha i}\,|\nu_\alpha\rangle\,. \tag{10}$$

Inserting Eq. (10) and Eq. (8) in Eq. (9), the oscillation probability $P(\nu_\alpha \to \nu_\beta)$ from a weak state $\alpha$ to a weak state $\beta$ can now be calculated as

$$\begin{aligned} P(\nu_\alpha \to \nu_\beta) &= |\langle \nu_\beta | \nu_\alpha \rangle|^2 \\ &= \left| \langle \nu_\beta | \left( \sum_{i=1}^{3} U_{\alpha i} \exp(-iE_i t) \sum_{\gamma=e,\nu,\tau} U^*_{\gamma i} |\nu_\gamma\rangle \right) \right|^2 \qquad (11) \\ &= \left| \sum_{i=1}^{3} U_{\alpha i} U^*_{\beta i} \exp(-iE_i t) \right|^2 \\ &= \sum_{i,j=1}^{3} U_{\alpha i} U^*_{\beta i} U^*_{\alpha j} U_{\beta j} \exp(-i(E_i - E_j)t). \qquad (12) \end{aligned}$$

From Eq. (11) it can be seen that if the phases of the exponential terms are different, oscillations between weak states are possible. To calculate oscillation probabilities of neutrinos with different momenta and without simplifications, a wavepacket treatment of the neutrino states is required [18]. In practice, however, assuming the neutrino states propagate as plane waves with identical momenta yields a satisfactory description for this work.

If we let $p_i = p_j = p$ in Eq. (11) and use the the Einstein energy-momentum relation

$$E^2 = p^2 + m^2, \tag{13}$$

the energy differences can be rewritten as

$$\begin{aligned} E_i - E_j &= p\left(\sqrt{1+\frac{m_i^2}{p^2}} - \sqrt{1+\frac{m_j^2}{p^2}}\right) \\ &\approx \frac{m_i^2 - m_j^2}{2p}, \end{aligned} \tag{14}$$

where only the first terms of the Taylor expansions of the square roots have been kept. If we approximate the neutrino energies and assume they propagate at the speed of light, we may set $t = L$ and $E \approx p$, where $L$ is the distance the wave has propagated. Hence, the exponential factors in Eq. (11) can be rewritten into

$$\exp(i(E_i - E_j)t) \approx 1 - 2\sin^2\left(\frac{\Delta m_{ij}^2 L}{4E}\right) + i\sin\left(\frac{\Delta m_{ji}^2 L}{2E}\right), \tag{15}$$

where $\Delta m_{ij}^2 = m_i^2 - m_j^2$. After some algebraic manipulations, the full oscillation probability therefore becomes

$$\begin{aligned} P(\nu_\alpha \to \nu_\beta) = \ & \delta_{\alpha\beta} - 2\sum_{i,j=1}^{3} \mathrm{Re}\left(U_{\alpha i}U_{\beta i}^* U_{\alpha j}^* U_{\beta j}\right)\sin^2\left(\frac{\Delta m_{ij}^2 L}{4E}\right) \\ & + \sum_{i,j=1}^{3} \mathrm{Im}\left(U_{\alpha i}U_{\beta i}^* U_{\alpha j}^* U_{\beta j}\right)\sin\left(\frac{\Delta m_{ji}^2 L}{2E}\right) \end{aligned} \tag{16}$$

From Eq. (16) it is seen that the oscillation probability frequencies depend on the squared mass differences of the neutrinos, their energy and distance travelled and the amplitudes on the PMNS matrix-elements. An example of the oscillation probability's dependence on energy and $\cos(\theta_{\text{zenith}})$ is shown in Figure 5. In this scenario, a $\nu_\mu$ is created in the atmosphere, travels through the earth and is detected in the IceCube detector, where the distance travelled $L = 2R_{\text{Earth}}\cos(\theta_{\text{zenith}})$ for $\cos(\theta_{\text{zenith}}) < 0$ .

*One can quickly confirm this relation by inspecting the isosceles triangle made by the radius of the Earth and the distance travelled of the neutrino.*

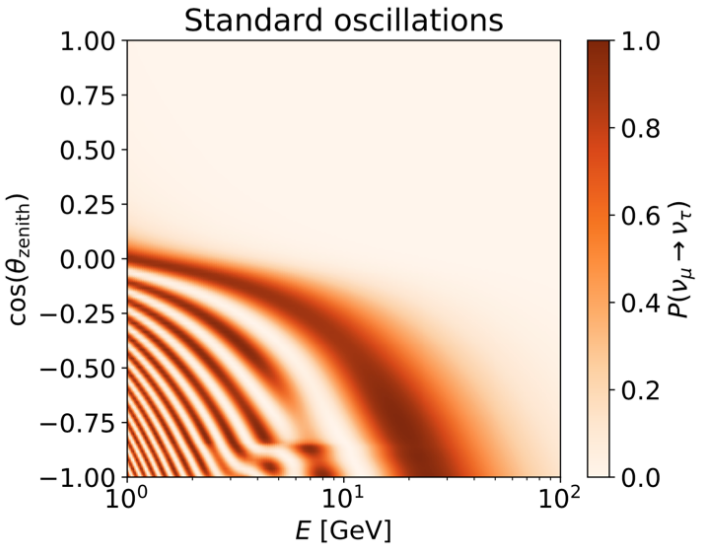



Figure 5: Example of $P(\nu_\mu \rightarrow \nu_\tau)$ for $\nu_\mu$ created in the atmosphere, where $\cos(\theta_{\text{zenith}})$ is the direction the neutrino came from. The zenith angle is directly related to the distance travelled. Figure courtesy of Tom Stuttard using neutrino parameters from [19].

To this point, the oscillation probability derivation has assumed oscillations in vacuum. When neutrinos propagate through a medium, electron neutrinos can interact with the electrons present in the medium, which directly affects $\nu_e$-oscillations and therefore indirectly affect $\nu_\mu$- and $\nu_\tau$-oscillations, since they oscillate into electron neutrinos. For neutrinos propagating in matter, this effect (named the MSW-effect) has to be taken into account [20, 21]. Matter effects are theoretically detectable with the IceCube detector: In Figure 5, a resonance due to the MSW-effect is visible around $E = 8$ GeV and $\cos(\theta_{\text{zenith}}) = -0.8$. Since this work explores neutrino reconstruction, we will not delve further into the topic of matter effects.

### 1.3.3 *Oscillation Results*

Neutrino oscillations can effectively be detected by either

- measuring deviations from the expected amount of neutrinos had they not oscillated, or
- by detecting neutrinos that are different from the neutrino flavour produced at the source.

Using these techniques, several of the oscillation parameters have been measured. The current, best global fits of the oscillation parameters are shown in Figure 6. The parameters are a combination of results from oscillation experiments from around the globe.

NuFIT 4.1 (2019)

| | | Normal Ordering (best fit) | | Inverted Ordering ($\Delta\chi^2 = 6.2$) | |
|---|---|---|---|---|---|
| | | bfp $\pm 1\sigma$ | $3\sigma$ range | bfp $\pm 1\sigma$ | $3\sigma$ range |
| without SK atmospheric data | $\sin^2\theta_{12}$ | $0.310^{+0.013}_{-0.012}$ | $0.275 \to 0.350$ | $0.310^{+0.013}_{-0.012}$ | $0.275 \to 0.350$ |
| | $\theta_{12}/^\circ$ | $33.82^{+0.78}_{-0.76}$ | $31.61 \to 36.27$ | $33.82^{+0.78}_{-0.76}$ | $31.61 \to 36.27$ |
| | $\sin^2\theta_{23}$ | $0.558^{+0.020}_{-0.033}$ | $0.427 \to 0.609$ | $0.563^{+0.019}_{-0.026}$ | $0.430 \to 0.612$ |
| | $\theta_{23}/^\circ$ | $48.3^{+1.1}_{-1.9}$ | $40.8 \to 51.3$ | $48.6^{+1.1}_{-1.5}$ | $41.0 \to 51.5$ |
| | $\sin^2\theta_{13}$ | $0.02241^{+0.00066}_{-0.00065}$ | $0.02046 \to 0.02440$ | $0.02261^{+0.00067}_{-0.00064}$ | $0.02066 \to 0.02461$ |
| | $\theta_{13}/^\circ$ | $8.61^{+0.13}_{-0.13}$ | $8.22 \to 8.99$ | $8.65^{+0.13}_{-0.12}$ | $8.26 \to 9.02$ |
| | $\delta_{\rm CP}/^\circ$ | $222^{+38}_{-28}$ | $141 \to 370$ | $285^{+24}_{-26}$ | $205 \to 354$ |
| | $\frac{\Delta m^2_{21}}{10^{-5}\ \text{eV}^2}$ | $7.39^{+0.21}_{-0.20}$ | $6.79 \to 8.01$ | $7.39^{+0.21}_{-0.20}$ | $6.79 \to 8.01$ |
| | $\frac{\Delta m^2_{3\ell}}{10^{-3}\ \text{eV}^2}$ | $+2.523^{+0.032}_{-0.030}$ | $+2.432 \to +2.618$ | $-2.509^{+0.032}_{-0.030}$ | $-2.603 \to -2.416$ |
| | | Normal Ordering (best fit) | | Inverted Ordering ($\Delta\chi^2 = 10.4$) | |
| | | bfp $\pm 1\sigma$ | $3\sigma$ range | bfp $\pm 1\sigma$ | $3\sigma$ range |
| with SK atmospheric data | $\sin^2\theta_{12}$ | $0.310^{+0.013}_{-0.012}$ | $0.275 \to 0.350$ | $0.310^{+0.013}_{-0.012}$ | $0.275 \to 0.350$ |
| | $\theta_{12}/^\circ$ | $33.82^{+0.78}_{-0.76}$ | $31.61 \to 36.27$ | $33.82^{+0.78}_{-0.75}$ | $31.61 \to 36.27$ |
| | $\sin^2\theta_{23}$ | $0.563^{+0.018}_{-0.024}$ | $0.433 \to 0.609$ | $0.565^{+0.017}_{-0.022}$ | $0.436 \to 0.610$ |
| | $\theta_{23}/^\circ$ | $48.6^{+1.0}_{-1.4}$ | $41.1 \to 51.3$ | $48.8^{+1.0}_{-1.2}$ | $41.4 \to 51.3$ |
| | $\sin^2\theta_{13}$ | $0.02237^{+0.00066}_{-0.00065}$ | $0.02044 \to 0.02435$ | $0.02259^{+0.00065}_{-0.00065}$ | $0.02064 \to 0.02457$ |
| | $\theta_{13}/^\circ$ | $8.60^{+0.13}_{-0.13}$ | $8.22 \to 8.98$ | $8.64^{+0.12}_{-0.13}$ | $8.26 \to 9.02$ |
| | $\delta_{\rm CP}/^\circ$ | $221^{+39}_{-28}$ | $144 \to 357$ | $282^{+23}_{-25}$ | $205 \to 348$ |
| | $\frac{\Delta m^2_{21}}{10^{-5}\ \text{eV}^2}$ | $7.39^{+0.21}_{-0.20}$ | $6.79 \to 8.01$ | $7.39^{+0.21}_{-0.20}$ | $6.79 \to 8.01$ |
| | $\frac{\Delta m^2_{3\ell}}{10^{-3}\ \text{eV}^2}$ | $+2.528^{+0.029}_{-0.031}$ | $+2.436 \to +2.618$ | $-2.510^{+0.030}_{-0.031}$ | $-2.601 \to -2.419$ |

(a)

NuFIT 4.1 (2019)

$$|U|^{\text{w/o SK-atm}}_{3\sigma} = \begin{pmatrix} 0.797 \to 0.842 & 0.518 \to 0.585 & 0.143 \to 0.156 \\ 0.244 \to 0.496 & 0.467 \to 0.678 & 0.646 \to 0.772 \\ 0.287 \to 0.525 & 0.488 \to 0.693 & 0.618 \to 0.749 \end{pmatrix}$$

$$|U|^{\text{with SK-atm}}_{3\sigma} = \begin{pmatrix} 0.797 \to 0.842 & 0.518 \to 0.585 & 0.143 \to 0.156 \\ 0.243 \to 0.490 & 0.473 \to 0.674 & 0.651 \to 0.772 \\ 0.295 \to 0.525 & 0.493 \to 0.688 & 0.618 \to 0.744 \end{pmatrix}$$

(b)

Figure 6: The global, current best estimates of the oscillation parameters as of July 2019[19, 22]. SK is the Super-Kamiokande experiment.

Interestingly, the oscillation probabilities depend on the *squared mass differences*, and therefore oscillation experiments are unable to determine the absolute values of the neutrino masses. Furthermore, a consequence of the dependence on differences leads to two, equally possible mass hierarchies coined the normal hierarchy shown in the left panel of

Figure 6a where $m_3 > m_2$, and the inverted hierarchy shown in the right panel of Figure 6a where $m_3 < m_2$.

### 1.3.4 *Sterile neutrinos*

Besides the neutrinos, all fermions of the SM can exist in both RH and LH states - a fact that is both experimentally confirmed and theoretically well grounded. Additionally, from EWK experiments it has been confirmed that there are exactly 3 active neutrinos coupling to the Z-boson as shown in Figure 7 [23].

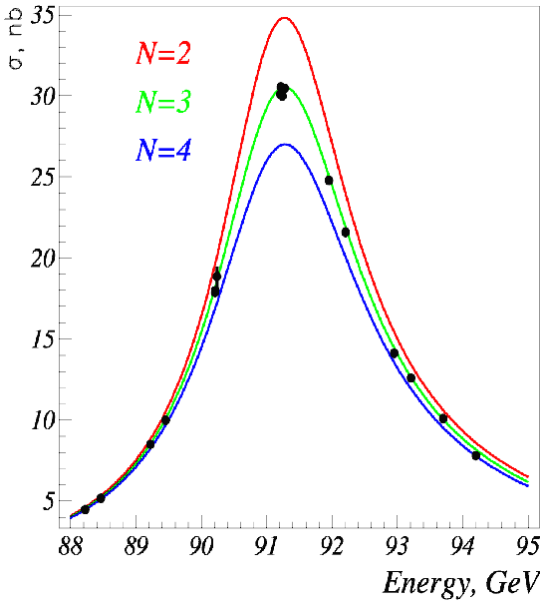



Figure 7: Cross section of the Z-resonance as a function of energy with fits of two, three and four active neutrinos. It can be seen that the data is incompatible with the existence of any more or any less active neutrinos than three. From [24].

There is, however, nothing in the SM that disallows the existence of neutrinos that does not couple to the EWK interaction. Taking these considerations into account, it is theoretically well motivated to extend the neutrino sector to include RH neutrinos and LH antineutrinos, the so called *sterile neutrinos* $\nu_s$. Sterile neutrinos only interact gravitationally and could potentially be dark matter [25].

Neutrino oscillations are straightforwardly extended to include sterile neutrinos by adding an additional dimension to Eq. (6) such that

$$\begin{bmatrix} \nu_e \\ \nu_\mu \\ \nu_\tau \\ \nu_s \end{bmatrix} = \begin{bmatrix} U_{e1} & U_{e2} & U_{e3} & U_{e4} \\ U_{\mu 1} & U_{\mu 2} & U_{\mu 3} & U_{\mu 4} \\ U_{\tau 1} & U_{\tau 2} & U_{\tau 3} & U_{\tau 4} \\ U_{s1} & U_{s2} & U_{s3} & U_{s4} \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \end{bmatrix}. \tag{17}$$

Now oscillation probabilites to and from $\nu_s$ can be derived in the same manner as Eq. (16). Currently, the only way of observing sterile neutrinos is through their oscillations into weakly interacting neutrinos.

# 2

# OBSERVING NEUTRINOS

Neutrinos are incredibly elusive particles. They can travel through lightyears of lead without interacting, which requires neutrino detectors to generally be very large to detect a satisfactory amount of neutrinos. In this chapter the IceCube detector will be introduced before discussing one source of neutrinos, the atmospheric neutrinos, and how they are detected.

## 2.1 THE ICECUBE DETECTOR.

The IceCube Neutrino Observatory is the largest neutrino detector in the world utilizing a volume of 1 $km^3$ of ice at the South Pole. The main array encompasses the largest volume of the detector, and a specialized array, DeepCore, is placed in the center of the detector.

### 2.1.1 *Detector geometry*

The detector is shown in Figure 8: The detector consists of an in-ice array of 5160 Digital Optical Modules (DOMs) spread across 86 vertical strings [26] equipped with (among other devices) PhotoMultiplier Tubes (PMTs). The strings are placed in a hexagonal structure with a spacing of 125 meters separation, and the vertical distance between DOMs is 17 meters. All DOMs are situated in the optically clearest ice between 1450 and 2450 meters below surface with the deepest DOMs located approximately 400 meters above the South Pole bedrock. The region between 2000 and 2100 meters does not contain DOMs, as this region has significantly higher light absorption due to a dustlayer deposited 65.000 years ago during the last glacial period [27].

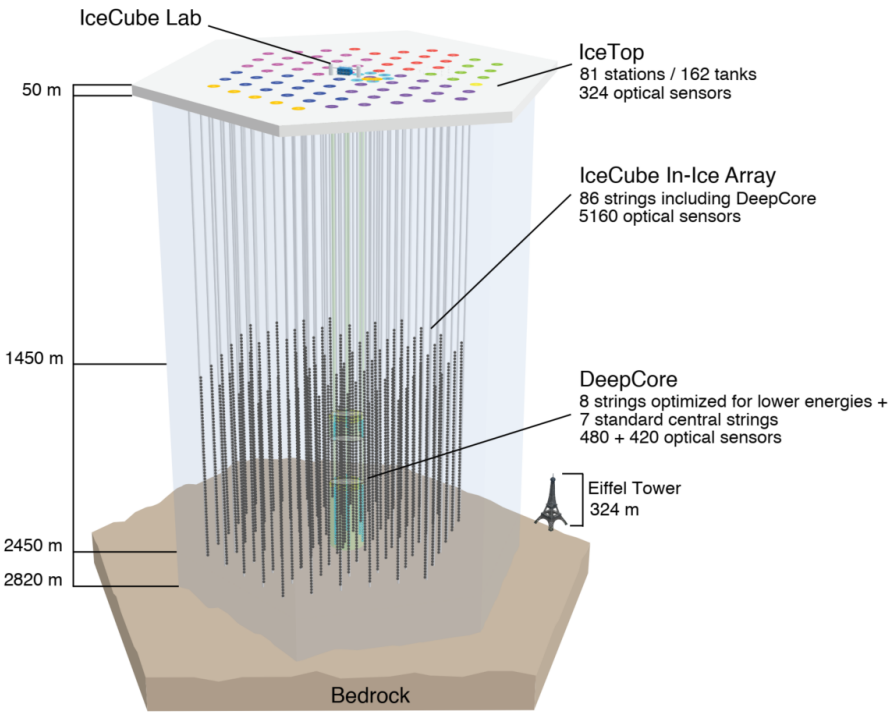
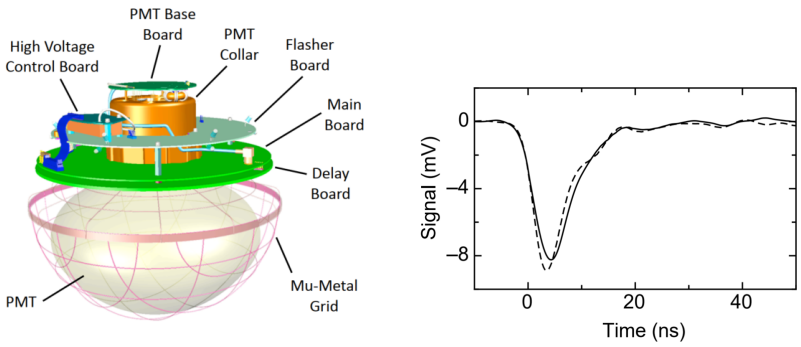



Figure 8: The IceCube Neutrino Observatory detector. The black dots located on the gray strings are the Digital Optical Modules used for light detection. In the center of the detector, the DeepCore-array optimized for low-energy particle detection is shown. Figure from [26].

In the center of the detector, the assymetric DeepCore-array is located, where the density of DOMs is higher: The average horizontal spacing between 13 of the strings is 72 meters and for 6 strings 42 meters [28]. The vertical spacing is also decreased to 7 meters. Furthermore, DeepCore is installed with PMTs with a higher quantum efficiency. All combined, the DeepCore-array greatly enhances the Icecube-detectors ability to observe particles with energies in the low GeV-range.

### 2.1.2 *The Digital Optical Module*

The DOM is the fundamental unit of IceCube and is shown in Figure 9a. It contains a downward-facing PMT sitting inside a spherical glass shell, power supplies and the electronics responsible for digitizing and sending acquired data to the IceCube laboratory at surface level. The PMTs are sensitive to light with wavelengths between 300 and 650 nm with a peak quantum efficiency of 25 % outside and 34 % inside DeepCore [29].




(a) (b)

Figure 9: (a): A Digital Optical Module (DOM). The DOM is the unit responsible for detecting light from events, digitizing the data and sending it to the surface. Figure from [26]. (b): Average of 10.000 single photoelectron waveforms recorded using two different transformer designs. Figure from [29].

Each DOM is equipped with two different kinds of digitizers: A fast Analog to Digital Converter (fADC) and an Analog Transient Waveform Digitizer (ATWD) [30]. The fADC uses a smaller sampling rate for conversion but also records for a longer time period compared to the ATWD as summarized in Table 1. When a PMT triggers, it produces a continuous signal referred to as a waveform; an average of several single photoelectron waveforms is shown in Figure 9b.

| Chip | Sampling rate | Record time |
|---|---|---|
| fADC | 40 MHz | 6.4 $\mu$s |
| ATWD | 300 MHz | 0.43 $\mu$s |

Table 1: Summary of digitizer characteristics.

Upon detection, the waveform is digitized and saved locally for a small period of time in the DOM. If the instrumentation deems a signal worthy of further processing, it is sent to the surface, where it is fitted and saved: The processed waveform, referred to as a *pulse*, can now be characterized by its width and integrated charge $Q$ and is saved along with its detection time for further analysis.

### 2.1.3 *Event Triggering*

The IceCube detector is inherently noisy: Standard PMTs have dark noise rates[1] of up to 500 Hz and high quantum efficiency PMTs as high as 800 Hz [29]. As a rough estimate, for an event in a time window of 10 $\mu$s, the noise rate corresponds to pulses from 25 DOMs originating from dark noise only. It is therefore of utmost importance to have methods

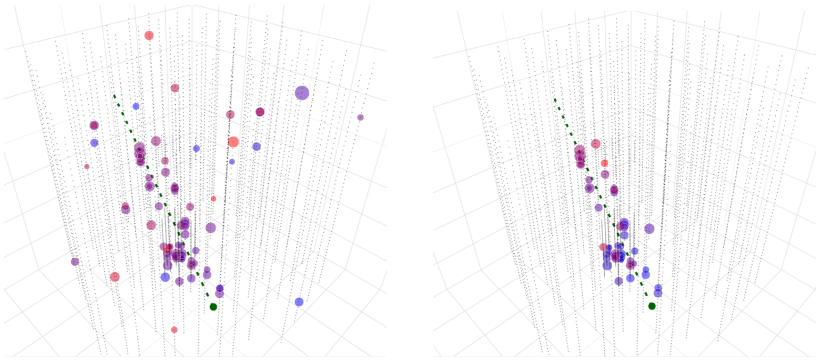
1 An umbella term with contributions from e.g. Poissonian noise and radioactive decays.

capable of separating noise from pulses stemming from true signal photons.

IceCube operates with a set of triggers. All DOMs in a given time window are read out and persistently saved if a trigger occurs. The **SMT8** trigger requires that the Hard Local Coincidence (HLC) condition is met for 8 or more DOMs within a time window of 5 $\mu$s [28]. The HLC condition requires that two or more neighboring or next-nearest neighboring DOMs register hits within a time interval of 1 $\mu$s. On the SMT8 trigger, all DOMs that register hits within 10 $\mu$s before and after the trigger time are read out. The DeepCore-detector employs an additional trigger, the The **SMT3** trigger, requiring only 3 DeepCore-DOMs to pass the HLC-condition within a 2.5 $\mu$s window.

### 2.1.4 *Pulse Cleaning*

As mentioned in Section 2.1.3, each readout is expected to contain a significant amount of pulses that do not emerge from a true signal. Attempting to remove the pulses stemming from noise has both advantages and disadvantages. Since there is no way of knowing for sure whether a pulse is noise or signal, the most obvious disadvantage is the risk of removing signal from an event. Furthermore, removing noise is not guaranteed to make machine learning (ML) algorithms perform better. On the other hand, if the sequence length of an event can be made smaller without removing signal, event processing times will decrease. Hence, it is worth investigating whether event cleaning can make ML algorithms more performant.



(a) Uncleaned event (b) Cleaned event

Figure 10: Uncleaned and SRT-cleaned simulated $\nu_\mu$ event with an energy of $E = 848$ GeV. Small black dots denote DOMs that did not activate during the event and coloured denote activated DOMs. The size of activated DOMs is proportional to the integrated charge of the pulse, and the colour runs from earlier times in blue to later times in red. The green dotted line is the true track. Figure courtesy of Mads Ehrhorn.

In this work, algorithms will be developed utilizing uncleaned events and Seeded Radius-Time cleaned (SRT) events. SRT is a combination of cleaning based on distances in space and time between DOMs and whether hits satisfy the HLC-condition: The RT-part of the cleaning only retains pulses that have atleast 1 accompanying hit within a radius of $R = 150$ m *and* is within a distance in time of $|\delta t| = 1$ $\mu$s. To further remove noise hits, an iterative addition process is exectuted. Starting out by putting all DOMs satisfying the HLC-condition into a list $l_{\text{DOMs}}$, then

- find all hits $H$ passing RT-cleaning with respect to $l_{\text{DOMs}}$.
- Update $l_{\text{DOMs}}$ with $H$.

This process is continued until the amount of kept DOMs stops increasing. The effect of SRT-cleaning is displayed in Figure 10 for a 848 GeV $\nu_\mu$-event. SRT cleaning has reduced the sequence length from 89 to 60 corresponding to the estimation of 29 noise pulses. For lower energy events, the fractional reduction is even larger, drastically reducing the number of operations required per reconstruction.

## 2.2 ATMOSPHERIC NEUTRINOS

In 1912, Victor Hess ascended from the surface of the Earth to just above 5000 metres in hot air balloons to measure the level of radioactivity in the atmosphere. He found that it increased dramatically with height above surface level [31]; in 1936 he was awarded the Nobel prize for the discovery of cosmic rays [32]. Today, cosmic rays and their origin is extensively studied, and understanding their compositions and quantities is important with regards to modelling the expected fluxes of the different neutrinos. Neutrinos generated through the interactions of the cosmic rays with the atmosphere, so-called atmospheric neutrinos, are the main source of neutrinos for oscillation analyses in Icecube.

Cosmic rays are stable, charged nuclei originating both from within and outside our galaxy that hit the Earths atmosphere. Cosmic ray energies vary wildly from a few to millions of GeV as shown in Figure 11. The different fluxes can be seen to approximately be described by powerlaws decreasing with energy with the total flux approximately decreasing like

$$\frac{\mathrm{d}N}{\mathrm{d}E} \sim E^{-2.7} \tag{18}$$

in the GeV and low TeV ranges. The by far most dominating contributors to the ray spectrum are hydrogen- and helium-cores making up approximately 94 % of all rays [33].

*For higher energies (above 'The Knee'), the exponent gets smaller.*

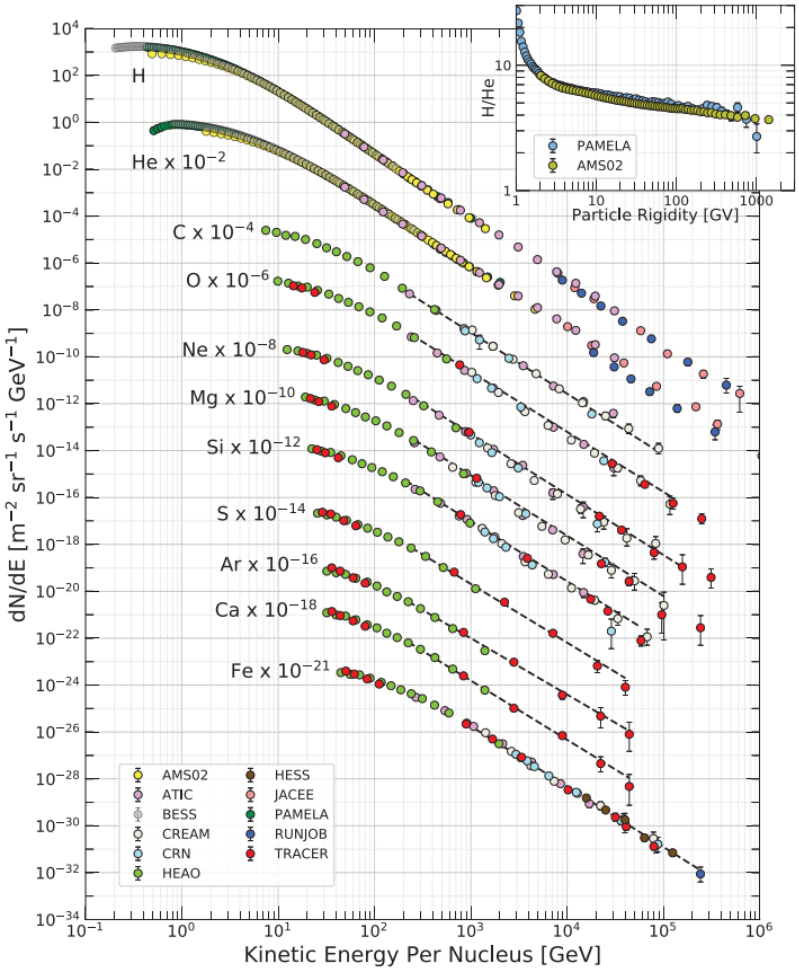



Figure 11: Cosmic ray flux and composition as a function of energy per nucleus. In the top right, the fraction of hydrogen- to helium-cores in rays as a function of particle rigidity (a measure of particle momentum) is shown. Note that the different fluxes have been scaled. Figure from [33].

When a highly energetic particle hits the Earth, it is very likely to interact in the upper part of the atmosphere producing a plethora of particles. Of particular interest to this work are the charged $\pi$-mesons. Being the lightest mesons, they are produced in the largest amount, which in turn produces the largest amount of $\nu_\mu$ and $\overline{\nu}_\mu$ through the interaction chains depicted in Figure 12, where for instance (not differentiating between particles and antiparticles)

$$\begin{aligned} \pi &\to \mu + \nu_\mu, \text{ then} \\ \mu &\to \nu_\mu + \nu_e + e, \end{aligned} \tag{19}$$

where the production of $\nu_\mu$ is twice that of $\nu_e$. Since the pion is not heavy enough to create $\tau$-particles and pions very rarely decay to electrons due to helicity suppresion[34], the production of different

neutrino flavours from pions are expected to roughly correspond to a $\nu_e : \nu_\mu : \nu_\tau$ ratio of $1 : 2 : 0$.

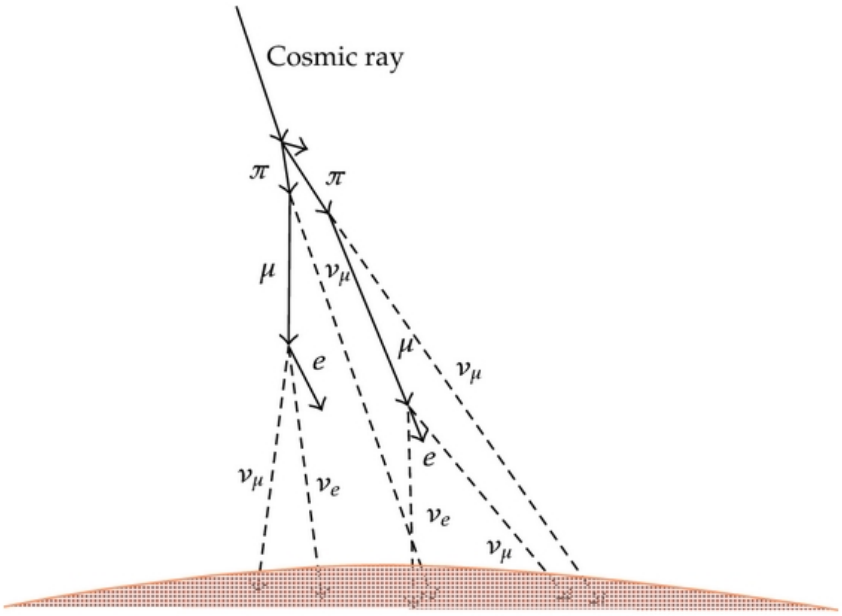



Figure 12: Sketch of how a cosmic ray produces interaction chains ultimately producing electron- and muon neutrinos. Figure from [35].

## 2.3 PARTICLE ENERGY LOSS

In the Icecube-detector, the neutrinos are not observed directly, but the products of their interactions are. In general, CC processes of the forms

$$\nu_\alpha + N \to l_\alpha^- + X \tag{20}$$

or

$$\overline{\nu_\alpha} + N \to l_\alpha^+ + X \tag{21}$$

take place in or near the detector, where the interaction products $l_\alpha^\pm$ and $X$ then deposit their energy in the detector leaving smoking gun evidence of a neutrino.

Consequently, the neutrino direction cannot be determined exactly; the angular difference $\Delta\Psi$ between the neutrino and the lepton directions roughly follow [36]

$$\Delta\Psi \approx 0.7^\circ \left(\frac{E_\nu}{\text{TeV}}\right)^{-0.6}. \tag{22}$$

Depending on the neutrino energy, the produced lepton loses energy in the form of photons primarily due to ionization or radiative effects.

For muons below 100 GeV, the primary source of energy loss is ionization for which the mean energy loss rate is well-described by the Bethe relation [33]

$$\left\langle -\frac{dE}{dx} \right\rangle = Kz^2 \frac{Z}{A} \frac{1}{v^2} \left( \frac{1}{2} \ln \frac{2m_e v^2 \gamma^2 W_{\max}}{I^2} - v^2 \right), \tag{23}$$

where $K = 4\pi N_A r_e^2 m_e$, $z$ is the charge number of the incident particle, $Z$, $A$ and $I$ are the atomic number, molar mass and mean excitation energy of the absorbing material and $W_{\max}$ is the maximal energy loss in a single collision. For energies closer to the TeV-range and beyond, radiative effects such as Bremsstrahlung take over as the main source of energy loss. A classical derivation of the radiated effect from Bremsstrahlung yields [37]

*A more thorough description of the energy loss is available in* [33].

$$\frac{dE}{dt} = \frac{2}{3} e^2 \gamma^6 \left( \dot{\vec{v}}^2 - (\vec{v} \times \dot{\vec{v}})^2 \right), \tag{24}$$

which in the relativistic limit (where $t \approx x$) becomes directly comparable to Eq. (23). The energy loss per unit distance for a muon travelling through copper is shown in Figure 13, where the energy region interesting to this work is from 1 GeV and up. The Bethe-regime is visible in the central part, and the radiative regime in the rightmost part of the figure. The IceCube detector does not utilize copper but ice as the absorbing medium, whose stopping power is similar.

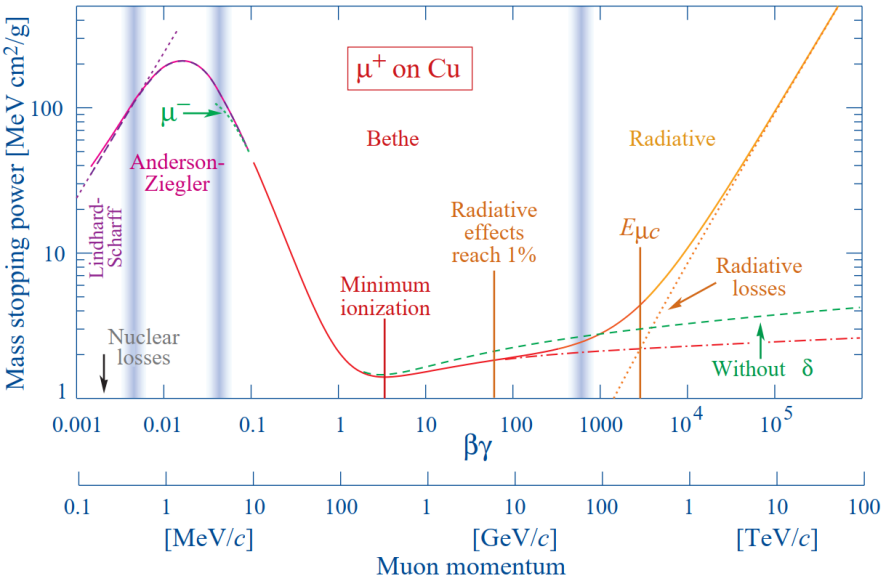



Figure 13: Energy loss per unit distance travelled for a muon propagating through copper with ionization (10 MeV - 100 GeV) and radiative regimes (>500 GeV). Figure from [33].

## 2.4 CERENKOV RADIATION

If charged particles travel through a dielectric medium faster than the phase velocity of light, they emit *Cerenkov Radiation* [38]. The phenomenon can be understood by examining how the wavefronts of the polarization created by the charged particle propagates: A superluminal, charged particle (indicated by the red arrow in Figure 14) polarizes a medium at $t = 0$, which propagates as a wave with phase velocity $v$. At a later time $t$, when the particle has travelled a distance $\beta t$, where $\beta = v_{particle}/c$ is the speed of the particle, another wave is emitted.

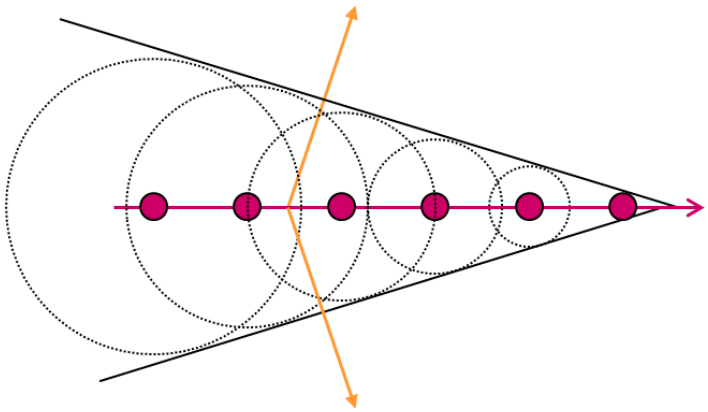


Figure 14: Sketch of the relation between wavefronts in Cerenkov radiation: A charged particle travels at superluminal speed along the red arrow, polarizing the dielectric medium. Due to geometry, constructive interference happens in certain directions, producing visible light. Figure from [39].

From the geometry of the figure, we see that the waves constructively interfere at an angle $\theta_C$, where $\cos\theta_C = \frac{vt}{\beta t}$, creating Cerenkov radiation. The speed of light is related to the index of refraction $n = \frac{c}{v}$, and therefore the angle $\theta_C$ of emission of Cerenkov radiation is

$$\cos\theta_C = \frac{1}{n\beta}. \tag{25}$$

Due to rotational symmetry, the superluminal particle emits cones of light. IceCube among other detectors utilize this effect to detect charged particles by detecting the Cerenkov radiation they emit. Assuming the charged particles have velocities of $\beta \approx 1$ and ice with a refractive index of 1.33, $\theta_C = 41.2$ degrees. The wavelength of the radiation and the amount of photons emitted per unit tracklength is determined by the Frank-Tamm formula [33]

*In reality, the refractive index in the South Pole ice is not a constant.*

$$\frac{dN}{dxd\lambda} = \frac{2\pi\alpha z^2}{\lambda^2}\left(1 - \frac{1}{\beta^2 n^2(\lambda)}\right), \tag{26}$$

where $\alpha$ is the fine-structure constant and $z$ is the charge number. It can be seen that the amount of radiation decreases with the inverse square of wavelength, and therefore Cerenkov detectors should be optimized towards detecting light towards the blue end of the light spectrum. In the IceCube detector, on the order of $O(10^5)$ visible photons are emitted per GeV of particle energy[26].

## 2.5 EVENT SIGNATURES

Due to the different natures of the lepton flavours, they leave different signatures in the detector. Since a $\tau$-particle decays very quickly to other particles, it does not travel very far. With a lifetime $\tau_\tau = 0.29 \cdot 10^{-12}$ seconds, low-energy taus have a characteristic travel length of approximately $c\tau_\tau = 87 \cdot 10^{-6}$ meters (neglecting relativistic time dilation). Therefore, to detect a $\tau$-particle, one must detect its decay products.

The electron does not travel very far in the detector either. Using the relation $E = \gamma m_0$, we can reexpress Eq. (24) in terms of $E/m$. Hence, in the relativistic limit

$$\frac{dE}{dx} \propto m^{-6}, \tag{27}$$

and therefore electrons quickly deposit all their energy, which in turn produce hadronic cascades and electromagnetic showers. Ultimately, it is the muons that leave the longest tracks in the detector, since they are stable enough to not decay too fast and heavy enough to not quickly lose all their energy as Bremsstrahlung.

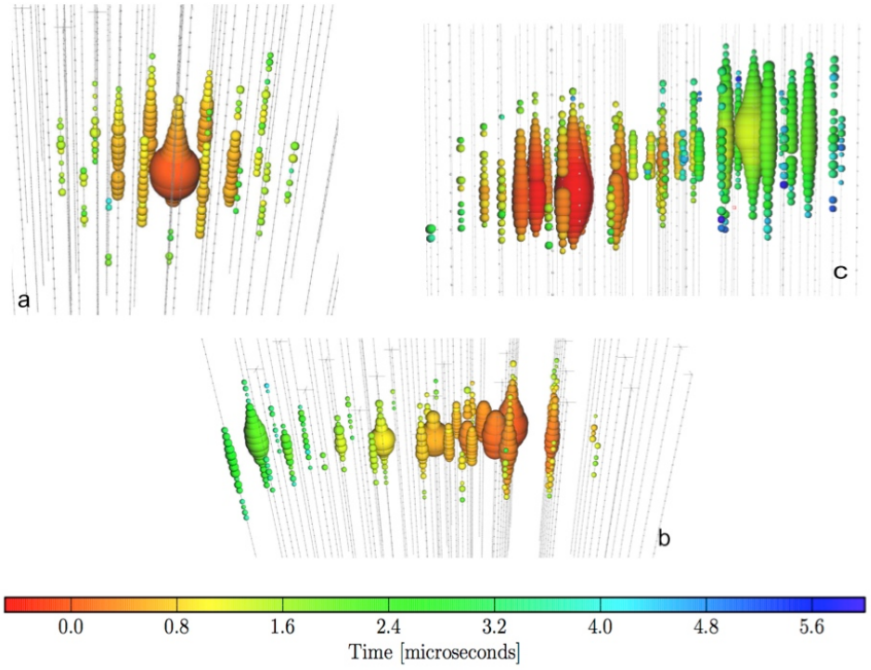



Figure 15: Different event signatures: A cascade from a 46.7 TeV $\nu_e$ event (a), a tracklike event from a 71.4 TeV $\nu_\mu$ event (b) and a simulated PeV-range double bang event (c). Figure from [40].

Hence, in general 3 different signatures are expected. *Tracklike* events from $\nu_\mu$ CC interactions, *cascade* events from $\nu_\tau$-, $\nu_e$ and NC $\nu_\mu$ (see Figure 4) interactions and *double bang* events from $\nu_\tau$ events, all shown in Figure 15. The double bang events can only occur for very high energies, where the $\tau$ does not decay immediately. Instead, the $\tau$ deposits some energy, decays, and the decay product deposits its energy further down the detector.

*To the author's knowledge, a double bang event has not yet been detected.*

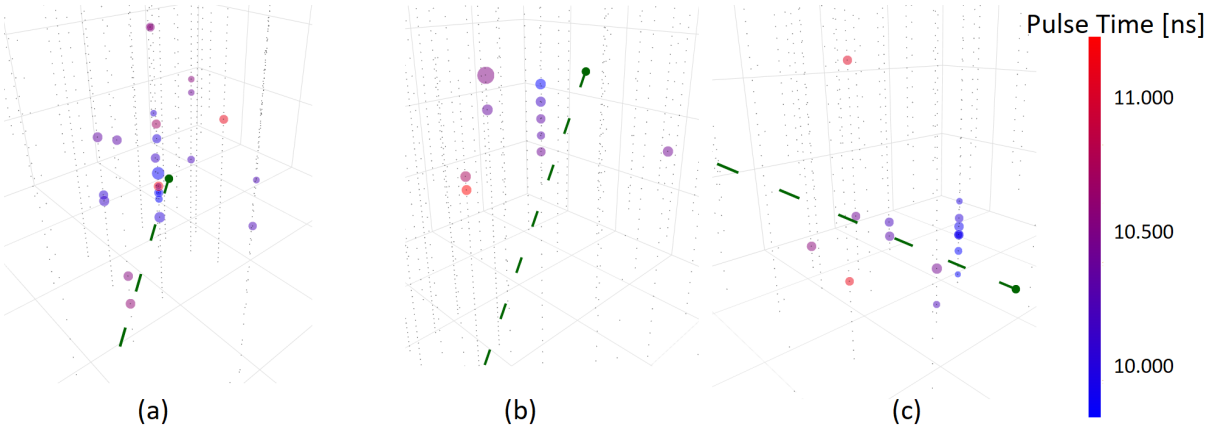



Figure 16: Simulated and SRT-cleaned $\nu_\tau$ (a), $\nu_\mu$ (b) and $\nu_e$ neutrino event at 30 GeV, all showing similar structure. Figure courtesy of Mads Ehrhorn

All events in Figure 15 are at considerably higher energies than the events analyzed in this work. These events are all in the GeV range with most being in the 10-100 GeV range of interest to oscillation analyses. Such events pose several difficulties with regards to reconstruction, since much fewer DOMs activate during $\nu_\tau$, $\nu_\mu$ and $\nu_e$ events as shown in Figure 16. All shown events are SRT-cleaned and at neutrino energies of 30 GeV, and as it can be seen, the differences between the event signatures are much more subtle (if there at all).

## 2.6 RECONSTRUCTION ALGORITHMS

For oscillation analyses, the Retro Reconstruction (Retro) is the current best reconstruction algorithm [41, 42]. It is a Maximum Likelihood Estimator (MLE) using photon propagation tables during likelihood maximization. The reconstructed values $h_{\text{track}}$ are determined as

$$h_{\text{track}} = \arg\max\left(p[t, x, y, z, l, \phi, \theta, E|\,\{\text{DOM}_1, .., \text{DOM}_N\}]\right) \qquad (28)$$

where $(x, y, z, t)$ is the spacetime interaction vertex, $(\phi, \theta)$ the direction, $l$ is the track length and $E$ is the cascade energy.

A photon propagation table is created by treating each DOM as if it emits light. Photons are then transmitted across the detector, and by counting the number of photons reaching a point $\vec{x}$ in space, the probability of detecting photons *from* $\vec{x}$ is estimated by flipping the role of the emitting and receiving ends. This procedure assumes that scattering and absorption are symmetric under time reversal.

Retro has both strengths and weaknesses. Each DOM table is produced using $100 \cdot 10^8$ photons and each table is approximately 2 GB in size. With 5160 unique DOMs, 1 TB memory is required for just storing the table values if no compression techniques are exploited.

In Figure 17 the reconstruction times for Retro are shown; these pose another disadvantage. Reconstructions can take several minutes to converge. If the quality of reconstructions is on par or better, a ML approach with reconstruction times on the order of $O(10^4)$ reconstructions per second would be a major improvement.

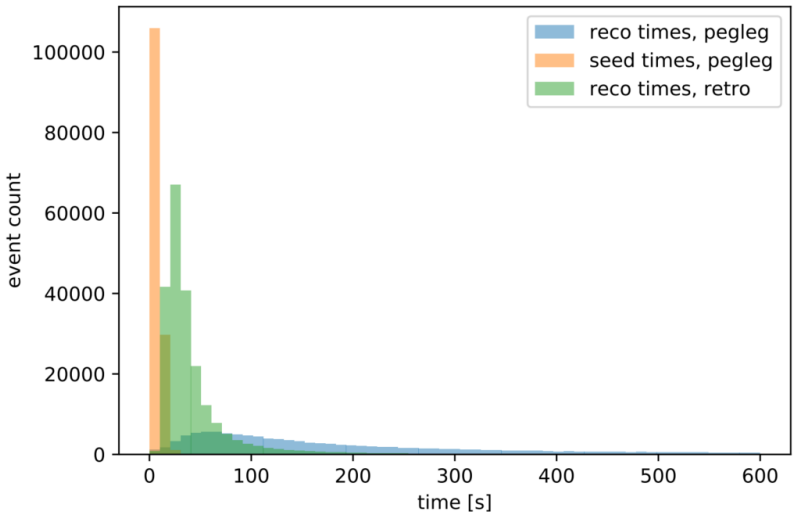



Figure 17: Reconstruction times for the Retro Reconstruction algorithm (green) and the PegLeg reconstruction algorithm (latter not used in this work). Retro reconstruction times can be seen to exceed several minutes. Figure from [41].

# 3

# DEEP LEARNING

The notorious field of artificial intelligence (AI) has given birth to chess machines of inhuman strength, self-driving cars [43] and it seems difficult to fully comprehend its limits.

Among one of the fantastic while less extravagant things AI is capable of is track reconstruction at the IceCube Neutrino Observatory using Deep Learning (DL). Technically, DL is a form of representation learning using several perceptrons to form Artificial Neural Networks (NNs) able to approximate a plethora of functions. In this chapter, the previously mentioned concepts among others will be unpacked, including DLs poorly understood theoretical grounds and how it practically is tweaked to produce meaningful results anyways.

## 3.1 ARTIFICIAL NEURAL NETWORKS

NNs come in many forms and shapes. The basic building block of all is the *neuron*. Informally, the $i'th$ neuron of a network (shown schematically in Figure 18) takes $N_{\text{in}}$ inputs, calculates a weighted sum $s_i$ (typically with a bias) and applies some nonlinear function $\sigma$ called an *activation* to it. Hence, a neuron is a function $f : \mathbb{R}^{N_{\text{in}}+1} \to \mathbb{R}$ which outputs

*The neuron or perceptron of specific types of networks may differ, but its modus operandi remains the same*

$$z_i = \sigma\left(b_i + \sum_{n=1}^{N_{\text{in}}} w_n x_n\right), \tag{29}$$

where $x_n$ are the inputs and $b_i$, $w_n$ are *learnable* parameters. The described neuron is the basic building block of *fully connected feedforward* layers (FF). To increase the expressive capacity of the layer, several neurons can be combined to produce $N_{\text{out}}$ outputs. Each neuron is described by Eq. (29), but the neurons learn different sets of weights and biases. Hence, a single-layer NN is function $l : \mathbb{R}^N_{\text{in}} \to \mathbb{R}^N_{\text{out}}$, where $N_{\text{out}}$ is referred to as the *width* of the layer.

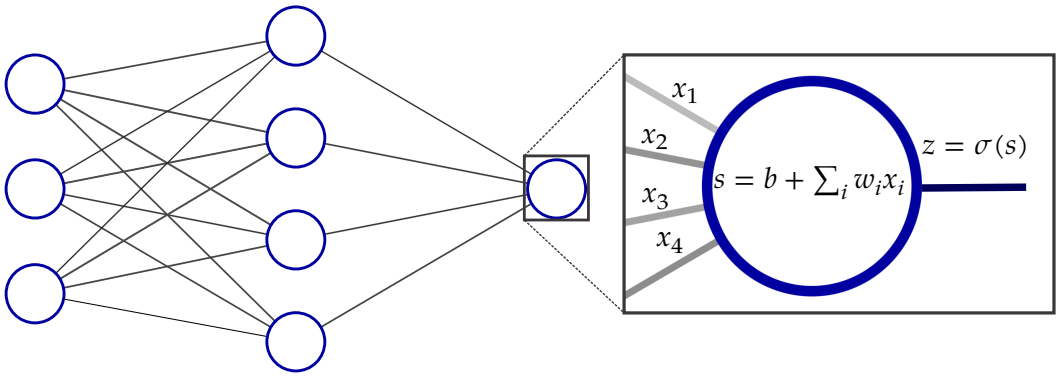



Figure 18: Schematic overview of a feedforward network with one hidden layer. Blue circles denote neurons and transparency of links between neurons denote weight size.

In practice, calculating the output $\boldsymbol{z}$ of a layer reduces to applying a nonlinear activation to a matrix product: For a layer with $i$ neurons in a NN with $j$ inputs collected into a vector $\boldsymbol{x}$

$$\boldsymbol{z} = \sigma\left(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}\right), \tag{30}$$

where $\boldsymbol{W} \in \mathbb{R}^{i \times j}$ and $\boldsymbol{b} \in \mathbb{R}^{\mathbb{i}}$ are a learnable weights and biases.

### 3.1.1 *Deep Learning*

NNs are powerful models capable of representing arbitrary functions with just a single hidden layer (and under mild assumptions on the used activation function), a result known as the Universal Approximation Theorem [44]. Once a single-layer NN is extended to include multiple layers, we move from *shallow* to *deep* networks. A NN of depth 3 as shown Figure 18. Formally, a FFNN of depth $d$ is desribed as the composition of functions

$$f(x) = (l_d \circ ... \circ l_1)(x), \tag{31}$$

where each $l_i$ is a function of the form given in Eq. (30). The advantage of increasing the amount of layers is that the representational power of the network increases faster with depth than it does with width; to represent some functions $f \in \mathcal{F}$, where $\mathcal{F}$ is a set of functions, a network $M_d^i$ of depth $d$ and width $i$ require

$$N_{\text{nodes}}(M_d^i) = O(d \cdot i) \tag{32}$$

whereas a network $M_2^i$ with 2 layers and width $i$ require

$$N_{\text{nodes}}(M_2^i) = O((i-1)^d) \tag{33}$$

to represent the same function $f$, where $N_{\text{nodes}}$ counts the number of nodes in the network [45].

### 3.1.2 *Recurrent Neural Networks*

If the FFNN is generalized to include feedback connections as well, a new family of Recurrent Neural Networks (RNNs) arise. This approach has proven to be beneficial for sequential modelling. An RNN layer is depicted in Figure 19: Each element of a sequence $\boldsymbol{x} = \{x_0, ..., x_t\}$ is input individually to the layer $A$, which applies some function $f$ and produces an output $h_i$. Along with $x_t$, the output of the previous element $h_{t-1} = f(x_{t-1})$ is fed to $A$, from which the recurrence relation is evident. Hence, the RNN can be *unfolded in time* as shown in the right part Figure 19. As shown, a RNN transforms a sequence $\boldsymbol{x}$ to a new representation $\boldsymbol{h} = \{h_0, ..., h_T\}$, effectively utilizing a network of depth $t$.

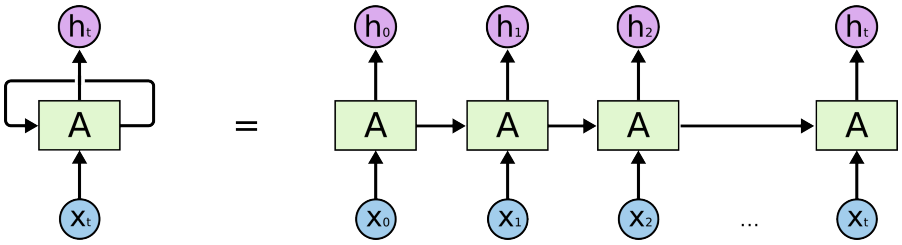



Figure 19: Illustrative drawing of a RNN and its unfolding in time. Figure from [46].

The objective of a RNN is to extract information hidden in a sequence and encode it in a new sequence (coined *seq2seq*-modelling). Typically, the last output $h_t$ is fed to FF-layers, whose objective is to decode the information. In a classic RNN, the elements of $\boldsymbol{h}$ are calculated as

$$\boldsymbol{h}_i = \tanh\left(\boldsymbol{W}_x \boldsymbol{x}_i + \boldsymbol{b}_x + \boldsymbol{W}_h \boldsymbol{h}_{i-1} + \boldsymbol{b}_h\right), \tag{34}$$

where each element of $\boldsymbol{h}$ has been represented by a vector. Since $h_t$ depends on the previous output $h_{t-1}$, a RNN must be initialized with a vector $h_0$ such that the first element of a sequence can be processed. In practice $h_0$ is typically initialized to be a random vector or a vector containing all o's.

In practice, obtaining satisfactory convergence of RNNs of the form given by Eq. (34) has proven to be difficult for long sequences[47]; therefore, several variations of the classic RNN neuron has been created. In this work, Long Short Term Memory networks (LSTMs) [48] and Gated Recurrent Unit networks (GRUs) [49] are employed. Both LSTMs

*Several researchers and practitioners have contributed to the modern version of LSTMs. Attempting to mention them all would be futile.*

and GRUs work in the same manner as a classic RNN, but the neurons are different. Formally, an LSTM layer applies the equations

$$\begin{aligned} i_t &= \sigma\left(\boldsymbol{W}_{ix}\boldsymbol{x}_t + \boldsymbol{b}_{ix} + \boldsymbol{W}_{ih}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{ih}\right), \\ f_t &= \sigma\left(\boldsymbol{W}_{fx}\boldsymbol{x}_t + \boldsymbol{b}_{fx} + \boldsymbol{W}_{fh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{fh}\right), \\ g_t &= \tanh\left(\boldsymbol{W}_{gx}\boldsymbol{x}_t + \boldsymbol{b}_{gx} + \boldsymbol{W}_{gh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{gh}\right), \\ o_t &= \sigma\left(\boldsymbol{W}_{ox}\boldsymbol{x}_t + \boldsymbol{b}_{ox} + \boldsymbol{W}_{oh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{oh}\right), \\ c_t &= f_t * c_{t-1} + i_t * g_t, \\ h_t &= o_t * \tanh(c_t), \end{aligned} \tag{35}$$

and the GRU layer applies

$$\begin{aligned} r_t &= \sigma\left(\boldsymbol{W}_{rx}\boldsymbol{x}_t + \boldsymbol{b}_{rx} + \boldsymbol{W}_{rh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{rh}\right), \\ z_t &= \sigma\left(\boldsymbol{W}_{zx}\boldsymbol{x}_t + \boldsymbol{b}_{zx} + \boldsymbol{W}_{zh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{zh}\right), \\ n_t &= \tanh\left(\boldsymbol{W}_{nx}\boldsymbol{x}_t + \boldsymbol{b}_{nx} + r_t * \left(\boldsymbol{W}_{nh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{gh}\right)\right), \\ h_t &= (1 - z_t) * n_t + z_t * h_{t-1}, \end{aligned} \tag{36}$$

where all matrices $\boldsymbol{W}$ and biases $\boldsymbol{b}$ are learnable, $*$ denotes elementwise multiplication and $\sigma = \frac{1}{1+\exp(-x)}$ is the sigmoid function.The information flow in LSTM (left) and GRU (right) layers is shown in Figure 20. Informally, the LSTM layer consists of an internal state $c_t$ and 3 gates controlling what is remembered and forgotten: An input gate $f_t$, a 'commit to memory'-gate $i_t$ and an output gate $o_t$, where the sigmoid functions let important information through (when $x > 0$) and suppress irrelevant information (when $x < 0$). A GRU layer shows resemblance to the LSTM layer and has fewer parameters, but in return it has no internal state.

*Elementwise multiplication is also known as the Hadamard product.*

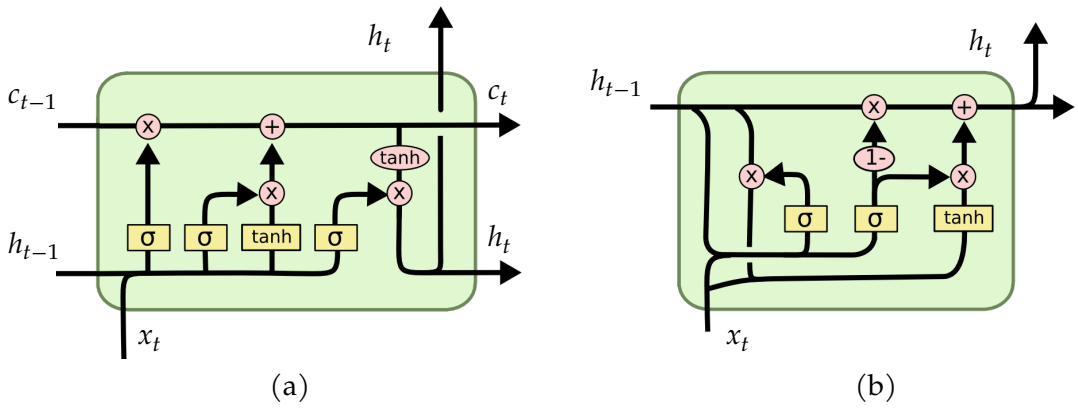



Figure 20: (a): Schematic overview of information flow in a LSTM layer. Figure from [46]. (b): Schematic overview of information flow in a GRU layer. Figure from [46].

Many more variations on LSTMs have been tested and created. Studies have however shown that most variants do equally well [50]. Both LSTMs and GRUs are used in this work.

### 3.1.3 *Attention Networks*

A different approach to sequence processing is an attention mechanism. Instead of having the processing of the element $h_t$ depend on only the previous elements, an attention mechanism can be utilized which allows the network to extract information from every element of the sequence during the processing of $h_t$. Networks based on such a mechanism have achieved state-of-the-art results in the field of natural language processing [51].

Attention works as follows: For each sequence element $\boldsymbol{x}_i$ represented by a vector, a query vector $\boldsymbol{q}_i$, key vector $\boldsymbol{k}_i$ and value vector $\boldsymbol{v}_i$ each with $d$ elements are calculated as

$$\begin{aligned} \boldsymbol{q}_i &= \boldsymbol{A}_q \boldsymbol{x}_i, \\ \boldsymbol{k}_i &= \boldsymbol{A}_k \boldsymbol{x}_i, \\ \boldsymbol{v}_i &= \boldsymbol{A}_v \boldsymbol{x}_i, \end{aligned} \tag{37}$$

where $\boldsymbol{A}_q$, $\boldsymbol{A}_k$ and $\boldsymbol{A}_v$ are learnable matrices. A new representation of the input is then calculated as

$$\boldsymbol{h}_i = \sum_{j=1}^{L} \text{softmax}\left( \frac{\boldsymbol{q}_j \cdot \boldsymbol{k}_i}{\sqrt{d}} \right)_j \boldsymbol{v}_j, \tag{38}$$

where the softmax function

$$\text{softmax}(x_j) = \frac{\exp(x_j)}{\sum_{i=1}^{N} \exp(x_i)} \tag{39}$$

converts the *j attention scores* (argument to the softmax) to a probability distribution. Since Eq. (38) is nothing but a linear function of the inputs, the outputs are individually sent through atleast 1 FF layer to introduce nonlinearity. Attention will be compared to the performance of and used in conjunction with RNNs during the search for a good reconstruction model.

*This specific kind of attention is also called dot-product attention.*

## 3.2 STATISTICAL LEARNING THEORY

An archetypical ML problem can be stated as: Given a set $S = \{(x_1, y_1), .., (x_N, y_N)\}$ of size $N$, where $x_i \in \mathcal{X}$ is the feature or features of the *i'th* observation and $y_i \in \mathcal{Y}$ is the *i'th* label, find a model (or hypothesis) $h$ capable of as accurately as possibe produce labels for unseen datapoints given their features $x$. The described problem is a *supervised learning* problem as opposed to an unsupervised learning problem, in which a training set $S$ without labels is given, and the objective of the model is to find patterns in data.

### 3.2.1 *Supervised Learning*

A supervised learning problem can either be a *classification* problem or a *regression* problem. In the former case, the labels $y$ are categorical variables and in the latter they are continuous. In the work presented, both classification and regression problems will be tackled.

Formally, given a loss function $l$, which measures the deviation from the true label, the objective of the learner is to find the *hypothesis* $h : \mathcal{X} \mapsto \mathcal{Y}$, $h \in \mathcal{H}$, where $\mathcal{H}$ is the set of all possible hypotheses given the constraints of the chosen hypothesis class (in this work neural networks), minimizing the expected loss

$$L(h) = \mathbb{E}\left[l(h(x), y)\right]. \tag{40}$$

Typically only an estimate of $L$ can be obtained and the empirical loss

$$\hat{L}(h) = \frac{1}{N}\sum_{i=0}^{N} l(h(x_i), y_i) \tag{41}$$

is minimized instead, where all pairs $(x_i, y_i)$ are assumed to be i.i.d. according to some distribution $p(x, y)$. Hence, the optimal hypothesis, we must find through *training*, is defined as

$$h^* = \arg\min_{h \in \mathcal{H}} \hat{L}(h). \tag{42}$$

### 3.2.2 *Bounding the Generalization Error*

The central requirement of a model is to produce accurate predictions on *unseen* data - we want the model to generalize well. Once a model has been optimized through training on $\mathcal{S}$, the hypotheses produced are optimized to reduce the error *on that set*. The hypotheses are functions of $\mathcal{S}$ and there is no guarantee that the error on unseen data will be anywhere near the training error. In other words, $\hat{L}(h^*_{\mathcal{S}})$ is a biased estimate of $L(h^*)$, where $h^*_{\mathcal{S}}$ denotes evaluation of the optimal hypothesis with respect to the set $\mathcal{S}$.

In some cases it is however possible to produce a bound on the difference

$$G(h) = \hat{L}(h) - L(h) \tag{43}$$

called the *generalization error*. Specifically, a bound decaying exponentially fast with the amount of data can be derived for many supervised learning problems using the two-sided Hoeffding's Inequality [52], which is given here without proof: If we let $X_1, .., X_n$ be i.i.d. real random variables, where $X_i \in [a, b]$, then for every $\epsilon > 0$:

*Some sources defines the generalization error simply as $L(h)$.*

$$P\left(\left|\sum_{i=1}^{n} X_i - \mathbb{E}\left[\sum_{i=1}^{n} X_i\right]\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{2}{n}\left(\frac{\epsilon}{b-a}\right)^2\right). \tag{44}$$

If we let $|\mathcal{H}|$ be the amount of different hypotheses in $\mathcal{H}$ (i.e. the amount of different ways a model is capable of labelling a training set), then it can be shown using Hoeffding's inequality that

$$P\left(|G(h^*)| \geq (b-a)\sqrt{\frac{\log\frac{2|\mathcal{H}|}{\delta}}{2n}}\right) \leq \delta, \tag{45}$$

which displays the bias-variance tradeoff: The more complex a model is made (expressed by the size of the hypothesis class), the less certain it is that a model generalizes well.

It turns out that NNs are immensely difficult to bound. To bound NNs used for regression, the measure of the complexity of the hypothesis class (here its size) has to be altered, since its size is uncountably infinite. Furthermore, the bounds tend to either be so loose that they become useless or the requirements of $\mathcal{H}$ are too restrictive to have any practical use. In other words, even though NNs are *capable* of representing arbitrary functions, it is theoretically poorly understood how these representations can be learned.

## 3.3 TRAINING NETWORKS

Despite the theoretical shortcomings mentioned above, practitioners are capable of training NNs with marvelous capabilities - inside as well as outside the laboratory. The models are trained using *backpropagation* [53] together with Stochastic Gradient Descent (SGD) or one of its variants for a number of *epochs* (one loop over the entire traning) until the model converges.

### 3.3.1 *Optimizers*

In general, there exists no closed-form solution to Eq. (42) for NNs, and therefore alternative methods must be utilized to optimize DL models. SGD is an iterative minimization process, where the weights of a network at each step are updated according to

$$w_{t+1} = w_t - \eta\nabla_w\hat{L}(w_t),\ \eta > 0, \tag{46}$$

where $\eta$ is the learning rate and $\hat{L}$ is the empirical loss defined in Eq. (5.4). The stochasticity stems from only using a subset of the training set in estimating the gradient, where the number of datapoints used is referred to as the *batchsize* (BS). Prior to training initiation, the weights of the network are initialized to random values drawn from some distribution.

*Outside of ML, $\eta$ is also known as the stepsize.*

The loss landscapes enountered in DL as a function of the weights can be highly nonconvex, which makes optimization difficult. Such a landscape is shown in Figure 21; by adding noise through small batchsizes, sharp, local minima can be escaped leading to better generalization, i.e. the noisy estimates of the gradient are beneficial to the optimization. [54, 55].

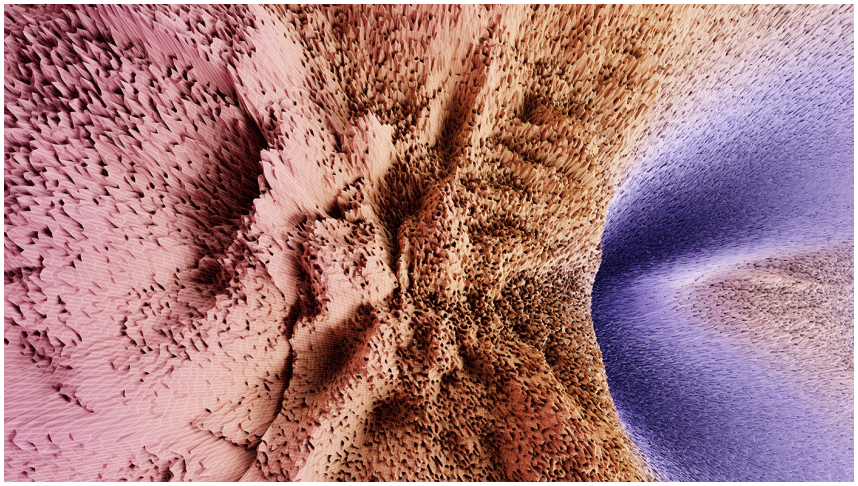


Figure 21: A highly nonconvex loss landscape sampled during optimization of a NN, where height and color indicates the loss value (blue is small). The landscape was created by varying 2 of the network's weights on a 2-dimensional grid. Figure from [56].

Experiments with 3 different optimization algorithms are performed in this work: SGD with Momentum (SGDM) [57], Nesterov Accelerated Gradient (NAG) [58] and the Adam optimizer [59]. SGDM includes a bit $\gamma > 0$ of the gradient at the previous step. Mathematically, Eq. (46) is altered to

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_w \hat{L}(w_t), \\ w_{t+1} &= w_t - v_t. \end{aligned} \tag{47}$$

Effectively, the added momentum increases the rate of change along dimensions, where the gradients at different timesteps tend to point in the same direction and reduces the rate of change along dimensions, where the gradients oscillate between negative and positive values at different timesteps. NAG given by

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_w \hat{L}(w_t - \gamma v_{t-1}), \\ w_{t+1} &= w_t - v_t \end{aligned} \tag{48}$$

is very similar to SGDM, but with one subtle difference as shown in Figure 22: the gradient is evaluated at a point *looking ahead* instead of at the current position.

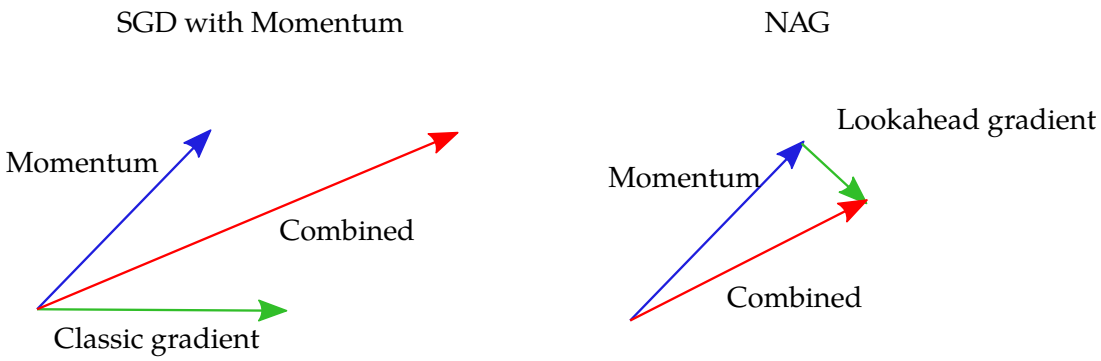



Figure 22: Comparison of Stochastic Gradient Descent with Momentum (left) and Nesterov Accelerated Gradient (right). Contributions to the final step are shown in different colours.

Finally, Adam is a more radical modification of the classic SGD. At each step, the weights are updated using

*Adam is an abbreviation for 'Adaptive moment estimation'*

$$g_t = \nabla_w \hat{L}(w_t) \tag{49}$$
$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{50}$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{51}$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{52}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{53}$$
$$w_{t+1} = w_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}. \tag{54}$$

Adam approximates the first and second moments of the gradient via an exponential moving average to produce an update rule. The brilliance of Adam comes from Eq. (54): The learning rate of each individual weight is adaptive.

SGDM, NAG and Adam all require the specification of hyperparameters with Adam requiring the most. Typical values used for some are $\gamma = 0.9$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and $\eta$ requires problem-specific tuning.

### 3.3.2 *Backpropagation*

All described optimizers require the evaluation of the gradient of the loss function, which can be quite costly; some NNs comprise billions of weights. But by updating weights one layer at a time, optimization becomes manageable. By applying the chain rule to Eq. (31), the derivative w.r.t. the *n'th* weight in the *i'th* layer becomes

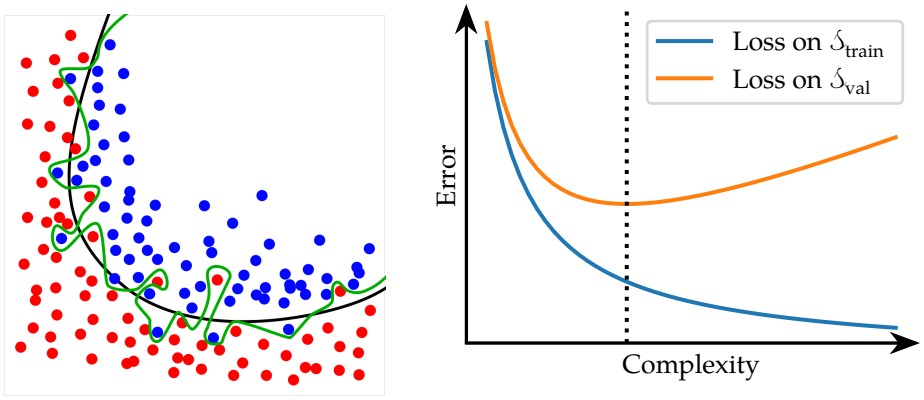
$$\frac{\partial \hat{L}}{\partial w_n} = \frac{\partial \hat{L}}{\partial l_d} \cdot \frac{\partial l_d}{\partial l_{d-1}} \cdot \ldots \cdot \frac{\partial l_i}{\partial w_n}, \tag{55}$$

where all partial derivatives can be evaluated one layer at a time. By updating the deepest layers first and saving their gradients temporarily, the errors can be propagated backwards through the layers with a minimum of computation and the weights can be updated.

### 3.3.3 *Validation and Test sets*

A remedy to the problem of estimating $L(h^*)$ is to split $\mathcal{S}$ into a *training set* $\mathcal{S}_{\text{train}}$, a *validation set* $\mathcal{S}_{\text{val}}$ and a *test set* $\mathcal{S}_{\text{test}}$. Several models can then be trained on $\mathcal{S}_{\text{train}}$ and for each model $\hat{L}(h^*_{\mathcal{S}_{\text{val}}})$ can be evaluated, where each model individually produces unbiased estimates of $L(h^*)$ since $\mathcal{S}_{\text{val}}$ is unseen data. The model achieving the lowest error on $\mathcal{S}_{\text{val}}$ can then be chosen as the final model $f_{\text{best}}$. But since $f_{\text{best}}$ was chosen to specifically be the best performing model on $\mathcal{S}_{\text{val}}$, the associated $\hat{L}(h^*_{\mathcal{S}_{\text{val}}})$ is again a biased estimate of $L(h^*)$. Hence, $\mathcal{S}_{\text{test}}$ is reserved to produce a final, unbiased estimate of the model's performance $\hat{L}(h^*_{\mathcal{S}_{\text{test}}})$.

The application of a validation set comes with an additional asset: Early stopping. Due to the powerful nature of NNs described in Section 3.1.1, they are prone to overfitting. This means that as the complexity of the model increases during training, it begins to fit the inherent noise of $\mathcal{S}_{\text{train}}$ illustratiely shown in Figure 23a, which in return leads to poor generalization. By monitoring $\hat{L}(h_{\mathcal{S}_{\text{val}}})$ during training, the point at which overfitting begins to emerge can be determined as shown in Figure 23b (denoted by the black, dotted line) and the training can be stopped. Early stopping is one form of *regularization* (a constraint meant to reduce the risk of overfitting) utilized in this work. Additionally, early stopping allows the use of deeper and more complex NNs, since overfitting becomes less of an issue.




(a) (b)

Figure 23: (a): An illustration of an overfitting classification model (green) and an optimal model (black). The green model manages to capture the idiosyncrasies of the data leading to poor generalization. Figure from [60]. (b): Illustration of how the emergence of overfitting during training. As the complexity of the model increases, the error on the training set $\mathcal{S}_{\text{train}}$ (blue) continues to decrease, while the error on an unseen validation set $\mathcal{S}_{\text{val}}$ (orange) eventually begins to increase. By stopping the training at the point indicated by the black, dotted line, overfitting can be avoided.

There is currently no theoretically correct way of splitting $\mathcal{S}$ into $\mathcal{S}_{\text{train}}$, $\mathcal{S}_{\text{val}}$ and $\mathcal{S}_{\text{test}}$. Increasing or decreasing the sizes of each set comes with advantages and disadvantages. For instance, increasing the size of $\mathcal{S}_{\text{train}}$ will expose the model to a better estimate of the true underlying distribution $p(x, y)$ likely increasing the model's performance, but in return the estimate of the generalization error will be less precise.

### 3.3.4 *Normalization*

Before data is fed to a NN, it is beneficial to standardize it (rescale to zero mean and unit variance) causing convergence to happen faster and to a lower loss. Furthermore, rescaling the outputs of the *i'th* layer before feeding it to the $(i+1)$*'th* layer has proven beneficial as well; a method known as *batch normalization* (BN) [61]. Formally, the input $x_{i+1}$ to the $(i+1)$*'th* layer is calculated as

$$x_{i+1} = \gamma_i \cdot \frac{z_i - \hat{z}_i}{\sqrt{\text{Var}(z_i) + \epsilon}} + \beta_i, \tag{56}$$

where $\gamma_i$ and $\beta_i$ are learnable parameters initialized to 1 and 0 respectively, $\epsilon$ is a small number to avoid division with 0 and the empirical mean $\hat{z}_i$ and variance $\text{Var}(z_i)$ are over the batch.

Intuitively, the success of BN can be understood by examining the distribution of a layer's output during training: As the weights of the

*i'th* layer changes, its output changes as well. This causes difficulties for the $(i+1)$*'th* layer, since its input distribution can change rapidly as training progresses. By standardizing the input data, a layer becomes more robust to such changes.

*The change in the input distribution is also known as 'internal covariate shift'.*

It is not obvious how to apply BN to RNNs, where inputs are sequences. Instead *layer normalization* (LN) can be applied to every single entry in the batch [62]: If the input is a sequence of vectors, Eq. (56) is applied to the *i'th* entry of each vector, where the mean and variance are calculated over the vectors in the sequence.

All mentioned forms of rescaling are utilizied in this work.

# 4 NEUTRINO DATA

Cutting to the bone, there are three questions that an Icecube neutrino reconstructor is interested in delivering high-quality answers to: What kind of neutrino was it? Where did it come from? And how energetic was it? This part of the work presents the development and results of applying RNNs to simulated neutrino data to answer these questions. First, the used datasets along with the applied preprocessing is described. Then the development of the models is set out before moving on to how well the models are performing .

## 4.1 SIMULATION OF NEUTRINOS

Understanding the generation of data is important, since a NN is never better than the dataset it has been trained on. For a model to generalize well, the data it is trained on must resemble the unseen data. The neutrino events used for training in this work has been created using extensive Monte Carlo simulations with several components. Roughly speaking, the simulation consists of

- generation and propagation of neutrinos,
- photon propagation,
- DOM photo response and
- DOM noise simulation.

The generation and propagation of neutrinos (including their interaction with the ice) is done using the GENIE simulation framework [63], which is a state-of-the-art neutrino simulator widely used in the experimental neutrino community. Photon propagation in the ice is carried out by Icecube's CLSim [64] and noise simulation is a combination of a Poissonian component, which is simulated by sampling from a Poisson distribution, and a non-Poissonian component simulated by Icecube's Vuvuzela module [41]. The downward-facing PMTs in the DOMs are modelled by introducing an *acceptance probability* of photons as a function of photon direction, which decreases with the zenith angle.

*The more upgoing a photon is, the more likely it is to be detected.*

Each step of the simulation process mentioned above might introduce simulation artefacts causing a discrepancy between reality and simulation. For instance, the simulation of absorption and scattering of photons in the ice requires the specification of a simplified ice model. In reality the optical properties of the South Pole ice are highly dependent on position as indicated by Figure 24: The optical properties vary both with depth and position in the xy-plane.

The discussion above serves to stress the fact that despite how well a NN performs on simulated data, some way of verifying the performance on real data is desirable.

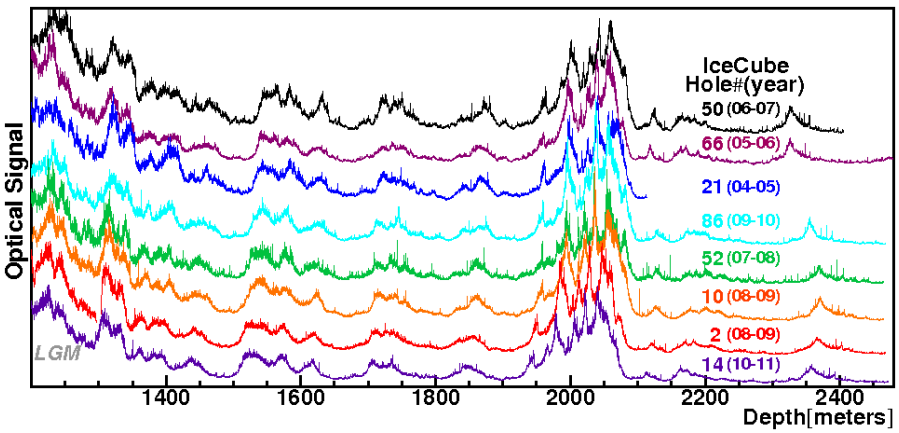



Figure 24: Optical response as a function of depth for various boreholes in the Icecube detector. Data from different boreholes has been offset along the y-axis. Higher values indicates less clear ice (= more scattering). For all holes, the dustlayer at a depth of approximately 2 km is visible. Figure from [65].

## 4.2 EVENT SELECTION

The vast majority of particles detected by Icecube are not interesting for most analyses: The signal-to-noise ratio across the whole detector is approximately $10^{-6}$ [28]. Therefore a dramatic removal of background, which primarily consists of cosmic muons, is necessary. In this work, simulated events chosen with the OscNext-scheme has been used [41]. OscNext uses a filtering process with 7 *selection levels* of which the important level 2 and level 5 are described in depth below. Level 1 are simply events activating the SMT3-trigger.

*Before cuts, Icecube roughly detects events at a rate of 2500 Hz.*

### 4.2.1 *Level 2 - DeepCore filter*

At level 2, a major selection coined the DeepCore filter is applied, which reduces the background cosmic muon rate from 280 Hz to approximately 17 Hz [28]. It proceeds with the following steps:

- Apply SRT cleaning.
- Calculate mean pulse time $\bar{t}$.
- Remove all pulses further away from $\bar{t}$ than one standard deviation.

- Calculate spatial $\vec{x}_{\text{CoG}}$ and temporal $t_{\text{CoG}}$ center of gravity given by

$$\vec{x}_{\text{CoG}} = \frac{1}{N}\sum_{i=0}^{N}\vec{x}_i \tag{57}$$

and

$$t_{\text{CoG}} = \frac{1}{N}\sum_{i=0}^{N} t_i - \frac{|\vec{x}_i - \vec{x}_{\text{CoG}}|}{c/n_{\text{ice}}} \tag{58}$$

- If any pulse speed $v_{\text{pulse}}$ of hits in the *veto region* (defined as DOMs outside DeepCore) is within the interval

$$0.25\,\frac{\text{m}}{\text{ns}} \leq v_{\text{pulse}} \leq 0.40\,\frac{\text{m}}{\text{ns}}, \tag{59}$$

where

$$v_{\text{pulse}} = \frac{|\vec{x}_{\text{CoG}} - \vec{x}_i|}{t_{\text{CoG}} - t_i} \tag{60}$$

discard the event.

The reasoning behind the selection is the fact that if the pulse speed from outside DeepCore to the center of gravity is approximately $c$, it likely corresponds to a relativistic cosmic muon. Hence a sharp peak around $v = c$ can be seen in Figure 25 along with the contributions from background and signal being cut away at level 2.

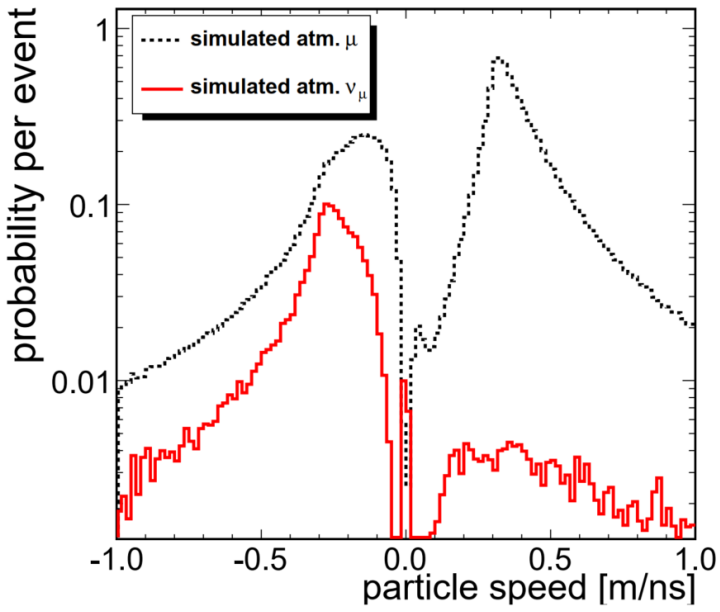



Figure 25: Probability of being background (black dashed line) or signal (red solid line) as a function of signal speed for simulated muons and muon neutrinos. A sharp peak about $v = c$ due to relativistic cosmic muons can be seen. Figure from [28].

### 4.2.2 *Level 3, 4 and 5*

At level 3, various cuts on variables are made to ensure better agreement between MC-generated events and real data and Level 4 is a selection based on Boosted Decision Trees (BDTs). The level 5 selection is designed to remove sneaky muons escaping the detection of the previous levels: Due to the quasi-hexagonal structure of the detector, there are corridors where muons can enter DeepCore. The Corridor Cut Module attempts to exactly remove these events by examining DOMs in known poorly instrumented directions from the Center of Gravity $\vec{x}_{\text{CoG}}$ [41].

At level 5, the total neutrino rate is approximately 2 mHz and the background has been brought to the sub 10 mHz range. Especially understanding the level 2 selection is important, since some signal events are inevitably discarded at this level. A DL approach to background rejection might be able to optimize the amount of signal being kept. There might be potential for optimizing the level 5 selection using DL as well, but with diminishing returns; it is at level 2 the largest selection is made.

## 4.3 OSCNEXT LEVEL 5 DATA

The dataset (referred to as the OscNext-set) used in this work is part of the OscNext V1 analysis [41], and it consists of simulated $\nu_e$-, $\nu_\mu$- and $\nu_\tau$-events. All events are Deepcore-events and have made it to lvl 5 in the event selection described in Section 4.2. The energy distributions of the neutrinos are shown in Figure 26: The dataset contains $11.3 \cdot 10^6$ neutrino events with energies ranging from 1 GeV to 10 TeV

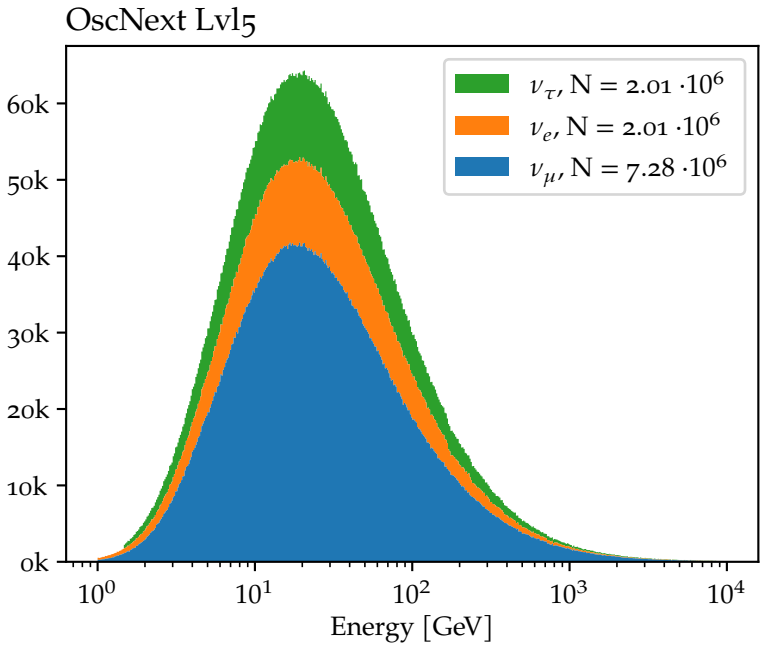



Figure 26: Neutrino energy distributions of the OscNext datasetat Lvl5.

The majority of all neutrino events have energies between 10 and 50 GeV with very few neutrinos with energies > 1 TeV. Since most events are $\nu_\mu$-events and the 10-50 GeV energy range is the range of biggest interest with respect to oscillation analysis, the developed models have been focused on muon-neutrinos below 1 TeV.

*When it is stated that a neutrino has an energy of $x$, it means that in the restframe of the detector at the time of interaction, the neutrinos energy was $x$.*

The distribution of the directions neutrinos are coming *from* is shown in Figure 27, where the directions have been mapped to points on an unfolded unit sphere. The directions of the events are not uniform on the unitsphere, but tend to have an upwards direction. The distributions of the $x$, $y$ and $z$-coordinates of the directional unitvectors are shown in Section A.1.

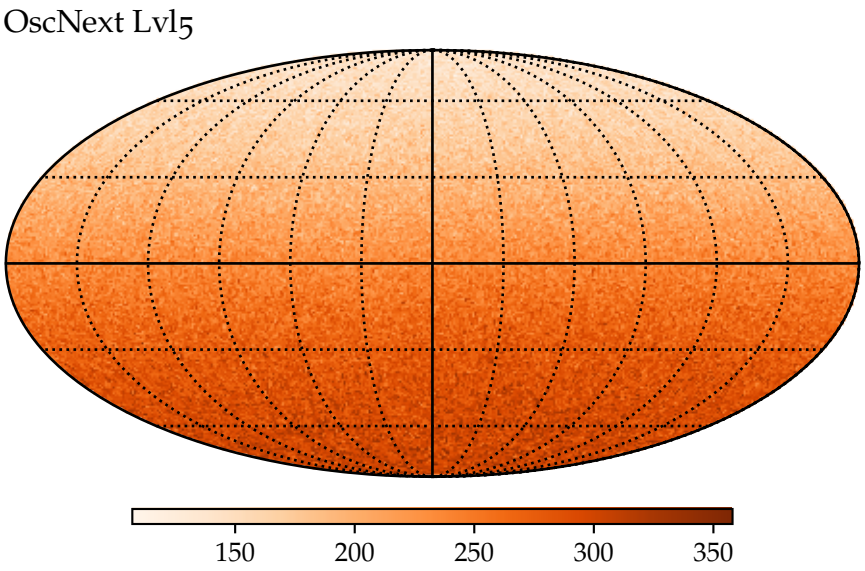



Figure 27: Distribution of direction neutrinos are coming from of the OscNext Lvl5 dataset shown on an unfolded unit sphere.

A single event consists of a sequential and a scalar part. The sequential part is the actual event, where each entry in the sequence is data related to a single DOM. A raw sequence entry consists of 6 numbers:

1. `DOM_x`: The x-coordinate of the DOM.
2. `DOM_y`: The y-coordinate of the DOM.
3. `DOM_z`: The z-coordinate of the DOM.
4. `DOM_q`: The charge extracted from the raw waveform.
5. `DOM_t`: The time (in ns) w.r.t. the triggertime at which the pulse was detected.
6. `DOM_ATWD`: A Boolean. 1 if a ATWD-digitizer recorded the waveform or 0 if a fADC-digitizer recorded the waveform.

Accordingly, an event with $L$ DOMs and $N$ features per DOM (such as `DOM_x`) can be represented by a $N \times L$ matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{v}_1 & \dots & \boldsymbol{v}_L \end{bmatrix} \tag{61}$$

with the features $v_i$ of the *i'th* DOM as columns.

The scalar part consists of the event truths and the Retro scheme's reconstruction. Both truth and reconstructions consist of

- interaction vertex (x, y, z and time),
- a direction parametrized by a unit vector and
- the neutrino energy.

The sequence lengths vary from $> 600$ to $< 20$ and are strongly dependent on cleaning. The majority of sequences have lengths between 10 and 25 for SRT-cleaned events and between 25 and 75 for uncleaned events as shown in Figure 28. On the order of 1 % of events have sequences longer than 200. These events are discarded to reduce training time.

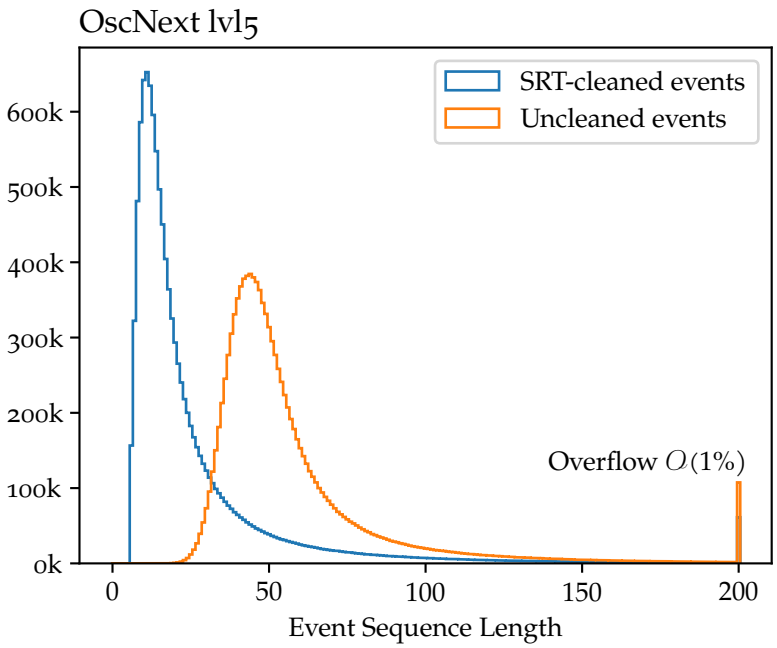



Figure 28: Sequence length distribution for SRT-cleaned (blue) and uncleaned (orange) events. Last bin is an overflow bin.

For all developed models, the OscNext-set was split into a training set consisting of 80% of selected data, a validation set consisting of 10% of selected data and a test set consisting of the remaining 10 % of the selected data. The size and splitting of the dataset before and after cuts on event sequence length and energy is summarized in Table 2.

| | All Data | | | After selection | | |
|---|---|---|---|---|---|---|
| | Train | Val. | Test | Train | Val. | Test |
| $\nu_e$ | 1.61 M | 0.20 M | 0.20 M | 1.59 M | 0.19 M | 0.19 M |
| $\nu_\mu$ | 5.82 M | 0.73 M | 0.73 M | 5.75 M | 0.72 M | 0.72 M |
| $\nu_\tau$ | 1.61 M | 0.20 M | 0.20 M | 1.59 M | 0.19 M | 0.19 M |

Table 2: Summary of the OscNext V1 dataset.

## 4.4 PREPROCESSING

As described in Section 3.3.4, NNs benefit from standardized inputs. In this work, different normalizations are applied to most input and target

variables. Three different normalization schemes have been used in this work: One affine transformation and two nonlinear transformations for robustness.

Both sequential and scalar variables whose distributions have adequately small tails have been scaled by their interquartile range (IQR) and centered around their median, i.e. the transformation

*The interquartile range is defined as the difference between the 75'th and 25'th percentiles.*

$$y_i = \frac{x_i - \text{median}(x)}{\text{IQR}(x)} \tag{62}$$

has been applied, where the median and IQR have been estimated from the sample. This transformation has the advantage of being more robust to outliers than a classic standardization. The nonlinear transformations have been applied to variables with long tails such as the neutrino energy $E$, which has been transformed according to

$$E_{\text{i, transformed}} = \frac{\log E_i - \text{median}(\log E)}{\text{IQR}(\log E)}. \tag{63}$$

The reasoning behind applying this transformation to energy specifically will be discussed later. The last transformation used is

$$y_i = \Phi^{-1}\left[F_x(x_i)\right], \tag{64}$$

where $\Phi^{-1}$ is the inverse of the cumulative distribution function for the unit normal distribution and $F_x$ is the empirical distribution function of the variable $x$. Such a transformation is shown in Figure 29, where the distribution of `DOM_charge` from 20.000 events before and after transformation is shown. Note that the distribution of `DOM_charge` is discrete due to digitization. In words, Eq. (64) maps the *i'th* quantile of the empirical distribution of $x$ to the *i'th* quantile of a unit normal distribution.

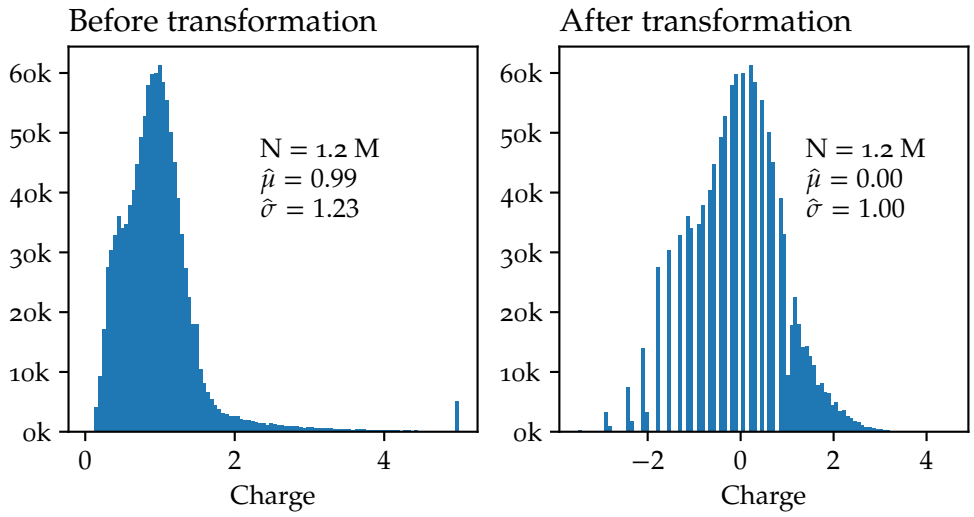



Figure 29: Distribution of `DOM_charge` before (left) and after (right) applying the nonlinear transformation given by Eq. (64). The visibly discrete nature of the distribution after the transformation is due to digitization.

| | Transformation | $\mu$, before | $\sigma$, before | $\mu$, after | $\sigma$, after |
|---|---|---|---|---|---|
| `DOM_charge` | ToNormal | 9.89e-01 | 1.04e+00 | 0.00 | 1.00 |
| `DOM_x` | Robust | 1.86e+01 | 2.12e+02 | 0.15 | 0.61 |
| `DOM_y` | Robust | -1.66e+01 | 2.02e+02 | 0.05 | 0.58 |
| `DOM_z` | Robust | -1.72e+02 | 2.84e+02 | -0.39 | 0.81 |
| `DOM_time` | Robust | 1.09e+04 | 2.42e+03 | 0.24 | 1.14 |
| `DOM_ATWD` | None | 0.32 | 0.47 | - | - |
| True energy | LogRobust | 8.04e+01 | 3.64e+02 | 0.06 | 0.75 |
| True vertex, x | Robust | 4.37e+01 | 8.41e+01 | -0.02 | 0.84 |
| True vertex, y | Robust | -3.13e+01 | 7.86e+01 | 0.07 | 0.72 |
| True vertex, z | Robust | -3.74e+02 | 1.14e+02 | -0.01 | 0.74 |
| True time | ToNormal | 9.66e+03 | 2.63e+02 | -0.04 | 0.99 |
| True direction, x | None | -0.04 | 0.58 | - | - |
| True direction, y | None | -0.00 | 0.58 | - | - |
| True direction, z | None | 0.14 | 0.56 | - | - |

Table 3: Summary of single-event variables and the transformations applied. Robust refers to Eq. (62), LogRobust refers to Eq. (63) and ToNormal refers to Eq. (64). The true vertex variables are the coordinates of the interaction vertex in the detector, true time is the interaction time and the true direction variables are the coordinates of the unit vector defining the neutrino direction. The before and after columns refer to the means and standard deviations before and after transformations have been applied.

The different transformations applied to the variables and targets of the OscNext-set along with statistical key numbers are summarized in Table 3 and figures of the DOM $x-, y-, z-$ and $t-$ and interaction vertex $x-, y-, z-$ and $t-$ distributions are shown in Section A.1.

# 5 NEURAL NETWORK CONSTRUCTION

Determining the method of optimal reconstruction is a difficult task. As mentioned in Section 2.6, the current best method of reconstruction is a slow, table-based likelihood fit. In this chapter the development of NNs competitive with respect to both reconstruction time and quality is described.

## 5.1 NETWORK BUILDING BLOCKS

All developed models can at the most general level be decomposed into an encoder and a decoder: The encoder takes a sequence of vectors as input and generates a new representation of the sequence. The encoding can be a multistage process, where the optimal architecture has to be searched for. The decoder takes the representation generated by the encoder as input and outputs estimates of the desired target variables, schematically shown in Figure 30. As a first step in the decoding process, the decoder takes a sequence of non-fixed length and performs some form of a many-to-one operation, where the sequence is converted to a fixed set of features. From the fixed set of features, a prediction is then generated, e.g. a reconstructed energy, a classification or all 8 target variables at once. We will now delve into the different building blocks used for encoding and decoding.

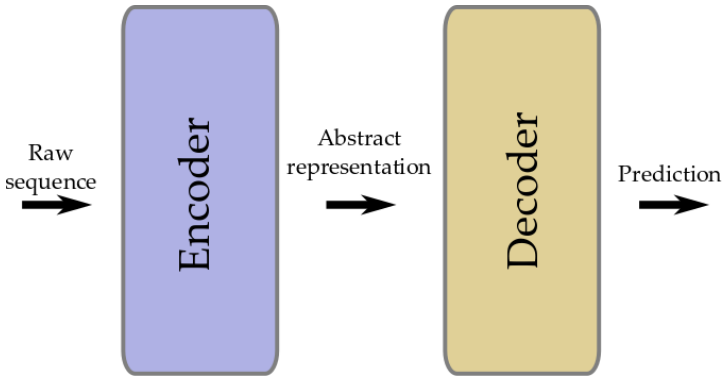



Figure 30: Sketch of the general structure of all developed models: A raw sequence of non-fixed length is fed to a decoder generating a new representation of the sequence. An encoder is then used to generate a prediction from the abstract representation of the sequence.

### 5.1.1 *Bidirectional RNNs*

The LSTM and GRU layers described in Section 3.1.2 generate outputs $h_t$ that are functions of previous inputs - we say that the layers look backward in time. A bidirectional LSTM (or GRU) [66] consists of 2 parallel layers. The input sequence is fed to the first layer as it is, but the second layer receives the sequence in reverse order as shown in Figure 31.

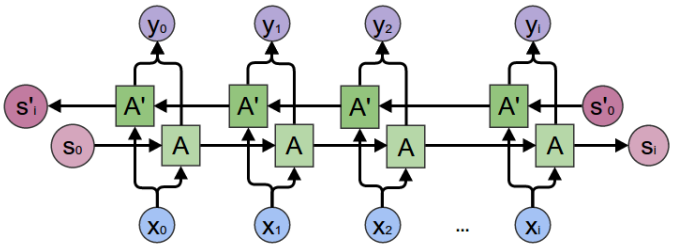



Figure 31: Illustration of bidirectional RNNs. The input sequence is fed to two, different RNN layers, where one of the layers receives the sequence in reverse, and the outputs are combined (by e.g. concatenation). Figure from [67].

In this work, the outputs of the layers are concatenated in the feature dimension. Using this method, the *i'th* output is generated to contain information about the past, the present and the future. A bidirectional layer is thus a function

$$f : \mathbb{R}^{L \times F} \rightarrow \mathbb{R}^{L \times 2F'}, \tag{65}$$

where $L$ is the sequence length, $F$ is the number of input features and $F'$ is the number of neurons (and consequently the number of outputs) in each LSTM (or GRU).

### 5.1.2 *Residual Connections*

Standard FF layers are used throughout this work in both encoders and decoders, but an extension of it is used as well called *residual blocks* [68]. A residual connection (or ResBlock) is shown in Figure 32: Instead of having a subnetwork with input $x$ approximate some function $y = f(x)$, the subnetwork approximates the *residual* $f(x) = y - x$ instead (hence $y = f(x) + x$). This corresponds to adding an identity connection between the input and output of a layer as shown.

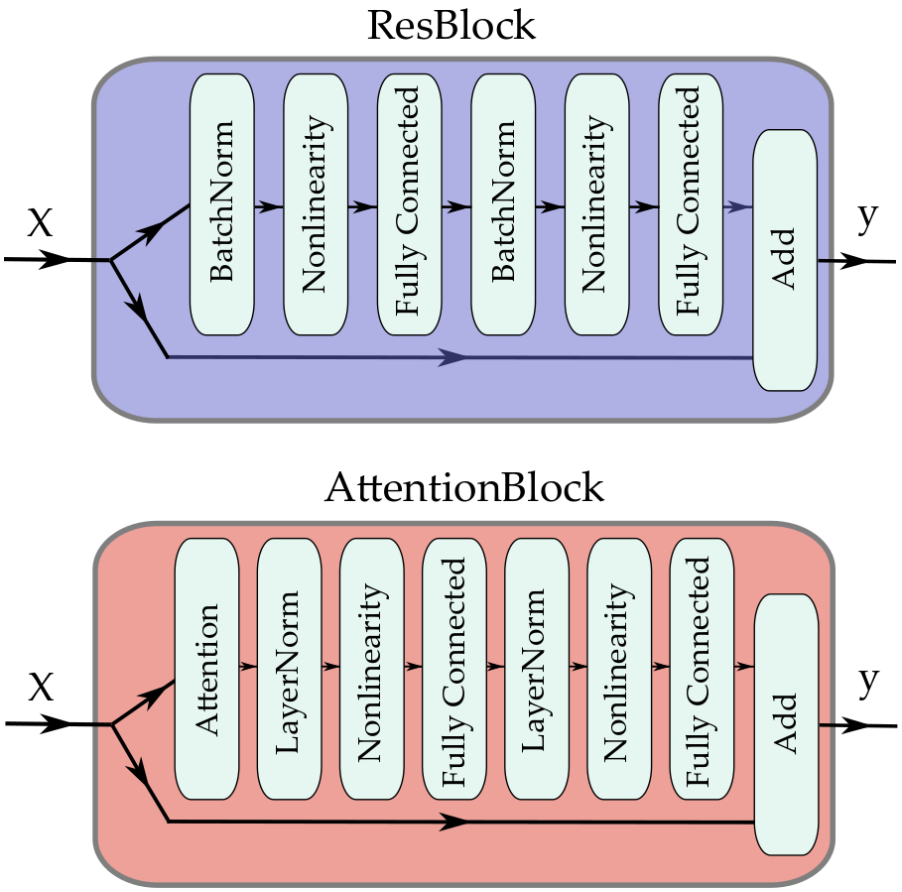



Figure 32: Schematic overview of the ResBlock (upper) and AttentionBlock (lower) modules utilized in this work.

Deep networks with residual connections are easier to train than their classic counterparts due to easier flow of information through the network. Several variants of residual connnections are used in applied DL. In this work, the ResBlock shown in Figure 32 is used [69] with BN and FF layers as described in Section 3.1.

*The popular ResNets can be more than a 1000 layers deep.*

### 5.1.3 *Attention Blocks*

In this work the attention mechanism introduced in Section 3.1.3 is applied in encoding layers with and without RNNs. The specific attention submodule used is a combination of the SelfAttention mechanism from [51] and the ResBlock introduced above: For each element of an input sequence, a linear combination of sequence elements $\boldsymbol{h}_i$ is created using the attention mechanism given by Eq. (38). Each $\boldsymbol{h}_i$ is then normalized, nonlinearized and sent through a FF layer two times as shown in the bottom panel of Figure 32. The ResBlock applies LN instead of BN, since the latter is not well defined for sequential data as previously discussed.

### 5.1.4 *Many-to-One Layers*

To decode the output sequence of the encoder, the non-fixed length sequences are mapped to a fixed-size feature vector using some function $f$ of the form

$$f : \mathbb{R}^{L \times F} \to \mathbb{R}^{F}. \tag{66}$$

In this work, three different methods are used: Max pooling (MaxPool), average pooling (AvePool) and dropping everything but the last output (in the case of using an RNN). Given a sequence $\mathcal{S} = \{\boldsymbol{s}^1, ..., \boldsymbol{s}^L\}$ of vectors $\boldsymbol{s}^i = [s_1^i, ..., s_F^i]$, MaxPool creates a new vector

$$\text{MaxPool}(\mathcal{S}) = \left[\max\left\{s_1^j\right\}_{j=1}^{L}, ..., \max\left\{s_F^j\right\}_{j=1}^{L}.\right] \tag{67}$$

Likewise, AvePool creates the vector

$$\text{AvePool}(\mathcal{S}) = \left[\text{mean}\left\{s_1^j\right\}_{j=1}^{L}, ..., \text{mean}\left\{s_F^j\right\}_{j=1}^{L}.\right] \tag{68}$$

## 5.2 HYPERPARAMETERS

With the model building blocks defined, we now move to a discussion of how the blocks must be composed, which requires tuning of several hyperparameters. Since the space of hyperparameters to choose from is horribly large, only the (arguably) most important hyperparameters have been investigated in this work. These (besides width, depth and submodule composition) are now unpacked further.

### 5.2.1 *Learning Rate*

The LR $\eta$ is widely agreed to be one of the most important hyperparameters to tune [70].

If the learning rate is too large, the optimizer will overshoot the target leading to poor generalization or the training error (error on the trainset) might begin to increase. A too small learning has consequences as well such as slowing down the training, both in convex and non-convex optimization. In convex optimization, a too small learning rate is not as critical as it is in the case of optimizing highly nonconvex NNs (see Figure 21), where a too small learning rate can lead to getting stuck in local minima.

In an attempt to avoid these scenarios, *LR scans* have been employed [71]. Before the actual training proces is begun, a LR scan consists of monitoring the loss while the LR is increased for a fixed number of optimizer updates. Here LR scans have been performed to find upper and lower bounds on the LR to employ during training. The maximal LR used during training was then chosen to be close to this maximum,

since high LRs tend to have a regularizing effect [72]. Such a LR scan using Adam as the optimizer with BS 256 is shown in the right panel of Figure 33. In this case, the LR becomes so high that the optimization becomes unstable close to $\eta = 4 \cdot 10^{-3}$, and therefore the maximal LR was chosen to be comfortably far below this limit.

The stochastic nature of the optimization is clearly seen as high frequency noise of the loss value. It should be noted that finetuning the LR is not as important when Adam is used due to its adaptive nature.

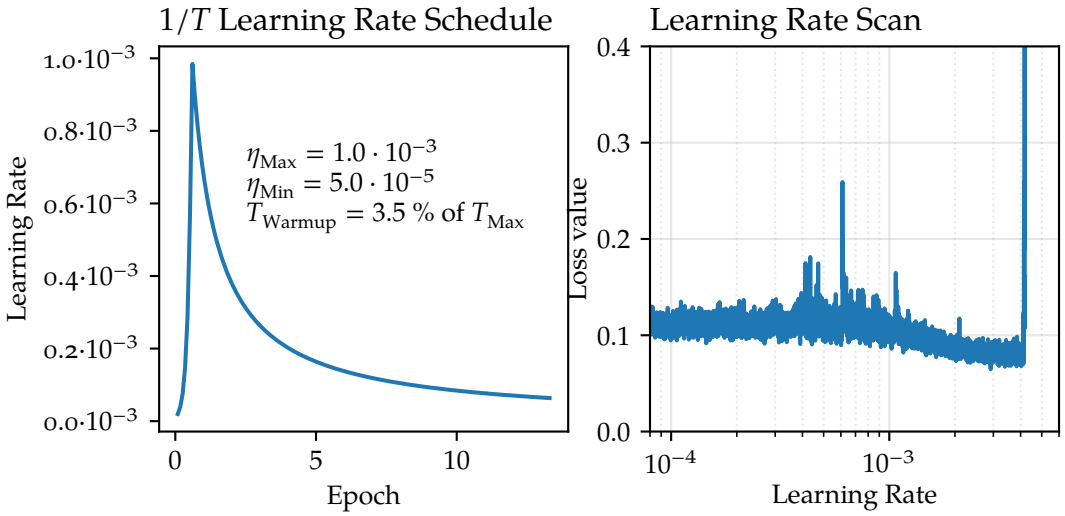



Figure 33: A LR scan (right) using a BS of 256 and the LR schedule used in the subsequent training (left).

Once the upper and lower bounds of the LR have been found, training can begin. It has been observed that optimizing NNs with Adam can lead to poor generalization if a warm-up heuristic is not employed [73]. This effect stems from the way the moments in Eqs. (49 - 54) are estimated. For instance, the second moment changes on a characteristic time scale

$$T \approx \frac{1}{1-\beta_2}, \tag{69}$$

except during the early stages of training, where it can rapidly vary due to the small amount of data the optimizer has seen. Therefore each training is initiated with a warm-up period, where the LR is exponentially increased from $\sim$ 1% of its maximal value. Once the maximal value has been reached, a LR decay is initiated. It can be shown [74] that to achieve optimal convergence rate, the LR must decrease like

$$\eta \sim \frac{1}{T}. \tag{70}$$

Hence, the LR schedule employed in all model optimizations is

$$\eta(t) = \begin{cases} a \cdot \eta_{\max} \cdot \alpha^t & \text{if } 0 < t < T_{\text{warmup}} \\ \frac{1}{1+\beta(t-T_{\text{warmup}})} \cdot \eta_{\max} & \text{if } T_{\text{warmup}} \leq t \leq T_{\max}, \end{cases} \tag{71}$$

where $a$ typically was chosen to be 5 % and $\alpha$ and $\beta$ were found by solving the equations $\eta(T_{\text{warmup}}) = \eta_{\max}$ and $\eta(T_{\max}) = \eta_{\min}$. $T_{\text{warmup}}$ was typically set to 2 % of the total number of optimization steps, and $T_{\max}$ was chosen to correspond to 10-20 epochs of training for all models, since such values led to the best results. The LRs used during one experiment is shown in the left panel of Figure 33 depicting the described schedule.

### 5.2.2 *Activation Functions*

The specific choice of activation function can have a tremendous effect on a deep NN's ability to converge. If sigmoid-like activations are used, NNs tend tend to suffer from *vanishing gradients* [75], where the gradients go to zero exponentially fast during backpropagation and consequently are unable to learn. Therefore only activation functions that are unbounded above have been investigated here. These are LeakyReLU [76] and Mish [77]. Their respective analytic expressions are

*The vanishing gradients problem arise, when* $\max(|\nabla f(x)|) < 1$ *for the used activation function* $f(x)$.

$$\text{LeakyReLU}(x) = \max(0.01 \cdot x, x), \tag{72}$$

$$\text{Mish}(x) = x \cdot \tanh\left[\ln(1 + e^x)\right], \tag{73}$$

and are shown in Eq. (34) (solid lines) along with their derivatives (dashed lines).

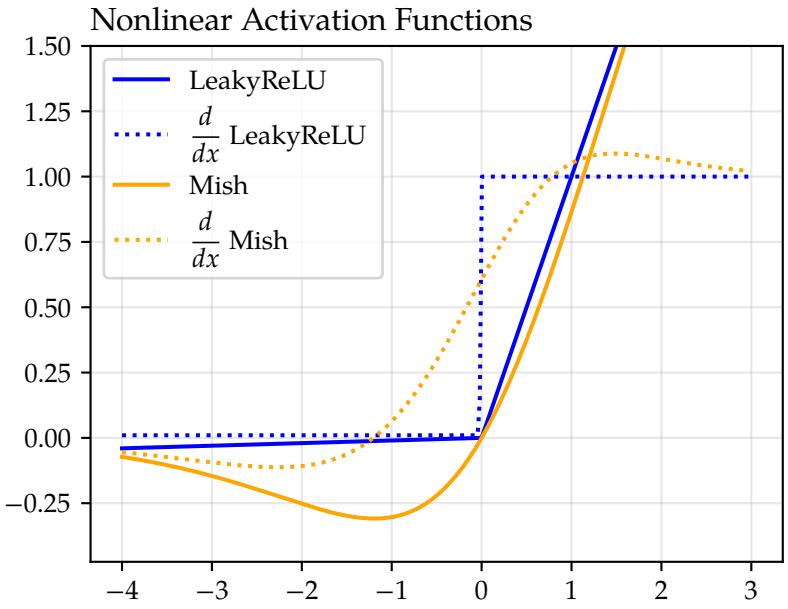



Figure 34: The ReLU (solid blue) and Mish (solid orange) activation functions along with their derivatives (dotted).

LeakyReLU has stood the test of time and has the advantage of being fast to evaluate and has a non-zero derivative everywhere (except at $x = 0$). The more sophisticated activation function Mish has the advantage of being smooth and partly self-normalizing (the output distribution is closer to a unit Gauss, when the input is standardized), which has been conjectured to be important when training deep networks.

### 5.2.3 *Dropout*

As discussed in Section 3.1.1, wide and deep networks are powerful models - and are therefore prone to overfitting. Dropout [78] is an elegant and simple mechanism attempting to prevent just that.

Dropout applied to a layer $l$ works as follows: During each iteration of training, a neuron $f$ is removed from the network with probability $p$ - it is dropped out. If a neuron is not removed, its output is scaled by a factor $\frac{1}{1-p}$. Formally, a Dropout layer can be described as

$$\text{Dropout}(f, p) = X \cdot \frac{1}{1-p} \cdot f, \tag{74}$$

where $X \sim \text{Bernoulli}(\text{p})$ is a Bernoulli random variable. During inference, Dropout is not applied. Qualitatively this corresponds to using a different subnetwork at every training step, effectively generating an ensemble of several networks at inference time. Besides early stopping, Dropout is the only regularization explicitly applied during model development.

### 5.2.4 *Weight Initialization*

Before training starts, the weights of the NN have to be initiated. If BN (or LN) is not applied, the weights of the network must be carefully initialized to achieve satisfactory convergence [79], and several initialization schemes exist. If, however, normalization layers are employed, the used weight initialization scheme becomes less important. Therefore only *Kaiming initialization*[80] has been used, where the weights $w$ of a layer with $n$ neurons are drawn from the normal distribution

$$w \sim \mathcal{N}(0, \frac{2}{n}). \tag{75}$$

For initial hidden states $h_0$ in RNN layers, the initial hidden states have been learned.

## 5.3 LOSS FUNCTIONS AND PERFORMANCE MEASURES

Models performing energy regression (ER), direction regression (DR), interaction vertex regression (IVR), interaction time regression and classification has been developed in this work and the diversity of the different tasks calls for an arsenal of different loss functions and performance metrics.

### 5.3.1 *Performance Measures*

To determine the quality of reconstructions, the primary performance measures are based on the distributions of errors. For each type of regression, the error measure might vary, but all error measures are functions $f$ taking a pair of either numbers or vectors $(x_{\text{reco}}, x_{\text{true}})$ as input and outputs a single number $e$ - the error. The *width* of the *distribution* of $e$ measures the performance of the reconstruction algorithm, where lower is better (a width of 0 would correspond to perfect reconstructions). In this work, 2 different kinds of error distributions are considered: Distributions bounded below by 0 and unbounded distributions. The unbounded error distributions are not necessarily Gaussian, and therefore the width $W$ is determined as

$$W(e) = \frac{IQR(e)}{1.349}, \tag{76}$$

where $IQR$ is the interquartile range (difference of 75th and 25th percentiles), and the constant 1.349 ensures that $W$ corresponds to one standard deviation, if one makes the assumption that the errors are Gaussian.

The *error on the width* $\sigma_W$ can be determined using *order statistics*[81], where the *q'th* order statistic $X_{(q)}$ of a sample is the value of the sample's *q'th* smallest value. In order for the *q'th* order statistic to have some

value $x$, there must be $q-1$ values smaller than $x$ and $n-q$ values larger than $x$. The probability that a value is *smaller* than $x$ is $F(x)$ and the probability that a value is *larger* than $x$ is $1-F(x)$, where $F(x)$ is approximated by the empirical distribution function. Furthermore, the probability $P(X_{(q)}=x)$ is exactly $f(x)$, where $f$ is approximated as the empirical probability density. The probability density function of $X_q$ is therefore given by

$$f_{X_{(q)}}(x) = \frac{n!}{(n-q)!(q-1)!} \cdot F(x)^{q-1} \cdot [1-F(x)]^{n-q} \cdot f(x), \tag{77}$$

which approaches a normal distribution as $n \to \infty$. Since the *k'th* percentile is nothing but $X_{(np)}$, where $p = k/100$, the percentiles are distributed as[82]

$$X_{(np)} \sim \mathcal{N}\left(x_p, \frac{p(1-p)}{nf(x_p)^2}\right), \tag{78}$$

where $x_p = F^{-1}(p)$. Using the law of error propagation (neglecting correlations)

$$\sigma_f^2 = \sum_i \frac{\partial f}{\partial x_i}^2 \sigma_{x_i}^2, \tag{79}$$

the estimated standard error on the width $\sigma_W$ is given by

$$\sigma_W = \frac{1}{1.349}\sqrt{\frac{0.25(1-0.25)}{n}\left(\frac{1}{f(e_{0.25})^2} + \frac{1}{f(e_{0.75})^2}\right)}. \tag{80}$$

For error distributions bounded below by 0 such as the distance between the predicted and the true interaction vertex, the width of the distribution is uninteresting. Therefore the performance $U$ of such distributions is reported as the 68th percentile of the error distribution

$$U(e) = e_{0.68}, \tag{81}$$

i.e. the 68 % confidence interval. Likewise, the standard error $\sigma_U$ on the upper bound is given by

$$\sigma_U = \sqrt{\frac{0.68(1-0.68)}{nf(e_{0.68})^2}}. \tag{82}$$

### 5.3.2 *Energy Regression*

Before constructing a loss function, one must consider what it means for a reconstruction to be 'good' or 'bad'. In the specific case of ER, being off with, say, 10 GeV has radically different interpretations, if the

true energy is 1 GeV or 1 PeV. These considerations lead to a relative error measure such as

$$\mathrm{RE}(E_{\mathrm{reco}}, E_{\mathrm{true}}) = \frac{E_{\mathrm{reco}} - E_{\mathrm{true}}}{E_{\mathrm{true}}} \tag{83}$$

being preferred to an absolute error measure.

Eq. (83) has an undesired consequence: When uncertain, a NN can reduce the loss by generating reconstructions that are too low. This effect stems from the fact that Eq. (83) is bounded below - for instance, a prediction of twice the true energy corresponds to a larger loss than a prediction of half the true energy (i.e. $|\mathrm{RE}(2 \cdot E_{\mathrm{true}}, E_{\mathrm{true}})| > |\mathrm{RE}(\frac{1}{2} \cdot E_{\mathrm{true}}, E_{\mathrm{true}})|$). Therefore another relative error measure is used instead: The logarithmic error (LE) given by

$$\mathrm{LE}(E_{\mathrm{reco}}, E_{\mathrm{true}}) = \log(E_{\mathrm{reco}}) - \log(E_{\mathrm{true}}) = \log\left(\frac{E_{\mathrm{reco}}}{E_{\mathrm{true}}}\right). \tag{84}$$

The LE has the desired properties of being unbounded both from below and above, and it treats predictions of twice the true energy as being equally as bad as predictions of half the true energy, i.e. ($|\mathrm{LE}(2 \cdot E_{\mathrm{true}}, E_{\mathrm{true}})| = |\mathrm{LE}(\frac{1}{2} \cdot E_{\mathrm{true}}, E_{\mathrm{true}})|$) as shown in Figure 35, where logcosh(LE) is compared to logcosh(RE). As can be seen, a NN is punished more for predicting too low energies, if the LE is used. The advantage of applying the transformation to the target energy given by Eq. (63) now becomes evident: Since

$$\begin{aligned}(A \cdot \log E_{\mathrm{reco}} + B) - (A \cdot \log E_{\mathrm{true}} + B) \\ = A \cdot \log\left(\frac{E_{\mathrm{reco}}}{E_{\mathrm{true}}}\right),\end{aligned} \tag{85}$$

Eq. (63) is designed to directly minimize Eq. (84), since they are equal up to a multiplication factor, which can be absorbed into the definition of the LR.

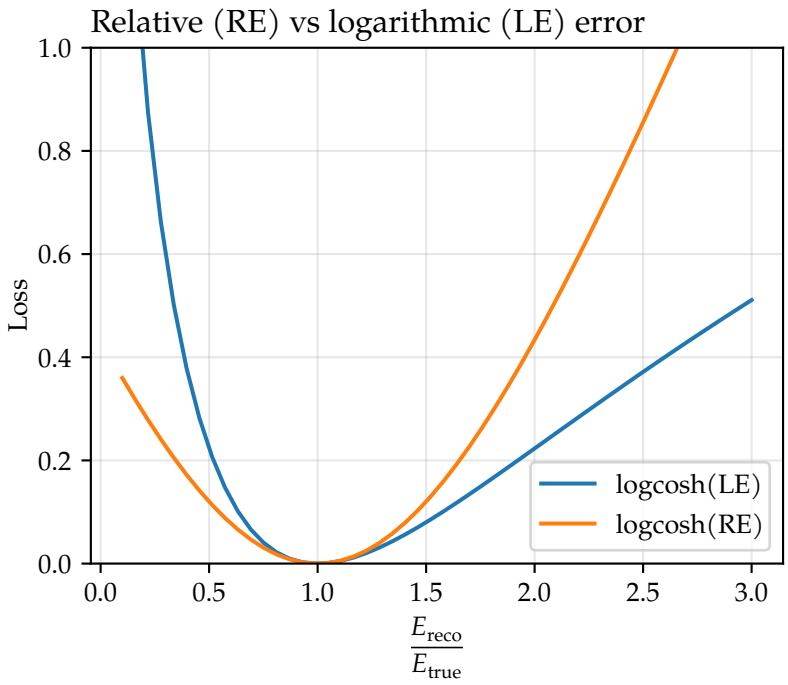



Figure 35: Comparison of the logarithmic error (blue) to the relative error (orange).

### 5.3.3 *Direction Regression*

The direction of an incoming particle is described by the unit vector parallel with the particle direction. Models have been developed with the unit vector solely parametrized using cartesian coordinates $(x, y, z)$, since parametrizations using e.g. spherical coordinates introduce discontinuous effects that are more difficult to take into account; it has been attempted, and it resulted in poor performance. For cartesian coordinates, however, small variations of the unit vector causes small variations to its cartesian parametrization.

*small changes to unitvectors with $\phi$ close to 0 may cause their azimuthal component to jump back and forth between $2\phi$ and 0*

The developed models for DR have not been forced to predict vectors of length 1, but merely vectors in $\mathbb{R}^3$ (despite the fact that a unit vector in $\mathbb{R}^3$ can be uniquely determined by 2 numbers), solely because of the fact that such models performed best. Loss functions based on 3 different error metrics have been tested:

1. Angular distance loss functions

$$\text{AngleDist}(\boldsymbol{r}_{\text{reco}}, \boldsymbol{r}_{\text{true}}) = \arccos \frac{\boldsymbol{r}_{\text{reco}} \cdot \boldsymbol{r}_{\text{true}}}{|\boldsymbol{r}_{\text{reco}}||\boldsymbol{r}_{\text{true}}|} \tag{86}$$

reducing the angle $\Psi$ between the reconstructed direction $\boldsymbol{r}_{\text{reco}}$ and the true direction $\boldsymbol{r}_{\text{true}}$,

2. the negative cosine similarity

$$\text{NegSim}(\boldsymbol{r}_{\text{reco}}, \boldsymbol{r}_{\text{true}}) = 1 - \frac{\boldsymbol{r}_{\text{reco}} \cdot \boldsymbol{r}_{\text{true}}}{|\boldsymbol{r}_{\text{reco}}||\boldsymbol{r}_{\text{true}}|}, \tag{87}$$

   where adding a different constant is just as viable. Adding 1 merely ensures that a loss of 0 corresponds to a perfect reconstruction. The final error measure is

3. the Euclidian distance between $\boldsymbol{r}_{\text{reco}}$ and $\boldsymbol{r}_{\text{true}}$ given by

$$\text{EuclidDist}(\boldsymbol{r}_{\text{reco}}, \boldsymbol{r}_{\text{true}}) = |\boldsymbol{r}_{\text{reco}} - \boldsymbol{r}_{\text{true}}|, \tag{88}$$

   which in cartesian coordinates is equivalent to L1-loss.

As previously mentioned, it is key to make the definition precise of what a 'good' reconstruction is. For oscillation analyses, the azimuthal component of the direction is not of great interest, since the distance travelled for an atmospheric neutrino solely depends on the zenith angle (see Figure 5). The azimuthal component is, however, of great interest, if one searches for e.g. supernovae or sneaky BG-particles travelling along directions with poor resolution. Therefore 2 different performance measures are used:

1. The widths of the error distributions (Eq. (76)) of the azimuthal and zenith angle reconstructions and
2. the upper bound $U$ (Eq. (81)) on the distribution of the angular error (Eq. (86)).

The latter performance measure has the advantage of being agnostic to the chosen coordinate system - a pleasant feature outside of oscillation analyses.

### 5.3.4 *Vertex Regression*

IVR is not directly important for oscillation analyses, but as mentioned, it is important with regards to separating BG from signal: Due to the quasi-hexagonal symmetry of the Icecube detector, there are certain paths through the detector with little instrumentation. The better the IV can be determined, the better the BG rejection therefore becomes.

In contrast to ER and DR, it is clearer what a well-reconstructed interaction vertex entails. In this work, all loss functions for vertex regression are based on the difference between the true $\boldsymbol{p}_{true} = (t_{true}, \boldsymbol{x}_{true})$ and reconstructed $\boldsymbol{p}_{reco} = (t_{reco}, \boldsymbol{x}_{reco})$ vertices given by

$$\text{VertexDist}(\boldsymbol{x}_{reco}, \boldsymbol{x}_{true}) = |\boldsymbol{p}_{reco} - \boldsymbol{p}_{true}| \tag{89}$$

and

$$\text{TimeDist}(t_{reco}, t_{true}) = t_{reco} - t_{true}. \tag{90}$$

The spatial components of the interaction vertex are inherently on the same scale and therefore contribute equally to loss functions. The temporal component, however, is not guaranteed to be on the same scale as the spatial components. It might therefore be the case that it is beneficial to split the vertex regression into independent temporal and spatial components.

The performance measures used for VR are the widths given by Eq. (76) of the distributions of each of the 4 components contributing to Eq. (89) and the upper bound given by Eq. (81) of the distribution of the Euclidian distance between the spatial components of $\boldsymbol{p}_{reco}$ and $\boldsymbol{p}_{true}$.

### 5.3.5 *Probabilistic Regression*

In addition to point-like predictions, probabilistic regression has also been investigated, where a *distribution* is predicted instead of a single *point*. In this work, it is assumed that each variable of interest $x$ is normally distributed, i.e.

$$x \sim \mathcal{N}(\mu, \sigma^2), \tag{91}$$

where the mean $\mu$ is and variance $\sigma^2$ is estimated. The NN is then trained to approximate a likelihood function, and the negative log-likelihood

$$l(x, \mu, \sigma^2) = -\log\left[P(x|\mu, \sigma^2)\right] \tag{92}$$

is used as loss function, which can be expanded to give

$$l(x, \mu, \sigma^2) = \log\sigma + \frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 + C, \tag{93}$$

where $C$ is an irrelevant constant.

*The constant $C$ is irrelevant, since it does not contribute to $\nabla_w l$ or the placement of the minimum.*

### 5.3.6 *Classification*

For all classification models, the only employed loss function is the cross entropy given by

$$l(p^{reco}, p^{true}) = -\mathbb{E}_{p^{true}}[\log p^{reco}], \tag{94}$$

where $p^{reco}$ is the predicted probability mass function (pmf), $p^{true}$ is the true pmf and $\mathbb{E}_{p^{true}}$ denotes expectation with respect to the true pmf. To force model predictions to be interpretable as probabilities, a final Softmax activation layer (see Eq. (39)) is added to classification NNs. Since the true distribution only takes on the values 0 or 1 (a particle is either type A or not type A), Eq. (94) reduces to

$$l(p^{reco}, p^{true}) = -\log p_a^{reco} \tag{95}$$

for both binary and multiclass classification, where $p_a^{reco}$ denotes the probability that an event belongs to class $\mathcal{A}$. For binary classification, the area under the curve (AUC) of the receiver operating characteristic (ROC) is used as performance metric. An illustration of a ROC curve and its relation to some decision threshold $t$ is shown in Figure 36: The ROC-curve is a plot of the true positive rate (TPR) as a function of the false positive rate (FPR), where the TPR is the probability of classifying signal as signal and the FPR is the probability of classifying background (BG) events as signal given the chosen decision threshold. Formally, $\text{TPR}(t) = 1 - F_{\text{sig}}(t)$ and $\text{FPR}(t) = 1 - F_{\text{BG}}(t)$, where $F_{\text{sig}}$ is the empirical distribution function of signal scores and $F_{\text{BG}}$ is the empirical distribution function of BG scores.

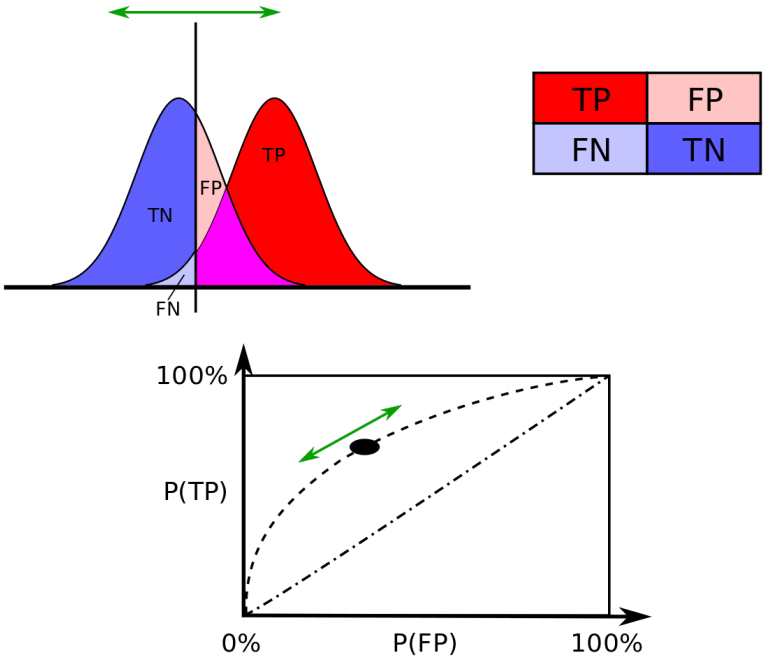



Figure 36: Illustration of a ROC curve (lower figure) along with its relation to the chosen decision threshold (vertical line in upper figure). P(TP) and P(FP) are equivalent to TPR and FPR respectively. Figure from [83].

The AUC is equal to the probability that a randomly selected signal event is given a higher score (here $p^{reco}$) than a randomly selected BG event, which can be seen by inspecting the definition of the AUC given by

$$AUC = \int_0^1 \text{TPR}(\text{FPR}) d\text{FPR}. \tag{96}$$

We now make a change of variables such that the integral is over the decision threshold; since

$$\text{FPR}(t) = 1 - F_{\text{BG}}(t) \Rightarrow \frac{d\text{FPR}}{dt} = -f_{\text{BG}}(t), \tag{97}$$

where $f_{\text{BG}}$ is the empirical pdf of BG scores. Inserting the substitution we find

$$AUC = \int_{\infty}^{-\infty} \text{TPR}(t)[-f_{\text{BG}}(t)]dt \quad = \int_{-\infty}^{\infty} \text{TPR}(t) f_{\text{BG}}(t)dt = \mathbb{E}_{f_{\text{BG}}}[TPR], \tag{98}$$

which is exactly a rescaling of the Mann-Whitney test statistic $U_{MW}$ [84, 85], which is related to the AUC by

$$AUC = \frac{U_{MW}}{N_{\text{sig}} \cdot N_{\text{bg}}}, \tag{99}$$

where $N_{\text{sig}}$ and $N_{\text{bg}}$ are the number of signal and BG events respectively. Furthermore, for arbitrary distributions $f_{\text{sig}}$ and $f_{\text{BG}}$

$$\lim_{N_{\text{sig}}, N_{\text{bg}} \to \infty} U_{MW} \sim \mathcal{N}(\mu_{U_{MW}}, \sigma^2_{U_{MW}}) \tag{100}$$

where $\mu_{U_{MW}}$ is some mean and $\sigma^2_{U_{MW}} < \frac{N^2_{\text{sig}} N^2_{\text{bg}}}{4 \min(N_{\text{sig}}, N_{\text{bg}})}$ [86, 87]. Therefore the standard error on the AUC is approximated by its upper bound

*By calculating the AUC using the Mann-Whitney test statistic, no numerical error is introduced from approximating an area using e.g. the trapezoidal rule.*

$$\sigma_{AUC} = \sqrt{\frac{1}{4 \min(N_{\text{bg}}, N_{\text{sig}})}}. \tag{101}$$

## 5.4 WEIGHTS

Since the distributions of energy, particle type, interaction vertex and direction (see Section A.1) are not uniform, the dataset is reweighted for training models. The reweighting is an attempt at avoiding models being biased towards certain predictions; if e.g. 99 % of targets in a classification problem were type A, the model could simply predict everything to be type A and achieve 99 % accuracy without actually learning the features characteristic to particle A.

For regression models, 4 different sets of weights have been used: A balancing weightset $w_{balanced}$, a low-energy focused weightset $w_{low}$, a high-energy focused weightset $w_{high}$ and a resolution-blinding weightset $w_{blinding}$ (what is meant by resolution-blinding is unpacked below). The aforementioned weights are all shown in Figure 37.

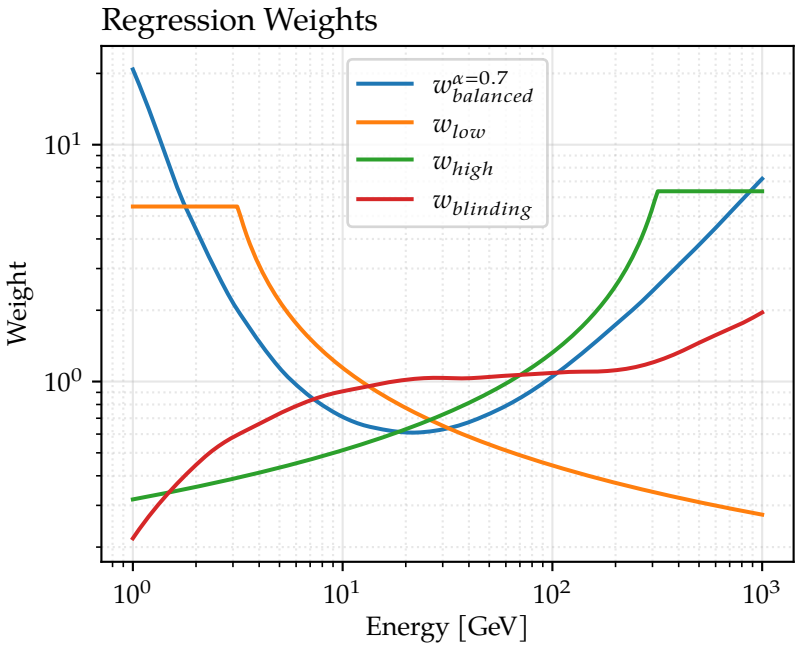



Figure 37: All energy-dependent weights used for regression. All weights are scaled such that the *average* weight is 1.

The balancing weights are calculated as

$$w_{balanced}(x) = C \cdot \frac{1}{f_{spl}(x)^{\alpha}}, \tag{102}$$

where $f_{spl}$ is a quadratic spline approximation to the histogram of the target of interest $x$, $\alpha$ is tuned to each specific regression model, but preferably $\alpha = 1$, and $C$ is a normalization constant which ensures that the average weight is 1. Hence, $C$ satisfies

$$\frac{1}{C} = \frac{1}{N_{tot}} \sum_{i=1}^{N_{bins}} w(x_i) N_i, \tag{103}$$

where $N_{tot}$ is the dataset size, $x_i$ is the *i'th* bincenter and $N_i$ is the number of entries in the *i'th* bin.

The low- and high-energy focused weights have been assigned using the functions

$$w_{low}(x) = \begin{cases} C & \text{if } x < x_{low} \\ C \cdot \frac{1}{1+\alpha(x-x_{low})} & \text{if } x \geq x_{low} \end{cases} \tag{104}$$

and

$$w_{high}(x) = \begin{cases} C & \text{if } x > x_{high} \\ C \cdot \frac{1}{1+\alpha[x_{max}-x]} & \text{if } x < x_{high} \end{cases}, \tag{105}$$

where $C$ is calculated using Eq. (103). For ER, $x_{low} = 0.5$, $x_{high} = 2.5$ and $\alpha$ was chosen such that

$$\frac{\max(w_{low/high})}{\min(w_{low/high})} = 20. \tag{106}$$

The motivation behind using different weights for different energy-ranges is that the predictive power of a feature might be energy-dependent. Furthermore, since there is no guarantee that the optimal set of weights and biases can be learned for a single model covering the entire energy range, different models might have to be used for different energy ranges.

The resolution of experiments like Icecube tends to improve when the energy increases. For a model to be as performant as allowed given the resolution, we require it to perform better at higher energies by placing more weight on those events - it is attempted to blind the model to the detector's inherent resolution. Here, a proxy for the resolution is chosen to be the performance of the Retro algorithm measured as the width $\sigma_{\text{Retro}}$ of the relative energy error distribution calculated using Eq. (76) and Eq. (83). Hence,

$$w_{blinding}(x) = C \cdot \frac{1}{\sigma_{\text{Retro}}}, \tag{107}$$

where, again, $C$ is calculated using Eq. (103).

With reweighted datasets, the empirical loss given by Eq. (5.4) is modified to

$$\hat{L} = \frac{1}{N} \sum_{i=1}^{N} w_i \cdot l_i. \tag{108}$$

Reweighted distributions using the different weights are shown in Figure 38. For $w_{balanced}$, $\alpha = 0.7$ to ensure that no weights are too large (the largest weight is 10). If larger weights are used, the risk increases of accidentally making a too large update to the model parameters during backpropagation, potentially ruining the model's performance.

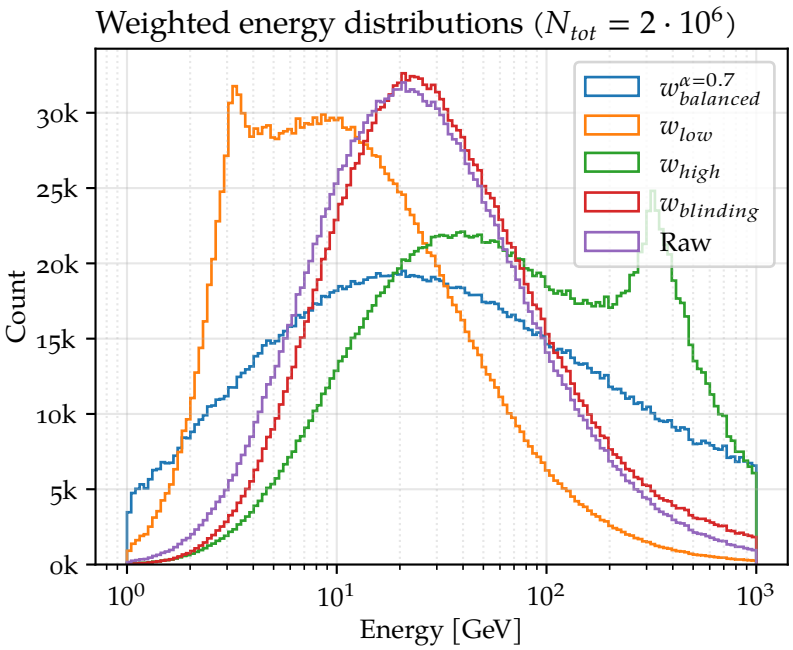



Figure 38: Reweighted neutrino energy distributions for a subset of the entire dataset. Weight calculations are given by Eqs. (102, 104, 105 107). The unweighted distribution is shown in purple for comparison.

All used weights have been calculated using the methods described above.

## 5.5 HYPERPARAMETER OPTIMIZATION

In search of the optimal model, several hyperparameter optimization algorithms can be applied such as Bayesian optimization[88], random search and grid search. In this work, a combination of grid search and educated guessing is used.

*The combination of grid search and guessing is infamously known as 'Grad Student's Descent'.*

A full grid search is an exhaustive search over all possible hyperparameter configurations, which often is not feasible. Since model performance is more sensitive to some hyperparameters than others, a full grid search would likely be a waste of time. The space of hyperparameters searched is summarized in Table 4.

| Hyperparameter | Searchspace |
|---|---|
| Batchsize | 32, 64, 128, 256, 512 |
| Optimizer | SGD, Adam, NAG |
| LR schedule | Inverse decay w. warmup |
| Layer Widths | 64, 128, 256, 512, 1028 |
| Decoding ResBlocks | 0, 1, 2, 3, 4, 5, 6 |
| Encoding Att. Blocks | 0, 1, 2, 3, 4, 5, 6, 7 |
| Encoding RNN layers | 0, 1, 2, 3, 4 |
| Encoding RNN type | Vanilla, GRU, LSTM, BiGRU, BiLSTM |
| Nonlinearity | LeakyReLU, Mish |
| Encoding norm. | None, LayerNorm |
| Decoding norm. | None, BatchNorm |
| Regularization | None, Dropout($p \in [30\%, 50\%, 80\%]$) |
| Regression loss | L1, L2, logcosh |
| Classification loss | CrossEntropy |
| Many-to-One | MaxPool, AvePool, KeepLast |
| Weight init. | Kaiming |

Table 4: Summary of the hyperparameter searchspace. The regression loss functions are not complete; regression-specific alterations are introduced later.

All models were created, trained and evaluated using the Pytorch ML framework [89].

### 5.5.1 *General Core Architecture*

Most ressources have been allocated to finding the right depth, the right widths and right submodule components for a general core architecture. This strategy is motivated by an a priori notion that such a general architecture exists, and it is capable of extracting an optimal amount of information from sequences from which specific regression problems can be solved. The various architectures investigated can be divided into 3 meta-types shown in Figure 39:

1. Networks with purely Attention-based encoders (top),
2. Networks with purely RNN-based encoders (middle), and
3. Networks with encoders constructed using blocks of an RNN layer immediately followed by an Attention layer (bottom).

For simplicity, all experiments with the purpose of finding a core model were carried out only using the $\nu_\mu$-sample, $w_{blinding}$ and doing all regression tasks (full reconstruction) with a single NN using SRT-cleaned pulses. The search was executed in the following way: Starting with the simplest, most (if not all) sensible combinations of RNN layers,

AttentionBlocks and ResBlocks were investigated. Model capacities were then increased until no further improvement was seen. As the optimal architecture was believed to be in close proximity, experiments with different nonlinearities, BSs etc. was done.

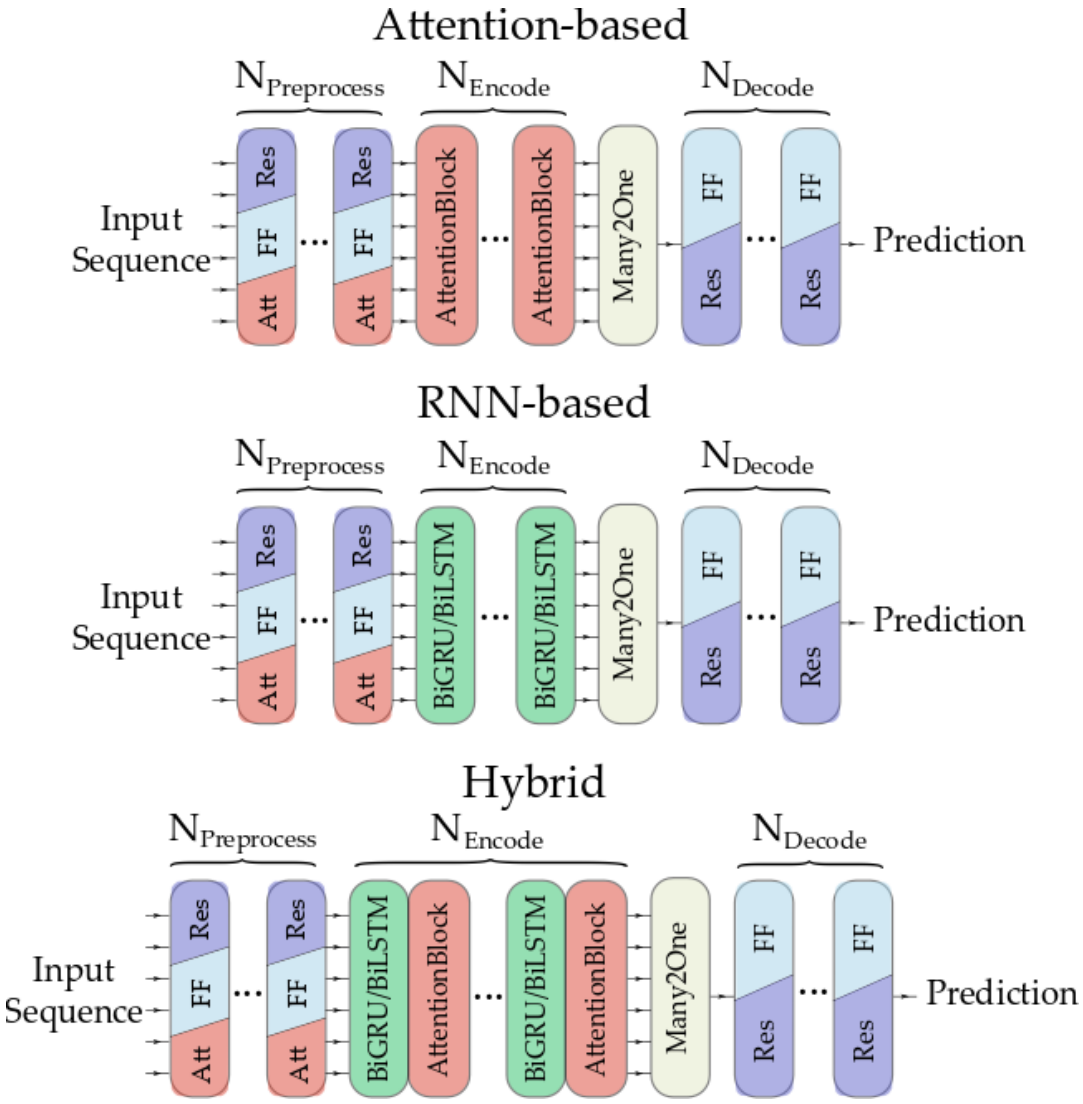



Figure 39: Graphical overview of the searchspace of the different architectures divided into 3 metatypes: Attention-based (top), RNN-based (middle) and a hybrid (bottom). Each metatype encoder is potentially preceded by a preprocessing scheme using either AttentionBlocks (red), standard FF-layers (light blue) or ResBlocks (dark blue) and followed by a decoder consisting of either FF-layers or ResBlocks.

To compare the performances of the different architectures, an architecture achieving a smaller minimum average loss is said to perform better. The disadvantage of this approach is that it is only viable when the same loss function is used, and therefore the loss function must be optimized afterwards. The assumption behind this approach is that if architecture A achieves a lower loss than architecture B using a loss

function $f$, A will also achieve a lower loss than B if another loss function $g$ is used. The used loss function was therefore

$$\begin{aligned} L = \frac{1}{N}\sum_{i=1}^{N} & \text{logcosh}(\text{LE}(E_{reco}, E_{true})_i) \\ & + \text{logcosh}(\text{EuclidDist}(\boldsymbol{r}_{\text{reco}}, \boldsymbol{r}_{\text{true}})_i) \\ & + \text{logcosh}(\text{VertexDist}(\boldsymbol{p}_{reco}, \boldsymbol{p}_{true})_i), \\ & + \text{logcosh}(\text{TimeDist}(t_{reco}, t_{true})_i), \end{aligned} \tag{109}$$

where the sum is over a batch of size $N$ and the different elements of the loss function are given by Eq. (84), Eq. (88), Eq. (89) and Eq. (90) .

It was found that RNN-based networks were vastly superior to Attention-based and hybrid architectures - in Figure 40, the validation loss during training of the best performing hybrid and Attention-based networks are compared to an RNN-based network; it is seen that the latter converges to a lower loss value. Hence, the subsequent section is devoted to presenting the optimization search for RNN-based networks.

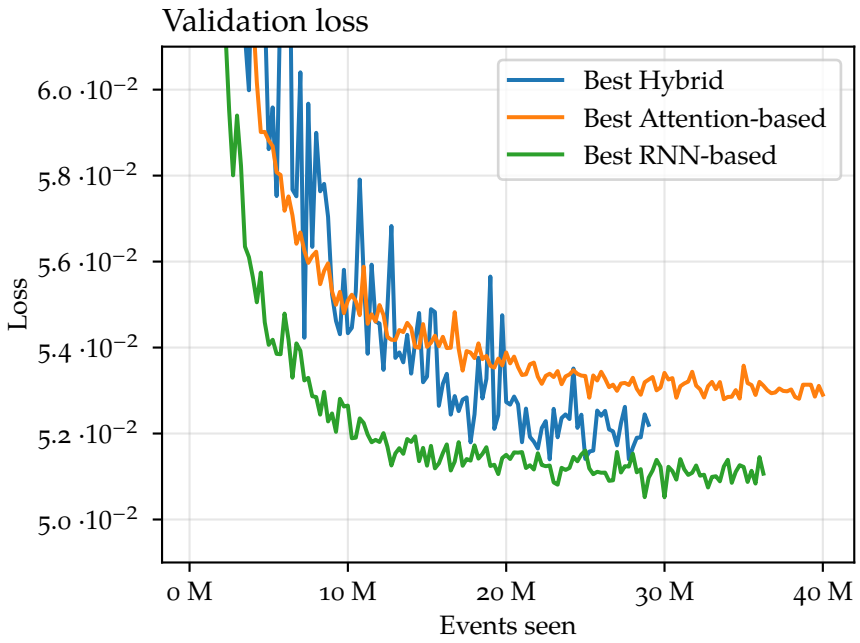



Figure 40: Validation loss during training for the best performing full reconstruction hybrid (blue), Attention-based (orange) and RNN-based (green) networks on muon neutrinos.

### 5.5.2 *Optimization Results*

The most important results of the search are summarized as a parallel coordinate plot in Figure 41, in which each line represents an exper-

iment, where red is better than black; displaying the result of every experiment would clutter the essential takeaway. It is found that

- there are seemingly little (if any) benefit to adding preprocessing layers,
- widths larger than 256 does not increase model performance, and widths of 128 are competetive with wider models.
- RNN-based decoders require atleast a depth of 2 for optimal performance, and there is a minimal (if any) gain in increasing the depth further,
- Networks with Attention-based and Hybrid decoders are severely outperformed by networks with RNN-based encoders,
- decoders with 2 ResBlocks perform best, and the difference in performance between networks with 2-6 blocks is minuscule, and
- the by far most important hyperparameter is encoder depth, where 3-4 stacked RNN-based decoders are most performant.

*Interestingly, state-of-the-art Attention-based networks completely dominate RNN-based networks in the NLP-sector.*

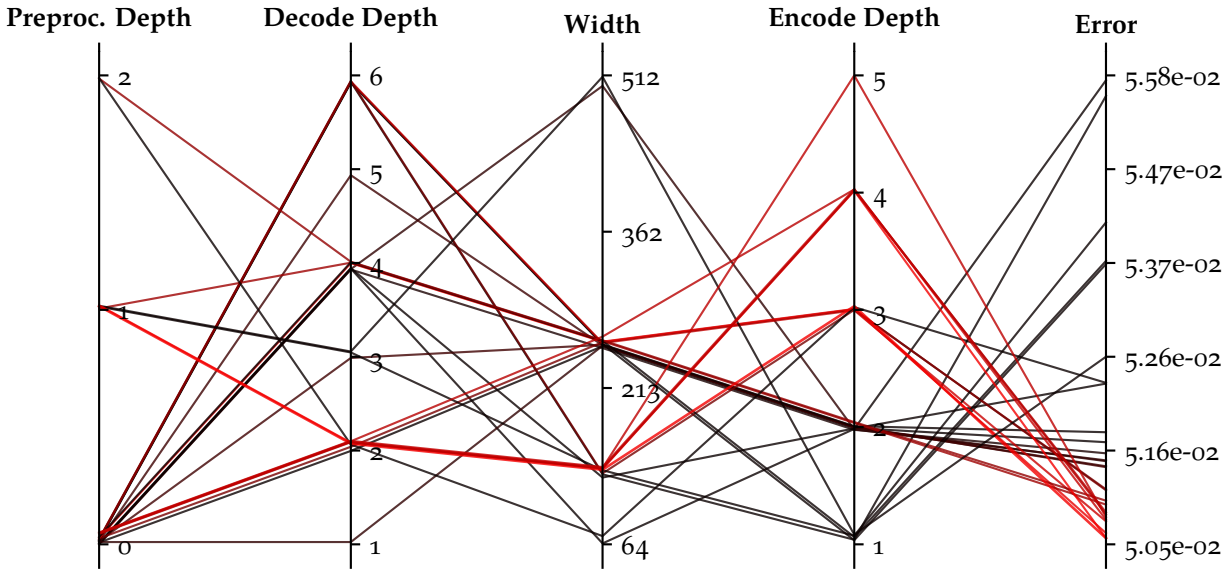



Figure 41: Parallel coordinate representation of RNN-based full reconstruction network performances. Performance is measured by the average loss over 100.000 randomly selected events from the validation set, where smaller (more red) is better.

Having determined a strong core model, the remaining hyperparameter searchspace was explored. The main results are that

- model performance is independent of BS, Dropout and choice of nonlinearity.

- Training networks with Adam consistently led to faster and better convergence compared to both SGD and NAG.
- Bidirectional RNNs outperform standard RNNs, and BiGRUs and BiLSTMs perform equally well, despite BiGRUs having fewer parameters. Both types outperform vanilla RNNs.
- Networks can be made deeper and achieve faster convergence with normalization layers.

In conclusion, the best performing core model is found to be a NN schematically shown in FIGREF with a RNN-based BiGRU-encoder of width 128 and depth 3, one preprocessing ResBlock and a decoder consisting of one ResBlock and a linear layer. A full Pytorch-style summary of the best model is shown in Section A.2.

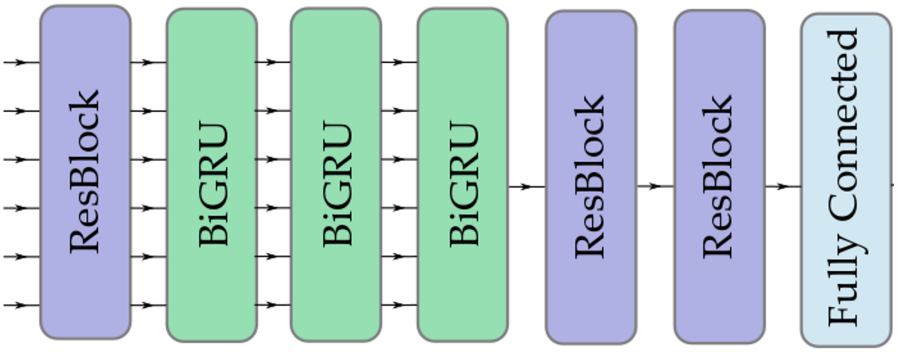



Figure 42: Sketch of the architecture of the best performing core model. The model consists of 1 preprocessing ResBlock, 3 stacked BiGRU-modules, 2 decoding ResBlocks and 1 FF layer without a nonlinearity.

In the next chapter, the performance of the best model is evaluated using the metrics introduced in Section 5.3. On the basis of the hyperparameter search, all models introduced from now on have been trained using

1. Adam,
2. BS of 256,
3. the learning rate schedule described in Section 5.2,
4. no regularization besides EarlyStopping.

Furthermore, attempts at improving the model is made by adding engineered features, using uncleaned events and gathering several NNs in an ensemble of models.

# 6 RECONSTRUCTION PERFORMANCE

Having found the best core model among several candidates, this chapter is devoted to evaluate and enhance its performance. It is attempted to improve performance through the use of different weights, by modifying the used loss function, adding additional input variables and altering the regression tasks. To begin this chapter, the core model's performance on $\nu_\mu$-reconstructions is analyzed.

## 6.1 MUON NEUTRINO RECONSTRUCTIONS

Muon neutrinos are the only neutrinos capable of producing track-like events, which makes them the easiest to detect. Furthermore, the large oscillation probabilities between $\nu_\tau$ and $\nu_\mu$ (see Figure 5), make them especially important. For these reasons (among others) the main focus in this work has been on muon neutrinos.

### 6.1.1 *Reconstructions of Cleaned and Uncleaned Events*

The model described in the end of the last chapter was found using SRT-cleaned events. When using cleaned events, there is a risk that pulses stemming from signal have been discarded, which is undesired. It is, however, not set in stone that the model should not be able to extract all relevant information, when the signal is more noisy. But since the model was found training on cleaned events, it is not guaranteed that the same architecture is optimal for training on uncleaned events; it is likely that a larger NN is required, since the NN has to distinguish signal from noise in addition to the reconstruction.

In Table 5, various models based on the core model described in the previous chapter are compared. Models are labeled as $RNN^{SRT}(d_{encode}, w_{encode}, d_{decode})$ with no superscript, if uncleaned events are used. The performance of each model is summarized in 12 numbers: The performances across the energy intervals $\log_{10} E \in [0,1)$, $\log_{10} E \in [1,2)$ and $\log_{10} E \in [2,3]$ in units of $\left[\frac{E}{\text{GeV}}\right]$, where the performances are measured as the width of the logarithmic energy error distribution (first column), the upper bound of the distance to the IV (second column), the upper bound on the directional error (third column) and finally the width of the error distribution of the time component of the IV.

| Model | $W\left(\log_{10}\left[\frac{E_{pred}}{E_{true}}\right]\right)$ | $U(\lvert\vec{x}_{reco}-\vec{x}_{true}\rvert)$ [m] | $U(\Delta\Psi)$ [deg] | $W(\Delta t)$ [ns] |
|---|---|---|---|---|
| | | $\log_{10} E \in [0,1)$, $\left[\frac{E}{GeV}\right]$ | | |
| $RNN^{SRT}(4,128,2)$ | **0.1413** | **20.77** | **61.66** | 28.86 |
| $RNN^{SRT}(3,128,2)$ | 0.1457 | 20.82 | 61.69 | 28.91 |
| $RNN(3,256,2)$ | 0.1419 | **20.77** | 61.97 | **28.67** |
| $RNN(3,128,2)$ | 0.1415 | 20.86 | 61.87 | 29.3 |
| $RNN(2,256,4)$ | 0.1436 | 20.93 | 62.53 | 28.99 |
| | | $\log_{10} E \in [1,2)$, $\left[\frac{E}{GeV}\right]$ | | |
| $RNN^{SRT}(4,128,2)$ | 0.171 | 33.12 | **38.8** | 56.91 |
| $RNN^{SRT}(3,128,2)$ | 0.1736 | 33.43 | 38.97 | 57.05 |
| $RNN(3,256,2)$ | **0.1658** | **33.01** | 38.89 | **56.06** |
| $RNN(3,128,2)$ | 0.1699 | 33.33 | 39.3 | 57.61 |
| $RNN(2,256,4)$ | 0.1726 | 33.23 | 39.12 | 57.69 |
| | | $\log_{10} E \in [2,3)$, $\left[\frac{E}{GeV}\right]$ | | |
| $RNN^{SRT}(4,128,2)$ | 0.2646 | 36.05 | 25.36 | 54.9 |
| $RNN^{SRT}(3,128,2)$ | 0.2658 | 36.8 | 25.49 | 54.14 |
| $RNN(3,256,2)$ | **0.2462** | 35.31 | **24.56** | **52.22** |
| $RNN(3,128,2)$ | 0.2496 | 35.87 | 25.28 | 55.95 |
| $RNN(2,256,4)$ | 0.2518 | **34.84** | 24.69 | 54.93 |

Table 5: Summary of performances of models taking cleaned (superscripted *SRT*) and uncleaned (no superscript) sequential inputs. Performances are calculated using Eq. (76) and Eq. (81) and are reported as the average performance (lower is better) over 3 different energy intervals: $\log_{10} E \in [0,1)$, $\log_{10} E \in [1,2)$ and $\log_{10} E \in [2,3]$ in units of $\left[\frac{E}{GeV}\right]$. The best performances are reported in bold.

From the performances it is seen that the difference between *RNN*- and $RNN^{SRT}$ is close to negligible at lower energies ($E < 10$ GeV); the differences between the best performing *RNN*- and $RNN^{SRT}$ models is in most cases less than 1 %. A different picture is painted at high energy ($E > 100$ GeV): In this range, *RNN*-models persistently outperform $RNN^{SRT}$-models with up to 5 % increase in performance. Despite the fact that the sequences are significantly longer for uncleaned events (see Figure 28), the *RNN*-models are able to extract the valuable information despite the noisy signal, which likely is due to the SRT-cleaning removing signal pulses.

As mentioned in Section 2.6, the current likelihood-based reconstruction approach may only achieve inference speeds (i.e. reconstructions per second) of $O(10^{-1})$. The NN-based models developed in this work achieve significantly higher inference speeds: In Table 6, the inference speed, number of parameters and size of some of the best performing models are compared.

| | $RNN^{SRT}(4,128,2)$ | $RNN^{SRT}(3,128,2)$ | $RNN(3,256,2)$ | $RNN(3,128,2)$ |
|---|---|---|---|---|
| Inference Speed | $7233s^{-1}$ | $8047s^{-1}$ | $6338s^{-1}$ | $7325s^{-1}$ |
| Parameters | 1.18 M | 0.89 M | 3.40 M | 0.89 M |
| Size | 4.73 MB | 3.55 MB | 13.61 MB | 3.55 MB |

Table 6: Inference speeds (events reconstructed per second) and sizes (in terms of memory requirements and number of parameters) of $RNN(...)$ and $RNN^{SRT}(...)$ models. Inference was run with a batchsize of 512 on 1 NVIDIA GeForce GTX 1080 GPU.

As shown, inference speeds above 6000 reconstructions per second can consistently be achieved for the best performing models. If the batchsize is increased further, even higher inference speeds can be achieved. The inference speed of all displayed model's have been achieved using 1 NVIDIA GeForce GTX 1080 GPU [90]. Comparing the number of parameters of the models $RNN(3,256,2)$ and $RNN(3,128,2)$ with their performances displayed in Table 5, we see that only a minor improvement in terms of reconstruction performance is achieved, when the number of model parameters is increased from 900.000 to roughly 3.500.000.

*The NVIDIA GeForce GTX 1080 is a consumer-grade graphics card sitting at a retail price of approximately 800 $ in Denmark*

### 6.1.2 *Full Reconstruction Performance*

In Figure 43, the performance of full reconstructions using $RNN(3,256,2)$ (blue) are compared to the performance of Retro's reconstructions (orange). On every plot, the gray histogram with a secondary, logarithmic y-axis on the right displays the amount of events in the training set for the given energy range. Additionally, the entire error distributions as a function of energy are shown in Figure 44 for polar angle and energy reconstructions, since these specific reconstructions exhibit biases, which are discussed below. In Section A.5, additional performance plots are shown.

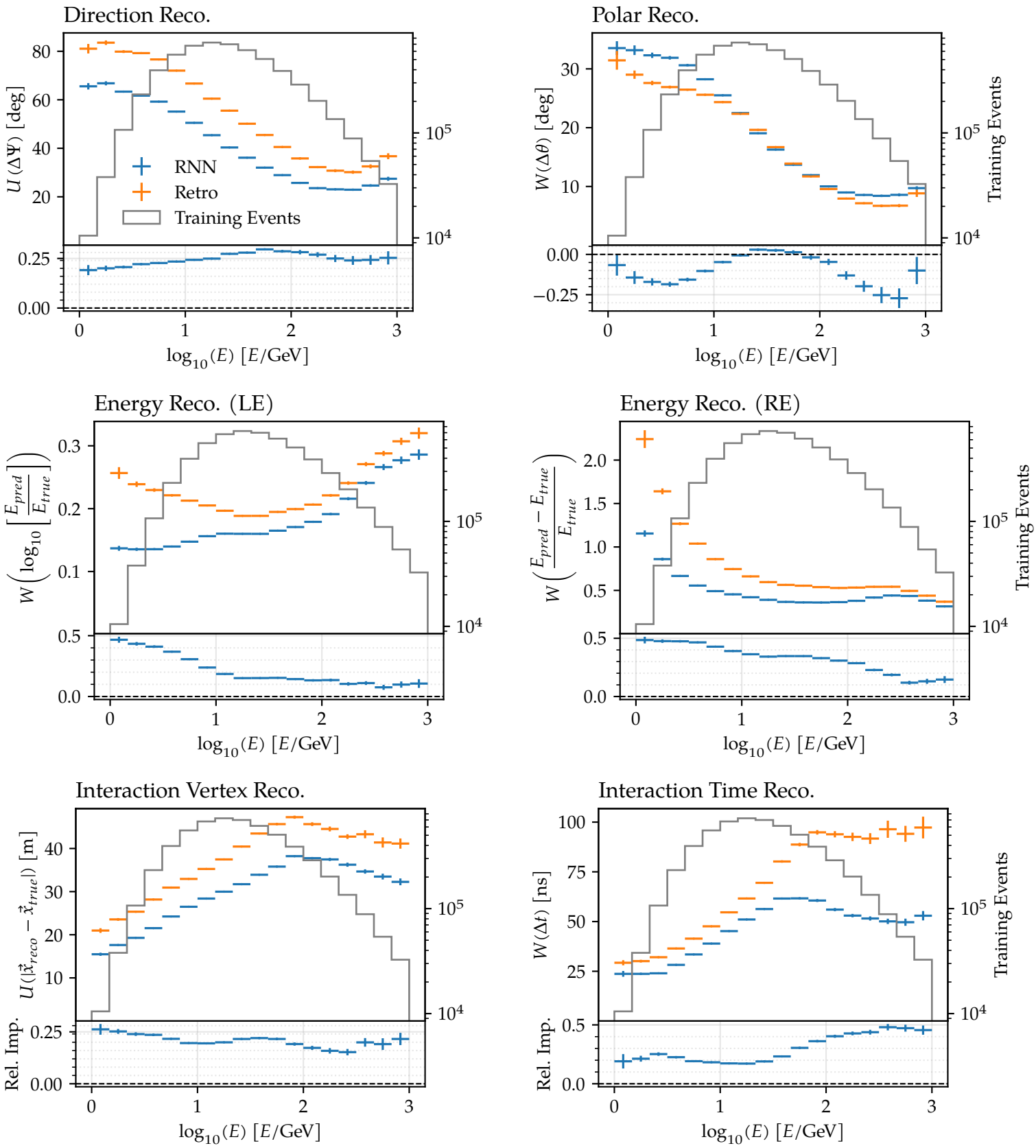
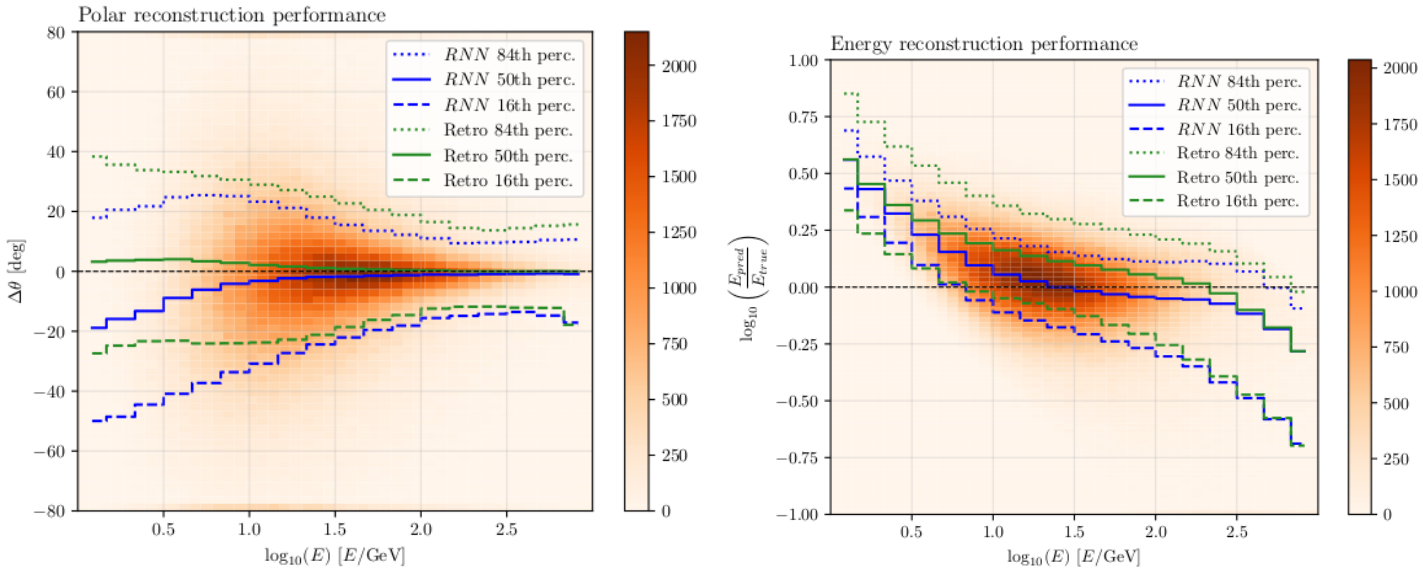



Figure 43: Performance as a function of energy for full reconstructions with $RNN(3, 256, 2)$ (blue) compared to Retro's reconstructions (orange). Performances are parametrized as the width $W$ (Eq. (76)) or upper bound $U$ (Eq. (81)) of the relevant error distribution. Uncertainties are calculated using Eq. (80) and Eq. (82) respectively.

In the top-left of Figure 43, the direction reconstruction performance is shown, where the angular difference $\Delta\Psi$ between the reconstructed and true directions is given by Eq. (86). As shown, *RNN* outperforms the Retro reconstructions across the entire energy range with 20 % or larger improvements with respect to the upper bound. Furthermore, the resolution steadily increases with energy for both *RNN* and Retro reconstructions up until $\log_{10} E \approx 2.5$, where the resolution begins to decrease again. It is currently not known why Retro direction reconstruction performance decreases above a certain threshold, but it has been conjectured to stem from the fact that high-energy events are less likely to be contained in the densely instrumented DeepCore. It should be noted as well that the error is calculated with respect to the $\nu_\mu$-direction, which is not aligned with the direction of the muon that leaves a track (see Eq. (22)). The reconstruction of the actual track is therefore better than what is suggested by the performance plot.

The direction reconstruction performance can also be probed by inspecting the distributions of the polar and azimuthal angle errors. As previously discussed, this approach has the disadvantage of being coordinate dependent, but since the polar angle is important for oscillation analyses, it is considered as well.

The polar reconstruction performance is shown in the top-right of Figure 43: In this case, Retro reconstructions generally outperform *RNN*, but with approximately equal performance in the energy range $\log_{10} E \in [10, 100]$. As previously mentioned, the error distribution of the polar reconstruction is shown in Figure 44a from which it is seen that *RNN* is biased towards predicting too small polar angles, especially at low energies. The reason behind this bias is likely to be the following: Since most neutrinos are going upwards (corresponding to low $\theta$-values), when the NN is faced with a sequence with little information (such as a low energy event), it defaults to predicting an upwards-going neutrino, since such a tactic reduces the loss the most - it can therefore very likely be removed by reweighting the data to be uniformly distributed on a sphere. Increasing the performance of polar angle reconstructions is discused further in Section 6.3.




(a) (b)

Figure 44: Error distributions as a function of energy for the polar reconstruction (left) and energy reconstruction (right) using the logarithmic error metric. The 16th, 50th and 84th percentiles of the error distributions are shown for *RNN* (red) and Retro (green) reconstructions.

Performance with respect to energy reconstruction is probed in two different ways: In the middle-left plot of Figure 43 the LE (given by Eq. (84)) is used and in the middle-right plot of Figure 43 the RE (given by Eq. (83)) is used. *RNN* performs much better compared to Retro for both performance metrics, despite the fact that the used loss function given by Eq. (109) is optimized to reduce the LE.

With respect to the LE as performance metric, *RNN* is consistently at least 10 % better than Retro. Interestingly, the performance for both *RNN* and Retro decreases as a function of energy, when the LE is used as metric, whereas the performance tends to increase, when the RE is used as metric. The kink in performance at $\log E \approx 2.5$ in the relative error plot of Figure 43 is likely due to a small amount of data at high energies compared to the rest of the spectrum. From the error distribution shown in Figure 44b it is seen that both Retro and *RNN* tend to assign a too large energy to low-energy events and a too low energy to high-energy events. In the case of Retro reconstructions, predicting too low energies begins at $\log E \approx 1.5$, which is punished harder, when a LE metric is used (see Figure 35). Additionally, *RNN* is most heavily biased at low energies despite the fact that the performance plots suggests best performance in this region. This can in part be attributed to the fact that *RNN* was trained on relatively few low-energy events, and the used weights, $w_{blinding}$, skews the distribution even further towards higher

*The kink in energy reconstruction performance with respect to the relative error of the Retro algorithm is likely due to the fact that the likelihood-model assumes energy loss purely due to ionization, whereas radiative losses becomes significant at energies above 100 GeV*

energies. Increasing the performance and removing the bias at low energies is discussed in Section 6.3.

Finally, the reconstructed IV errors are shown in the bottom-left and -right plots of Figure 43. The performance of both the spatial and temporal *RNN* reconstructions outperform Retro across the entire energy range. Strangely, for IV reconstructions, performance decreases up until $\log E \approx 80$ GeV, whereafter it increases as a function of energy. As previously mentioned, the cause of such performance shapes is unknown, but since *RNN* and Retro tend to agree on the shapes, it is plausible that it is a feature of the dataset.

### 6.1.3 *Feature Engineering*

A phrase commonly heard in ML communities is that a model is only as good as the data it is trained on. It might therefore be beneficial in terms of performance to enrich the input sequences with information, which the NNs might have difficulties extracting. Therefore models taking additional sequential and scalar features have been tested. The added features can be separated into two categories: Features extracted directly from the sequences, including simple estimates of the neutrino direction and signal speed from DOM $i$ to DOM $i+1$ and features containing information about nearest neighbors. The idea behind adding the latter is that there might be valuable information hidden in the part of the detector that did *not* trigger. The added scalar features are

1. **LineFit**[91]: A rough and fast estimate of the particle direction parametrized as $x$-, $y$- and $z$-components of a unit vector. The particle trajectory is found by minimizing the Huber loss[92] of the distances $\Delta d$ between the trajectory and the triggered DOMs given by

$$\Delta d_i = |\vec{v} \cdot (t_i - t_0) - (\vec{r}_i - \vec{r}_0)|, \tag{110}$$

   where $\vec{r}_i$ and $t_i$ are the $i$'th DOM coordinates and triggertime and $\vec{v}$, $\vec{r}_0$ and $t_0$ are fitted.

2. **Tensor of Inertia** (ToI)[93]: An additional rough and fast estimate of the particle direction. The particle direction is reported as the principal axis of the ToI with the smallest eigenvalue, which approximates the direction of the particle. The ToI is calculated in the classical way, where the charge of each triggered DOM acts as a virtual mass. Additionally, the ratio

$$\frac{\lambda_{min}}{\sum_i \lambda_i} \tag{111}$$

   is calculated, where $\lambda_{min}$ is the smallest eigenvalue. The smaller this ratio is, the more track-like the event is.

In addition to the above-mentioned scalar features, the following features are added to each sequence element

1. **DOM_charge_significance**: A measure of how significant the energy deposited to a single DOM is. Calculated as

$$\texttt{DOM\_charge\_significance} = \frac{N_{DOMs} \cdot \texttt{DOM\_charge}}{\sum \texttt{DOM\_charge}} \tag{112}$$

2. **DOM_d_prev**: Euclidian distance to the previous DOM in sequence.

3. **DOM_v_from_prev**: Signal speed from the previous DOM to current. Calculated as

$$\texttt{DOM\_v\_from\_prev} = \frac{\texttt{DOM\_d\_prev}}{\texttt{DOM\_time}_i - \texttt{DOM\_time}_{i-1}} \tag{113}$$

4. **DOM_Minkowski_prev**: The Minkowski distance to the previous DOM in sequence. The Minkowski distance can be used to determine whether there is a causal relationship between triggers. It is calculated with metric signature $(+, -, -, -)$.

5. **DOM_d_closest**: The Euclidian distance to the closest DOM in the sequence.

6. **DOM_Minkowski_closest**: The Minkowski distance to the closest DOM in the sequence.

7. **DOM_closest1**: The $x$, $y$, $z$, $t$ and charge values of the *nearest neighbor*, i.e. a DOM that potentially did not trigger.

8. **DOM_closest2**: The $x$, $y$, $z$, $t$ and charge values of the *second nearest neighbor*.

Due to the nature of the detector, the nearest neighbors are always on the same string. If information on DOMs that did not trigger proves to be useful, the neighborhood could be extended to include DOMs on different strings.

To investigate whether the added features increase performance, the previously described $RNN(3, 256, 2)$ is compared to an an identical model and an almost-identical model, which only differs by taking the features described above as additional inputs. The comparison with a retrained identical model is included to get an idea of the inherent variance in performance due to the stochastic nature of training NNs. The relative improvements in performance over the baseline-model in Figure 43 are shown in Figure 45.

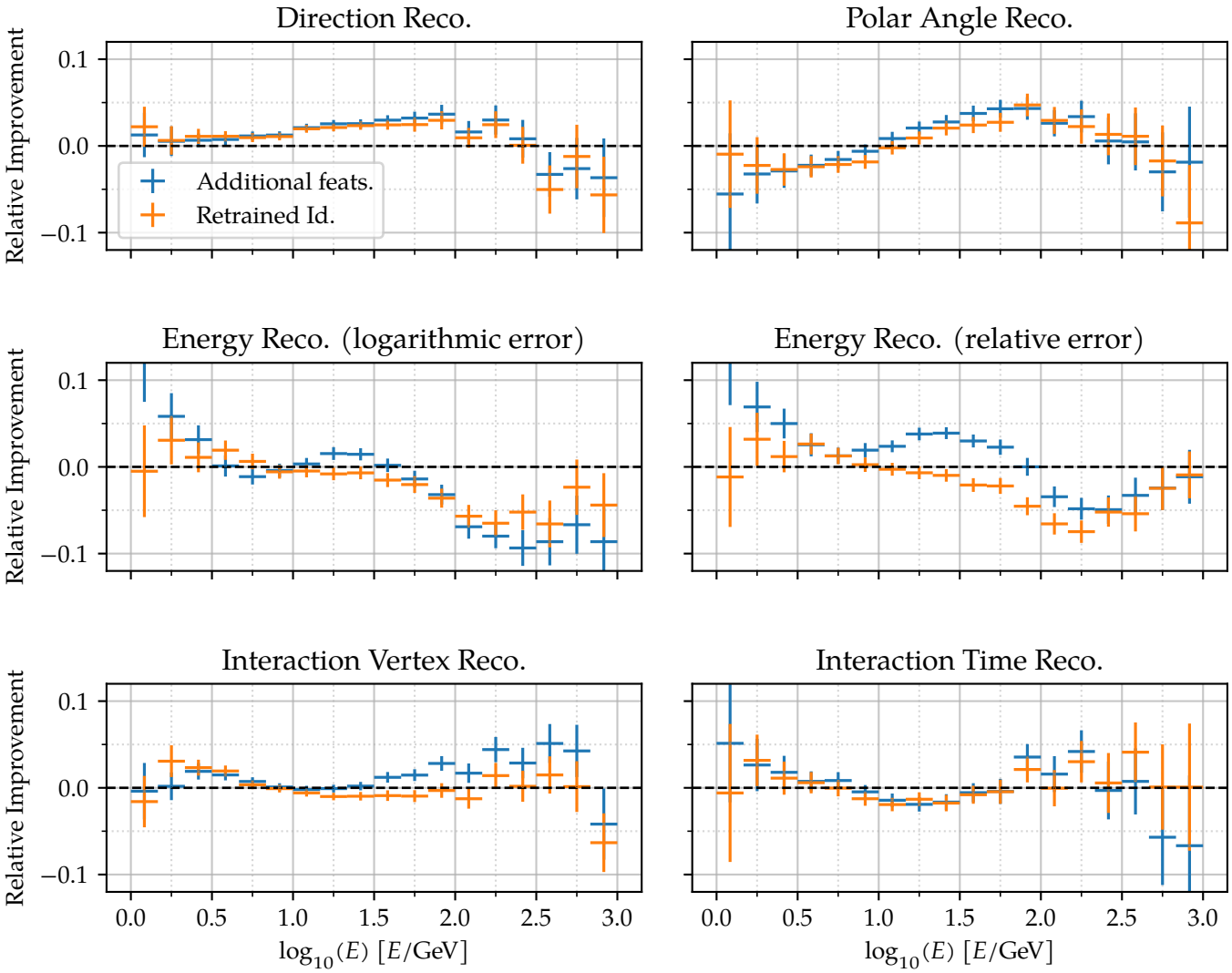



Figure 45: Relative improvements in performance of $RNN(3, 256, 2)$ when taking the engineered features described above as additional inputs (blue) compared to an identical $RNN(3, 256, 2)$ only taking the features described in Table 3 (orange). The deviation from equal performance of the identical model is due to the inherent variance of training NNs. The relative improvement is measured in terms of the performance measures used in Figure 43

It is difficult to determine whether or not an overall increase in performance is achieved; most performance measures of the model taking additional inputs see both increases and decreases at $< 5$ % in performance as a function of energy. Likewise, the identical model see similar increases and decreases in performance, which stems from the inherent variance in training. The difference in performance is therefore unlikely to be attributable to the added features but merely statistical fluctuations. Since some of the variance is explained by the stochastic nature of training, it is difficult to quantify how big the decrease or

increase in performance is. Bayesian attempts at quantifying variance due to training can be carried out using Dropout [94], but due to time limitation it has not been done in this work.

### 6.1.4 *Feature importance*

NNs are notoriously known for being black box models. It is now attempted to partially open the black box and glimpse into the inner workings of the developed models through the use of Permutation Feature Importance (PFI) [95]. PFI is a method that can be used to gauge the importances of the different features to the model. Let the model performance be described by a number $P$, which for e.g. the energy reconstruction is the width of the logarithmic error distribution shown in the middle-left plot of Figure 43. The PFI is then defined to be

$$\text{PFI}(x) = \frac{P_{Permuted} - P}{P}, \tag{114}$$

where $P_{Permuted}$ is the model performance, when the feature $x$ has been replaced by a randomly chosen value from the empirical distribution of $x$. In other words, it is the fractional decrease in performance, when the information from a certain feature is removed. The hypothesis is then that the larger the decrease in performance, the more important the feature is to the model. In the specific case of RNNs, the PFI of a sequential feature $x_{seq}$ such as `DOM_charge` is attained by replacing its value in every sequence element by randomly chosen values from the entire distribution of `DOM_charge` .

*PFI was first used on random forest classifiers with tabular data, where the column with feature x was permuted - hence the name.*

In figure Figure 46 the 24 most important PFI scores are shown for direction, polar angle, energy, azimuthal angle, interaction vertex and time reconstructions for the $RNN(3, 256, 2)$-model taking additional inputs. Each PFI is calculated on the basis of 100.000 randomly selected events. Generally, the by far most important features for all reconstructions are the non-engineered features, likely due to the fact that these features carry all the information. Interestingly, the features carrying information on what parts of the detector that did *not* trigger achieve high FPI scores across all reconstructions, despite the fact that the performances are unlikely to have increased because of them as previously seen. Furthermore, all engineered features are compatible with having PFI scores of 0 at 5 % significance level, meaning they are not important at all.

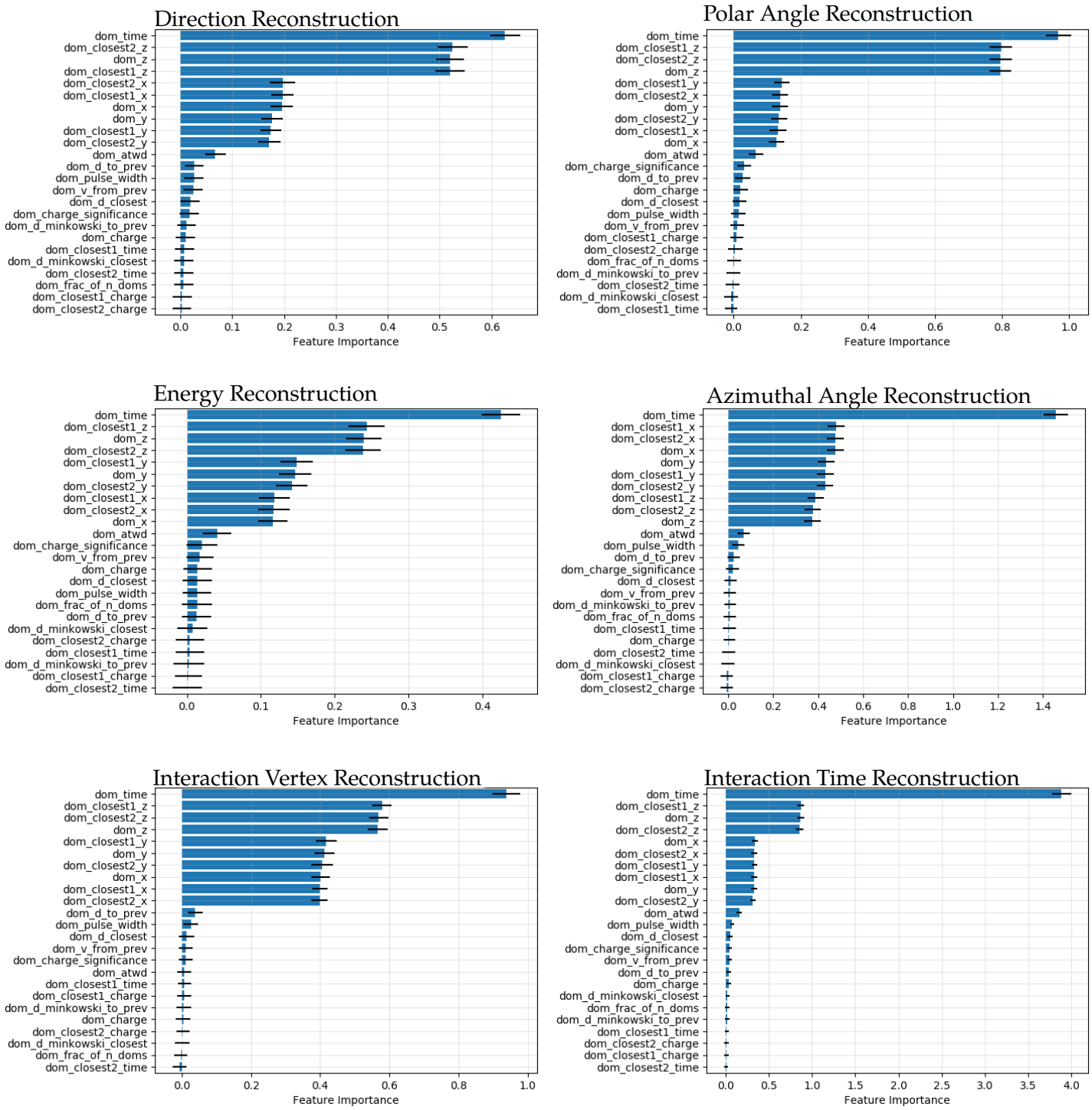



Figure 46: Permutation feature importance for direction, polar angle, energy, azimuthal angle, interaction vertex and time reconstructions for $RNN(3, 256, 2)$. The 24 most important features are shown. Higher values means a feature is more important. Note the different scales in each plot.

Perhaps surprisingly, `DOM_charge` is among the features compatible with having a PFI score of 0 for all reconstructions. This result might be suggesting that at GeV-energies, each triggered DOM registers too few photons to convey meaningful information about the particle track. Since the input sequences only consist of triggered DOMs, the triggering of each DOM essentially only carries one bit of information (besides its position and triggertime): Triggered or not triggered.

For the direction reconstruction, the spatiotemporal features are the most important, with the time- and z-components scoring highest. This is in agreement with the fact that the inherent resolution of the detector is highest with respect to the time- and z-components of the pulses. For polar angle reconstruction, the discrepancy between the importance of the time- and z-components and the x- and y-components is even larger, which is expected; the azimuthal component only depends on the $x-$ and $y-$components of the direction vector (hence these features achieve a higher score for the azimuthal reconstruction than the $z-$component) and the polar component only depends on the $z-$component.

The energy reconstruction also depend on all spatiotemporal components, all achieving relatively low PFI scores between 0.1 and 0.5, but with the $t-$ and $z-$components being the most important. Compared to the interaction time reconstruction, which predictably depends strongly on `DOM_time`, it is not clear a priori which feature the energy reconstruction should depend mostly on. It can, however, be said that it turns out to depend the most on the features with the highest resolution.

## 6.2 ELECTRON-, MUON- AND TAU RECONSTRUCTION

As discussed in Section 2.5, different neutrino-types have different signatures. It is therefore not unthinkable that different models should be used for different types of neutrinos. Such an approach, however, requires the experimenter to be able to distinguish between the events, and therefore classifiers must be developed. Such a classifier has been developed and is introduced in the following section. Afterwards, it is investigated whether a single model reconstructing all neutrinos can compete with 3 different models, each handling its own type of neutrino

### 6.2.1 *Track and Cascade Classification*

In Section 1.2.1, it was discussed how neutrinos interact via neutral or charged currents (NC and CC). Any NC event will produce a cascade response giving the experimenter no way of distinguishing between neutrino types. Furthermore, $\nu_e$- and $\nu_\tau$-events at low energies produce cascade-responses as well, and only CC $\nu_\mu$-events produce track-like responses. Traditionally, events have therefore been classified as either cascades or tracks; hence this approach is taken in this work.

The classifier is trained using the (by now well-known) $RNN(3, 256, 2)$ architecture and the cross entropy loss function given by Eq. (94). The model is trained on a subsample of the entire OscNext-sample summarized in Table 2. Approximately 90 % of $\nu_\mu$-events were CC-events leading to a samplesize of $5.2 \cdot 10^6$ CC $\nu_\mu$-events, and all $\nu_e$-events ($1.6 \cdot 10^6$) were used. Since $\nu_e$-events are guaranteed to produce cascades, and $\nu_\tau$-events can produce track-like responses as well (due to the decay of the $\tau$ into a $\mu$), $\nu_\tau$-events are not included in the sample. Weights are used to create a balanced training set with an average weight of 1 such that no bias is introduced, i.e. $\nu_e$-events are given the weight

$$w_{\nu_e} = \frac{N_{\nu_e} + N_{\nu_\mu}}{2N_{\nu_e}} \tag{115}$$

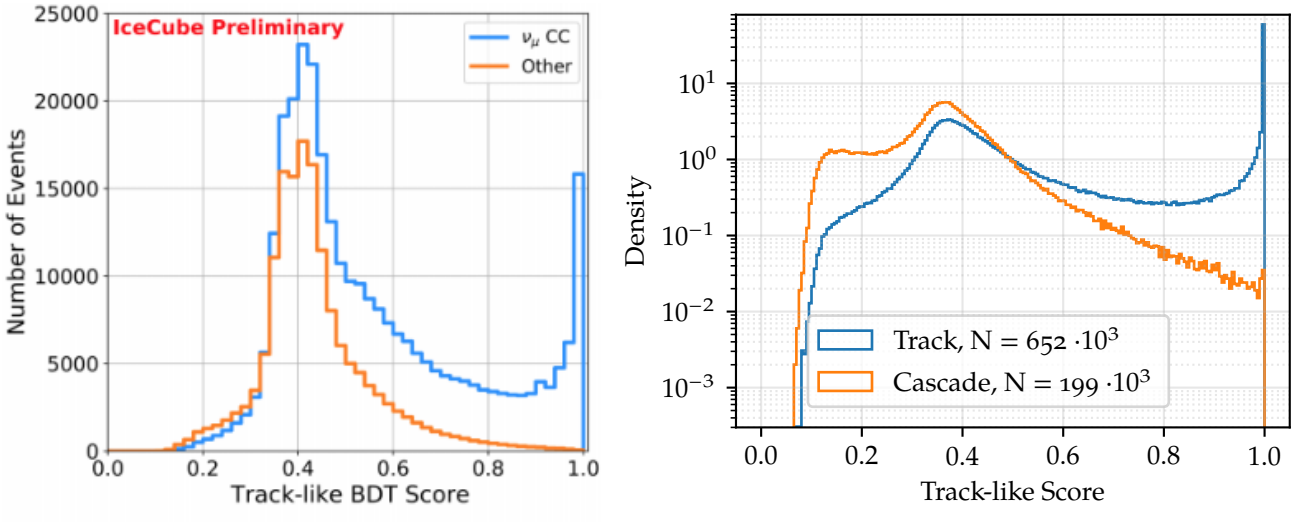
and $\nu_\mu$-events are given the weight

$$w_{\nu_\mu} = \frac{N_{\nu_e} + N_{\nu_\mu}}{2N_{\nu_\mu}}. \tag{116}$$

*Unfortunately, it has not been possible to compare classification scores based on predictions on the same sample nor retrieve the data shown for the BDT-classifier*

In Figure 47 the distribution of the predicted track-like scores on the held-out testset are shown (right) along with the distribution of track-like scores for the currently used BDT-classifier (left) generated on a different sample of CC muon neutrinos and electron neutrinos - one should therefore be cautious about making definite claims about which classifier that performs best. It should be noted that the BDT-scores are not normalized and are shown on a linear scale, whereas the NN-scores are normalized and shown on a logarithmic scale.

Despite these differences, the distributions follow the same pattern. A noteworthy difference between the distributions is the track-peak heights at a classification score of 1, where the NN peak is is relatively larger than its BDT counterpart. Based on the apparent looks of the distributions, the NN-based classifier appears to outperform the BDT. To make a conclusive determination, however, a data-based comparison should be conducted.




(a) (b)

Figure 47: Track-like classifier scores for the currently used BDT-based classifier (left, courtesy of Tom Stuttard) and the developed NN-based classifier (right). The classifiers are evaluated on different datasets. Additionally, the BDT-scores are not normalized and are shown on a linear y-scale, whereas the NN-scores are shown on a logarithmic scale and the distributions are normalized.

The ROC curves for the NN-based classifier for different energy ranges are shown in Figure 48 along with their corresponding AUC-values. Each curve displays a characteristic steep rise in TPR with a more or less constant FPR. Unsuprisingly, the performance of the classifier increases with energy. For the highest energies, AUC-scores of 0.95+ are achieved, and an almost completely clean CC $\nu_\mu$-sample can be extracted while only discarding approximately 10% signal.

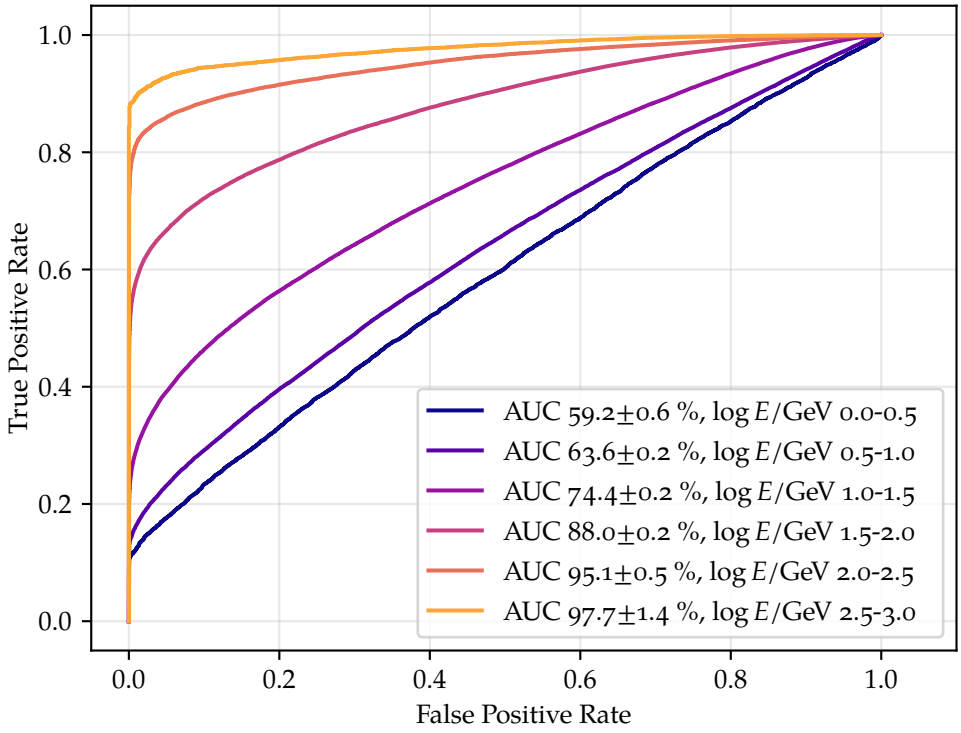



Figure 48: ROC curves of the NN-based classifier at different energy ranges along with their corresponding AUC-values.

At lower energies, it is much more difficult to distinguish between tracks and cascades; AUC-scores are below 0.65 for events with energies $< 100$ GeV. It might therefore be necessary to build a model capable of reconstructing all neutrino flavours in one go.

### 6.2.2 *Reconstruction with a Single Model*

As shown above, it is difficult at low energy to efficiently generate clean samples of either type of neutrino, and therefore one might have to resort to reconstructing any neutrino with the same model.

The focus in this work has been on optimizing the performance with respect to reconstructions of muon neutrinos. However, to get an idea of how well electron- and tau neutrinos can be reconstructed, an $RNN(3, 256, 2)$-model has been trained and evaluated on the entire dataset summarized in Table 2 for a total of $8.8 \cdot 10^6$ training events, i.e. a model reconstructing all flavours of neutrinos. This model is compared to the performance of 3 different models: A $RNN(3, 256, 2)$-model only reconstructing electron neutrinos, a $RNN(3, 256, 2)$-model only reconstructing muon neutrinos, and finally a $RNN(3, 256, 2)$-model only reconstructing tau neutrinos.

Training of the abovementioned models was carried out using the previously described logcosh-loss given by Eq. (109) and the previ-

ously described weights $w_{balanced}^{\alpha=0.7}$. In Figure 49, the performance of the above-mentioned models are shown with the all-neutrino model in blue, the single-flavour models in orange and the Retro reconstruction in green for selected reconstruction variables: The energy reconstruction performance is shown in the left column and the polar angle reconstruction performance is shown in the right column. In the top row, $\nu_e$-performances are shown, in the middle row $\nu_\mu$-performances and finally $\nu_\tau$-performances are shown in the bottom row.

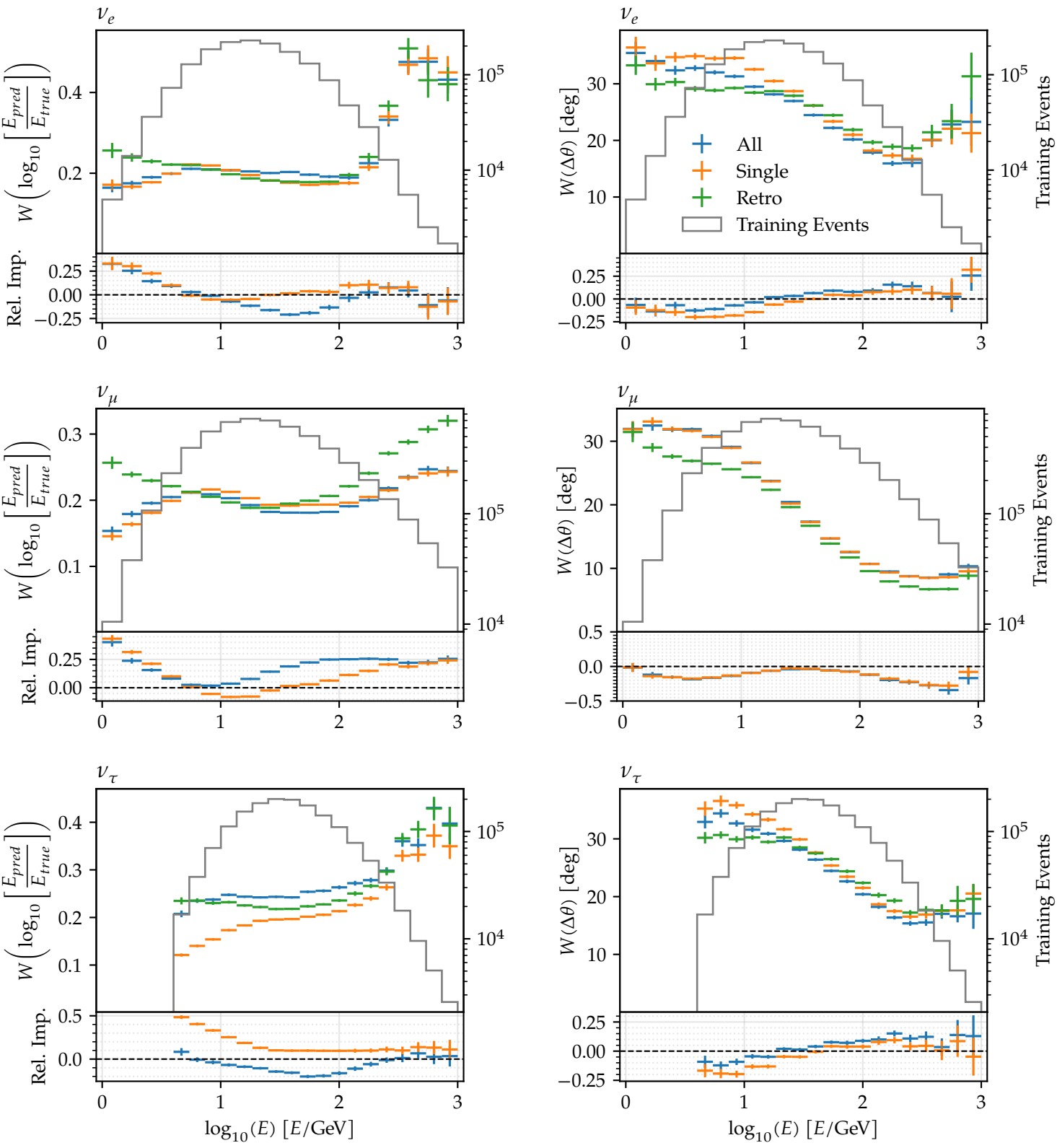



Figure 49: Energy (left) and polar angle (right) reconstruction performances of a single model reconstructing all neutrinos (blue) compared to neutrino-specific reconstruction models (orange) and the performance of the Retro reconstruction (green).

Interestingly, for the polar reconstruction the all-neutrino model outperforms the single-flavour models. For $\nu_e$- and $\nu_\tau$-reconstructions, this effect is likely due to the fact that both event-types primarily produce cascade-like responses, and therefore reconstructing both neutrinos at once corresponds to doubling the amount of data. Nonetheless, the same effect is not seen with respect to the energy reconstruction: In this scenario, the $\nu_\mu$-reconstructions benefit from reconstructing all flavours at once in the mid energy-range, while the $\nu_e$-reconstructions become worse for the same range.

Perhaps the biggest deterioration is seen for the $\nu_\tau$ energy reconstructions. At low energy, this is expected, since there is a significant lower bound on the $\nu_\tau$-energy due to the $\tau$-mass, which a single-flavour model learns. At higher energies, it is more unclear why the single-flavour model is performing better.

## 6.3 SPECIALIZED RECONSTRUCTION MODELS

When one model is built to reconstruct all parameters of interest (energy, direction, interaction vertex and interaction time), the used loss function potentially has to be tuned to put all contributions on an equal footing. For instance, there is no a priori guarantee that the energy- and direction-components in Eq. (109) contribute equally, and there is therefore a possibility that the model will only achieve optimal performance with respect to one of the regression tasks.

The correlation coefficients between the different reconstruction errors of $RNN(3, 256, 2)$ trained with the logcosh-loss function given by Eq. (109) is displayed in Figure 50. As shown, some components correlate more strongly with the loss value indicative of a larger overall contribution to the loss: The IV reconstruction is seen to be the dominant contributor, which might be at the cost of performance w.r.t. polar angle reconstruction, as this error correlates less strongly with the loss-value.

*If the performance with respect to one parameter is 'orthogonal' with respect to the performance of another parameter, the described phenomenon will not be present.*

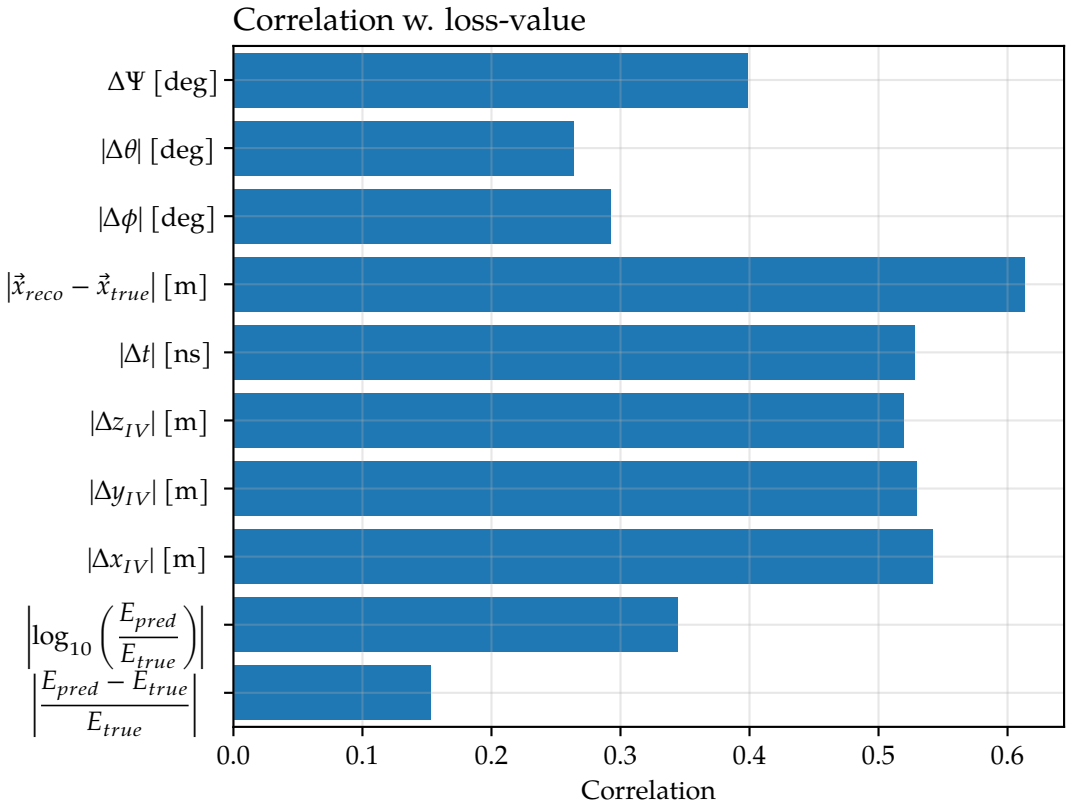



Figure 50: Correlation coefficients between different performance measures and the loss value for $RNN(3, 256, 2)$ trained using the loss function given in Eq. (109).$x_{IV}, y_{IV}$ and $z_{IV}$ are the cartesian components of the interaction vertex.

Since some regression tasks might be suppressed due to the loss function construction, it has been investigated whether different models should be used for different regression tasks. By splitting Eq. (109) into 3 separate loss functions given by

$$L_{Energy} = \frac{1}{N}\sum_{i=1}^{N} \text{logcosh}(\text{LE}(E_{reco}, E_{true})_i), \tag{117}$$

$$L_{IV} = \frac{1}{N}\sum_{i=1}^{N} \text{logcosh}(\text{EuclidDist}(\boldsymbol{r}_{\text{reco}}, \boldsymbol{r}_{\text{true}})_i), \text{ and} \tag{118}$$

$$L_{Direction} = \frac{1}{N}\sum_{i=1}^{N} \text{logcosh}(\text{VertexDist}(\boldsymbol{p}_{reco}, \boldsymbol{p}_{true})_i), \tag{119}$$

and training and evaluating 3 different models (labelled $RNN_{Energy}(3, 256, 2)$, $RNN_{IV}(3, 256, 2)$ and $RNN_{Direction}(3, 256, 2)$) based on each loss function, whether or not performance is lost is deduced.

The relative improvement in performance w.r.t. polar angle (upper left), energy (upper right), IV (lower left) and interaction time (lower right) reconstructions, when the models $RNN_{Energy}(3, 256, 2)$,

$RNN_{IV}(3, 256, 2)$ and $RNN_{Direction}(3, 256, 2)$ perform the regressions compared to the single model $RNN_{all}(3, 256, 2)$ performing all regression tasks is shown in Figure 51 (azimuthal and direction reconstruction performance improvements are shown in Section A.3). Several of each model were trained to ensure that performance changes were not merely due to statistical fluctuations; hence, representative model performances are shown.

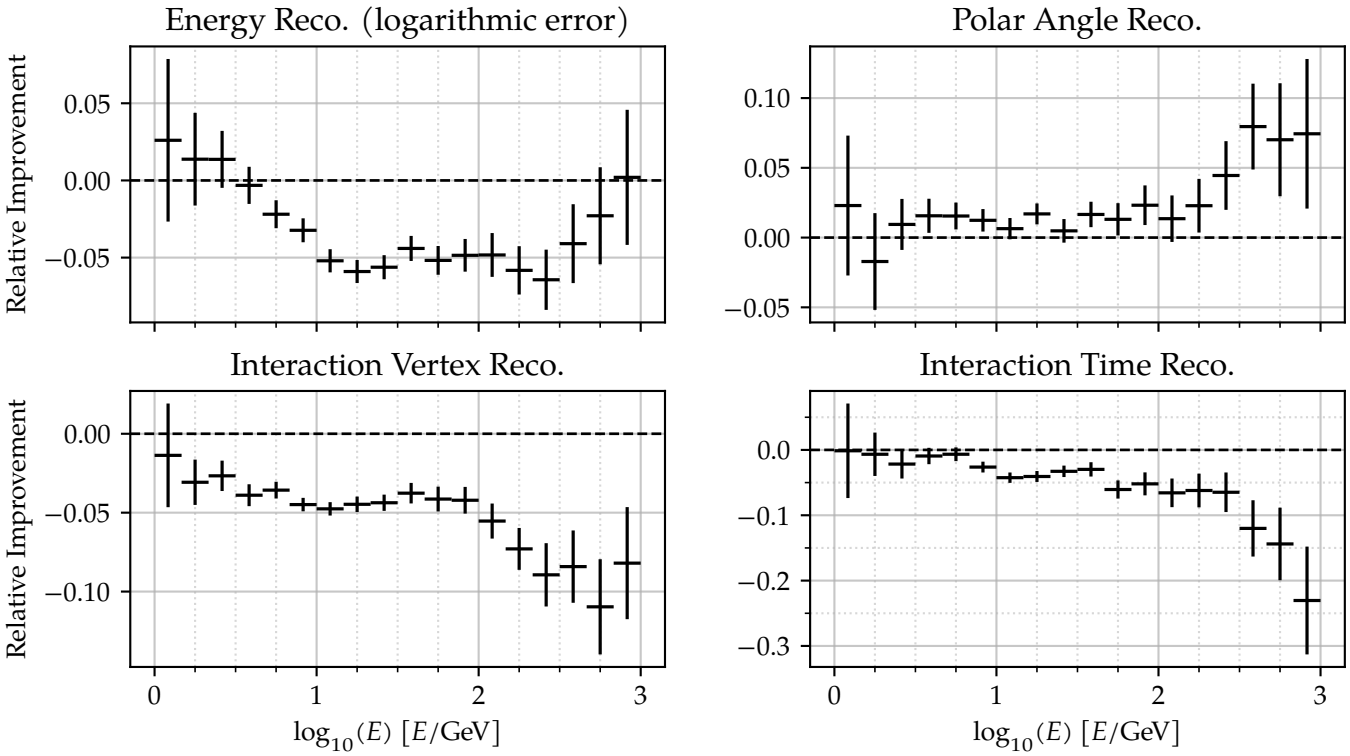



Figure 51: Relative improvements in performance w.r.t. $RNN_{all}(3, 256, 2)$, when the regression tasks are divided among 3 models. All architectures are identical, but the loss functions have been altered to Eqs. (117 - 119). As can be seen, the directional reconstruction benefits from being handled by a separate model, $RNN_{Direction}(3, 256, 2)$.

Performing all regression tasks with a single model appears to have a regularizing effect with respect to the IV reconstruction: $RNN_{IV}(3, 256, 2)$ performs worse than $RNN_{all}(3, 256, 2)$across the entire energy range. On the opposite, the direction reconstruction appears to benefit from splitting up the regression tasks. This effect is most prominent at higher energies. The next section is therefore devoted to optimizing the direction reconstruction by tuning the used loss function.

### 6.3.1 *Tuning the Direction Loss*

As described in Section 5.3.3, other error measures have been investigated to optimize the direction reconstruction, specifically the angular difference directly given by Eq. (86) and the negative cosine similarity given by Eq. (87). Both error measures introduce difficulties in each their way.

The angular difference has the interesting feature that as the predictor gets better (i.e. $\cos x$ approaches 1), the gradients become larger and larger: Since the derivative of the inverse is given by

$$\frac{d}{dx} \arccos x = -\frac{1}{1-x^2}, \tag{120}$$

the derivative diverges, when the predictor makes perfect predictions. Needless to say, this unwanted feature makes it very difficult to optimize NN.

The negative cosine similarity introduces a different problem. Since the NN is not guaranteed to predict a vector of unit length, it will occasionally produce predictions of length 0 or close to it, making the loss function unstable. This can be partially resolved by altering the loss function to

$$\text{NegSim}(\boldsymbol{r}_{\text{reco}}, \boldsymbol{r}_{\text{true}}) = 1 - \frac{\boldsymbol{r}_{\text{reco}} \cdot \boldsymbol{r}_{\text{true}}}{|\boldsymbol{r}_{\text{reco}}||\boldsymbol{r}_{\text{true}}| + \epsilon}, \tag{121}$$

where $\epsilon > 0$ is a small number, but it does not solve the problem completely; if $\epsilon$ is made too large, the NN can simply reduce its loss value by predicting short vectors, but if it is not large enough, short-length vectors can generate large gradients making the optimization process unstable. This issue can partially be alleviated through gradient clipping [96], but the process is messy. Experiments with all error measures mentioned in Section 5.3.3 have been carried out, but due to the above-mentioned issues, a loss function incorporating the Euclidian distance between the direction vectors (Eq. (88)) gave the best results.

When the unit vector is predicted by a NN, the network learns that target values above 1 or below -1 are not possible. This fact causes networks to too seldom predict an $x-$, $y-$ or $z-$component close to one of the bounding values. This effect is strengthened further, when a convex loss function with respect to the errors is used. This effect is due to the fact that in the face of uncertainty, the NN can minimize the loss by predicting a value close to the mean of the target distribution, ultimately resulting in too short predictions as shown in the right plot of Figure 52. In the left plot of Figure 52, the distribution of the $x-$component of the direction vector is shown to be strongly affected by the used loss function, where the described effect is strongest for the logcosh-loss (blue) due to its stronger convexity compared to a regular L1-loss (orange).

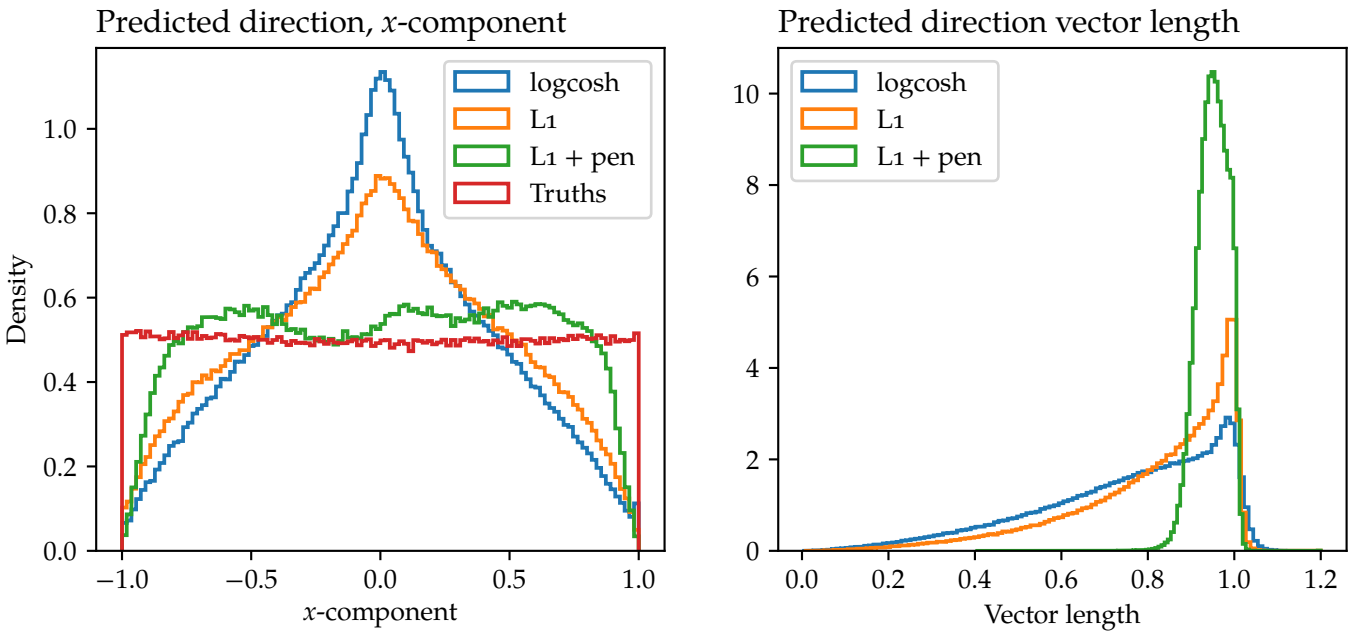



Figure 52: Left: Distribution of the $x-$component of the direction vector $\boldsymbol{r}_{\text{reco}}$ predicted by $RNN_{Direction}(3, 256, 2)$, when EuclidDist$(\boldsymbol{r}_{\text{reco}}, \boldsymbol{r}_{\text{true}}))$ is input to a logcosh (blue), L1 (orange) or L1 with a penalty term (green) given by Eq. (122). The true distribution is shown in red. Right: The distribution of the lengths of the predicted direction vectors.

The problem is solved to a certain extent by adding a penalty term to the L1-loss, transforming the loss to

$$l(\boldsymbol{r}_{\text{reco}}, \boldsymbol{r}_{\text{true}})) = L1(\boldsymbol{r}_{\text{reco}}, \boldsymbol{r}_{\text{true}})) + \alpha \cdot (1 - |\boldsymbol{r}_{\text{reco}}|)^2, \tag{122}$$

where $\alpha$ is a tunable parameter. When $RNN_{Direction}(3, 256, 2)$ is trained with this loss (here with $\alpha = 1$), more correct distributions (orange) of the components are achieved as shown in Figure 52. The distributions of the predicted $y-$ and $z-$components with the different loss functions are shown in Section A.4.

As mentioned, adding a penalty term to the L1-loss does not completely solve the problem of too few predictions with component-values close to -1 and 1. In the end, this might not be as big of an issue as it could seem, since it is the *relative* sizes of the components with respect to each other that determines the direction. Therefore, if the absolute values of all the components generally are too small, the individual errors might neutralize each other. Using this line of argument, adding a penalty term could be futile, but there are two reasons why that is not the case. First and foremost, it turns out that the penalty term leads to better performance at low energies with respect to the polar angle reconstruction (as we will see later). Secondly, achieving high accuracy with respect to the individual direction components becomes important,

when estimates of the errors on the azimuthal and polar angles are generated using probabilistic loss functions.

## 6.4 ENSEMBLE MODELS

The high variance in single model performance means that different models produce different predictions. It is therefore possible to combine several identical (with respect to architecture) models into an *ensemble* of models utilizing the strength of each individual submodel[97]. In this section, the description of a $\nu_\mu$-reconstructing ensemble is described.

The more uncorrelated the used models are, the better performance can be achieved. In this work, the used submodels are therefore trained with different weights and loss functions, and the models are combined using another NN (called a *meta-learner*). The performance of $RNN_{all}(3, 256, 2)$is attributed to its superior ability to extract information. Hence, an identical architecture is used for the meta-learner. The general ensemble architecture used is sketched in Figure 53: $N$ different models make reconstructions, where the performance of each model is controlled via the used loss-function and used weights. Then a meta-learner outputs a final reconstruction based on the event and the individual reconstructions

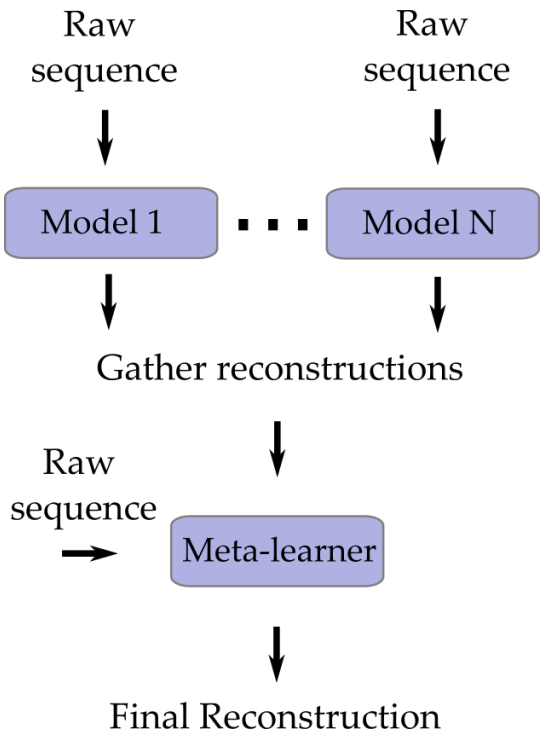


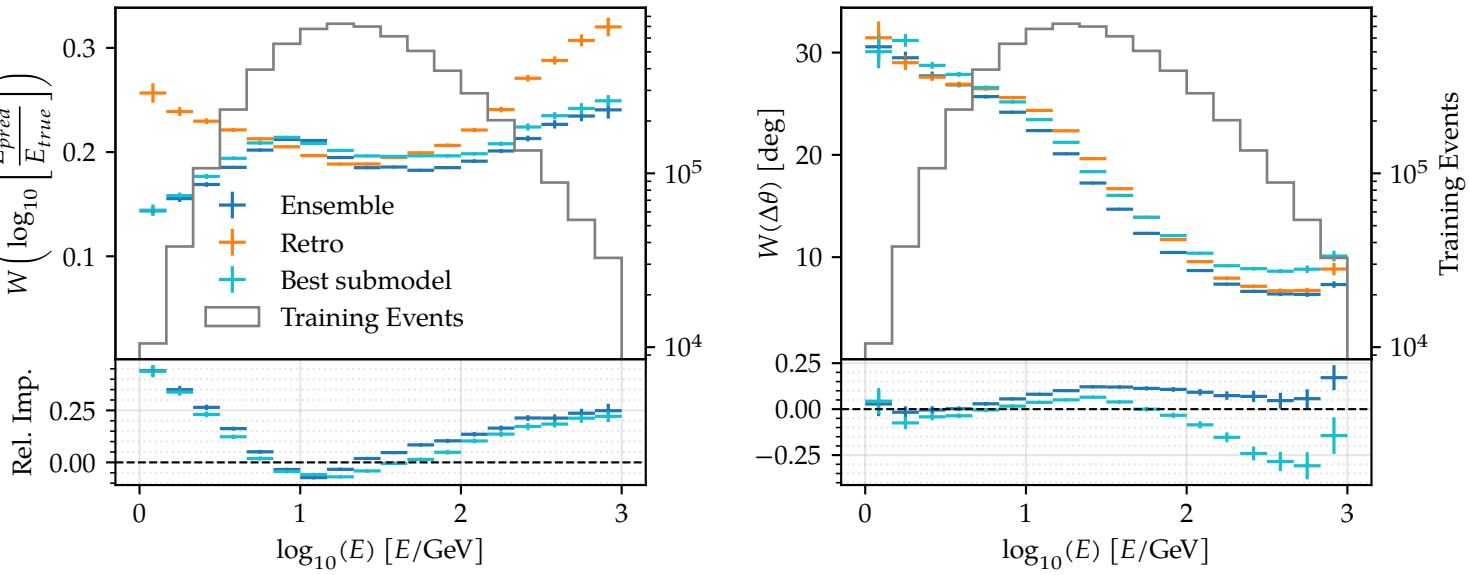

Figure 53: Sketch of ensemble architecture. $N$ models predict a set of target variables, whereafter a meta-learner combines the individual predictions into a final reconstruction. In addition to the $N$ reconstructions, the meta-learner is also fed the event.

By using point-predicting models (as opposed to models predicting a distribution) for the submodels and a distribution-predicting meta-learner, an error estimate is generated as well. In this work, errors are assumed to be Gaussian, and are generated by letting the NN approximate a likelihood-function, i.e. the used loss function is the negative log-likelihood given by Eq. (93). In accordance with the discussion in Section 6.3, two different ensembles have been developed: One energy- and IV-reconstructing ensemble and one direction-reconstructing ensemble. In Table 7, the used submodels in each case are summarized: Each ensemble consists of 8 submodels (4 unique) trained with different weights and loss functions.

| Energy + IV | | | Direction | | |
|---|---|---|---|---|---|
| Loss | Weights | $N$ | Loss | Weights | $N$ |
| logcosh | $w_{blinding}$ | 2 | logcosh | $w_{blinding}$ | 2 |
| logcosh | $w^{\alpha=0.7}_{balanced}$ | 2 | L1+ penalty | $w_{blinding}$ | 2 |
| logcosh | $w_{low}$ | 2 | L1+ penalty | $w_{Direction}$ | 2 |
| logcosh | $w_{high}$ | 2 | L1 | $w_{Direction}$ | 2 |
| $N_{total}$ | | 8 | | | 8 |

Table 7: Summary of the training procedures of the submodels in the energy + IV-reconstructing ensemble and the direction-reconstructing ensemble: The used event-weights, the used loss function and the number of identical models in the ensembles. All submodels have the $RNN(3, 256, 2)$-architecture. $w_{Direction}$ reweighs the distribution to be uniform on the unit sphere.

The weights and loss functions are described in Section 5.4. The final meta-learners were in both cases trained using $w^{\alpha=0.7}_{balanced}$ and were evaluated on the held-out test set. For the direction-reconstructing ensemble, the regularizing penalty term given by Eq. (122) was added to the loss function. The performances of the ensembles are shown for the polar angle reconstructions (right) and the energy reconstructions in Figure 54 - the additional reconstruction performances are shown in Section A.6.




(a) (b)

Figure 54: Ensemble model performances (blue) compared to the Retro reconstruction performance (orange) and the best performing submodel of the ensemble (cyan) evaluated on the held-out testset. The performances of the energy reconstructions (left) and polar angle reconstructions (right) are shown along with the relative improvements compared to Retro.

As shown, the ensemble models perform similarly or better across the entire energy range for both reconstructions. For the energy reconstructions, more than a 20 % increase in performance is seen and the polar angle reconstructions attain increases in performance between 5 % and 10% for for the majority of the energy range. Furthermore, the performance is seen to increase, when several models are gathered together in an ensemble.

If Figure 53a is compared to the middle-left plot of Figure 43, at first glance the performance would appear to have decreased. This is however an artefact of the fact that performance is measured as a width of a distribution not taking potential bias into account. As shown in Figure 44b, the width of the logarithmic error distribution at low energy is small, but heavily biased. The ensemble trained with a different set of weights more or less completely removes this bias in the range, where the vast majority of data is as shown in Figure 55.

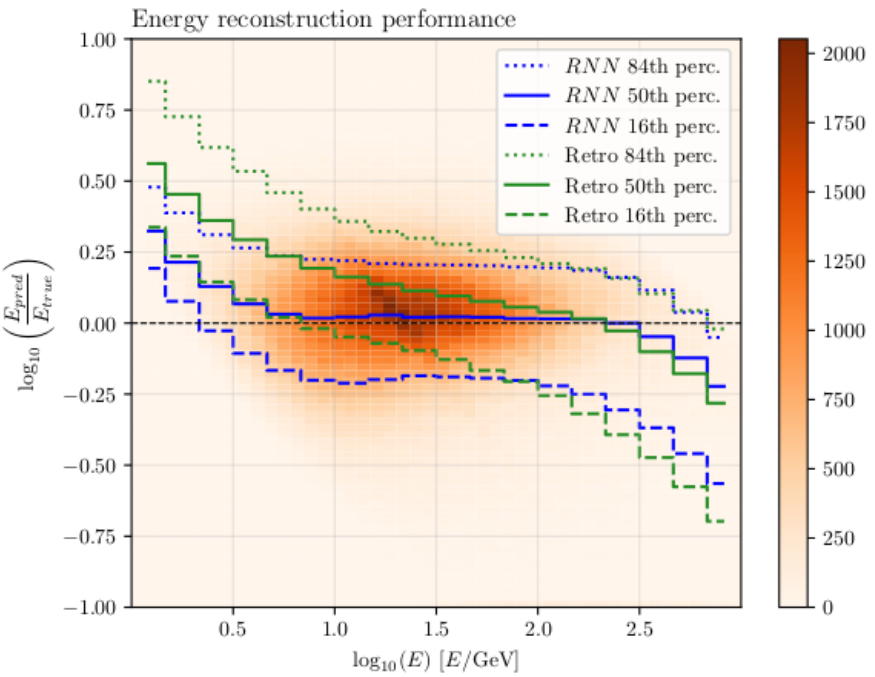



Figure 55: Logarithmic error distribution of ensemble-model (blue) compared to the logarithmic error distribution of the Retro reconstructions (green).

Being able to produce point-estimates of the energy and the direction that outperform the existing state-of-the-art reconstruction algorithm is an accomplishment in itself. Additionally, the ensemble models also produce sensible error estimates, which the Retro algorithm to the author's knowledge is not capable of.

As previously mentioned, the estimated errors are forced to be Gaussian. If some random variable $X \sim \mathcal{N}(\mu, \sigma^2)$, then the transformed variable

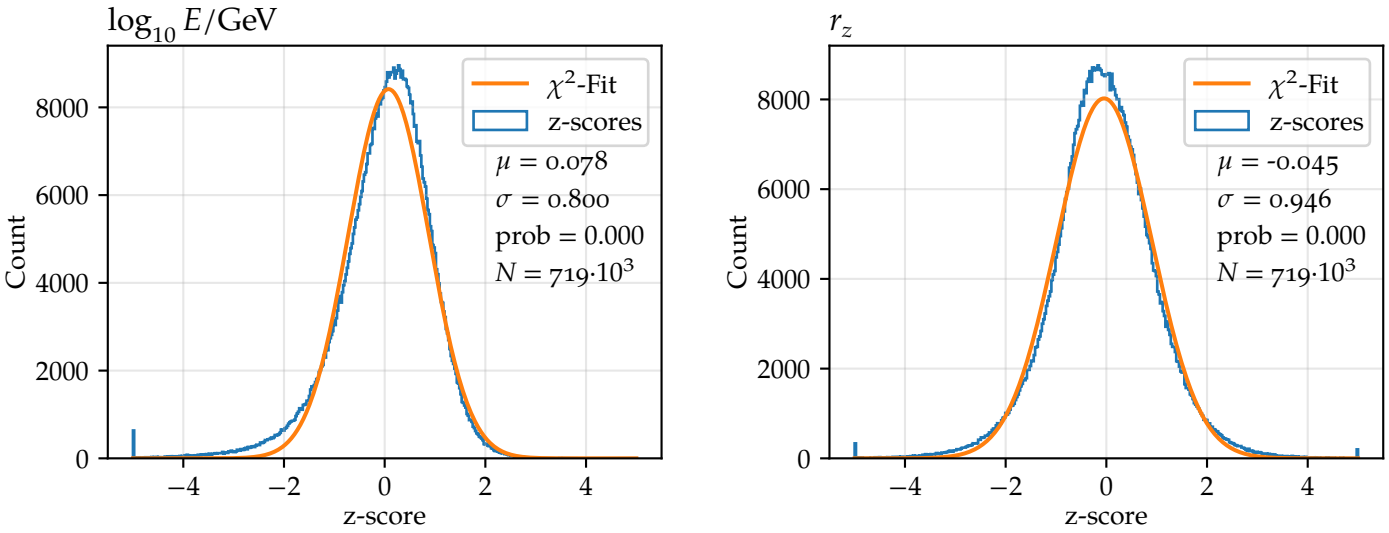
$$z = \frac{X - \mu}{\sigma} \tag{123}$$

will be distributed as

$$z \sim \mathcal{N}(0, 1), \tag{124}$$

where $z$ is known as the $z$-score. In other words, if the individual point- and error-estimates $\mu_i$, $\sigma_i$ are correct, applying the transformation given by Eq. (123) and histogramming the values should produce a unit Gauss. The z-score distributions for the logarithm of energy (left) and z-component $r_z$ of the direction vector (right) are shown in Figure 56 along with Gaussian fits to each distribution. Z-score distributions for the remaining components of the direction vector along with the IV reconstruction are all similar and are shown in Section A.7. Neither of the distributions are unit Gaussians: Both widths are too small indicative of an overestimation of the errors, the tails are too long (too

many 2+ $\sigma$-events) and both distributions are slightly skewed. Judging by the looks of the distributions, however, they are reasonably close to unit Gaussians meaning the error estimates are sensible




(a) (b)

Figure 56: Z-score distributions for the reconstructed logarithm of the energy (left) and z-component of the direction vector (right) along with a Gaussian $\chi^2$-fit to each. The first and last bins are overflow bins.

Depending on the wanted level of accuracy (and the size of the errors), various methods of estimating the error on *functions* of e.g. the reconstructed logarithm of the energy or the direction vector (such as the energy or the polar angle) exist such as calculation through the law of error propagation or through simulation.

# 7 CONCLUSION & OUTLOOK

In the past 6 chapters, a small fraction of the fascinating world of neutrino physics and deep learning has been investigated. Algorithms based on Recurrent Neural Network building blocks such as the Gated Recurrent Unit have been employed to answer 3 important questions about the neutrinos in the Icecube detector: What were you? Where did you come from? And how energetic were you? In this short chapter, the main results presented in this work are summarized before an outline of where to go next is presented

## 7.1 CONCLUSION

Several reconstruction algorithms have been developed capable of classifying and reconstructing $\nu_e-$, $\nu_\mu-$, and $\nu_\tau-$ interaction vertices, interaction times, energies and directions. When reconstructions are carried out on an out-of-date consumer-grade GPU (NVIDIA GeForce GTX 1080), inference speeds of 5000+ events per second were reached; a significant upgrade in itself over the current state-of-the-art low energy reconstruction algorithm Retro only reaching speeds on the order of $O(10^{-2})$ events per second.

A deep learning-based classifier has been developed to classify events into track- or cascade-like signal. The classifier appears to outperform the currently used BDT-classifier, achieving AUC-scores ranging from $59.2\% - 97.7\%$ for the lowest and highest energies in the GeV-range respectively.

The best performing $\nu_\mu$-reconstruction model developed in this work is an ensemble model consisting of 8 different $3M$-parameter submodels. The model achieves energy and polar angle reconstruction performances more than 10 % better than the Retro algorithm across the majority of the GeV energy range.

In addition to producing better reconstructions faster, the developed ensemble model is capable of producing reasonable error estimates: For instance, the z-score distribution of the logarithm of the reconstructed energies are approximately distributed as $z_{\log E} \sim \mathcal{N}(0.08, 0.80^2)$.

## 7.2 OUTLOOK

A natural question to ask at this point must be: Can we do even better? The answer is likely 'yes'. Since neural network performance tends to increase when more data is added, it is highly probable that additional performance increases can be achieved. In Figure 57 the validation

loss is shown for $RNN_{all}(3, 256, 2)$when the amount of training data is varied. As can be seen, the loss-value upon convergence decreases as a function of the train set size, and there is no reason to believe that this trend should not continue (and performance is positively correlated to the loss value upon convergence)

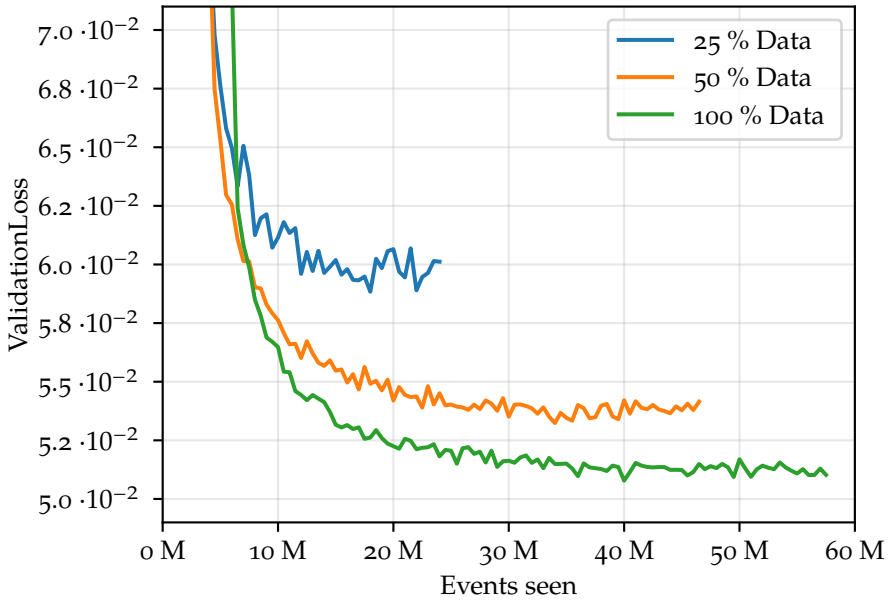



Figure 57: Validation loss during training for $RNN_{all}(3, 256, 2)$when using 100% (green), 50% (orange) and 25% (blue) of the available training data.

Since the training data stems from simulation, it is probably fairly easy to further improve the performance of the developed models; simply simulate more data.

It is one thing to perform well on simulated data, a feat that might not be reproducable on real events. To ensure that the developed models are able to accurately reconstruct true events, some form of validation is therefore desirable. The Icecube detector has previously been used to detect the cosmic ray shadow of the Moon [98], which is an approach that can be used to determine whether the direction reconstructions of the models are sensible. If the RNN-models also are superior to the traditional likelihood-based approach on real data, the obtained resolution on the shadow of the Moon should be better. To determine whether energy reconstructions on real data are sensible, a different approach must be taken such as matching the reconstructed energy spectrum to the expected spectrum.

As discussed in Section 2.1.3, the vast majority of detected events is background. Due to the high noise rates of the DOMs, several triggers are even caused purely by noise and with the upcoming upgrade to Icecube [99], noise rates are expected to increase even more. Hence,

there might be a NN-based classification algorithm yet to be discovered capable of outperforming the currently used BDT-approach for background and noise detection.

Finally, to the author's knowledge there is currently no working reconstruction algorithm ready for when Icecube Upgrade is implemented - and there is no apparent reason as to why a NN should not be able to produce even better reconstructions, when the detector resolution is increased.

# A

# APPENDIX

## A.1 TARGET- AND INPUT DISTRIBUTIONS

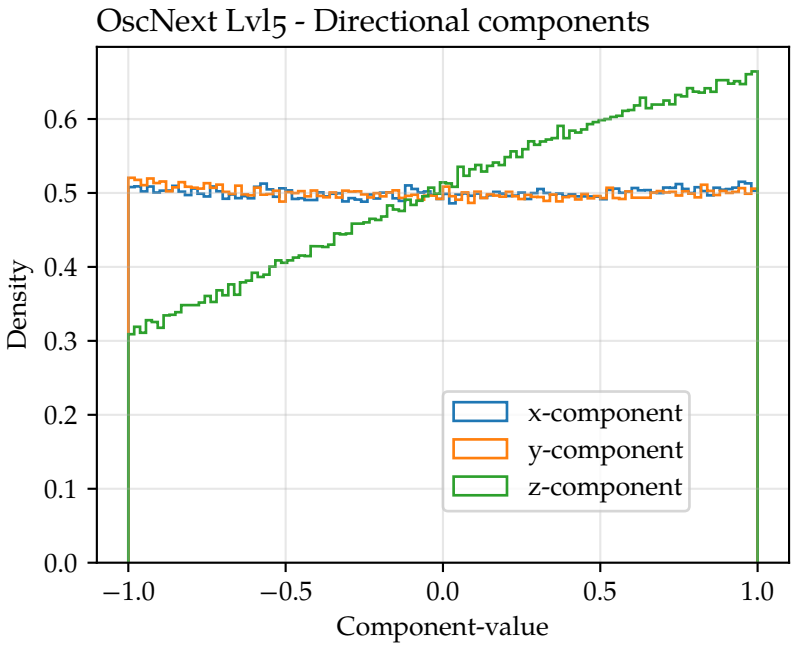



Figure 58: Distributions of $x-$, $y-$ and $z$-components of the neutrino directions.

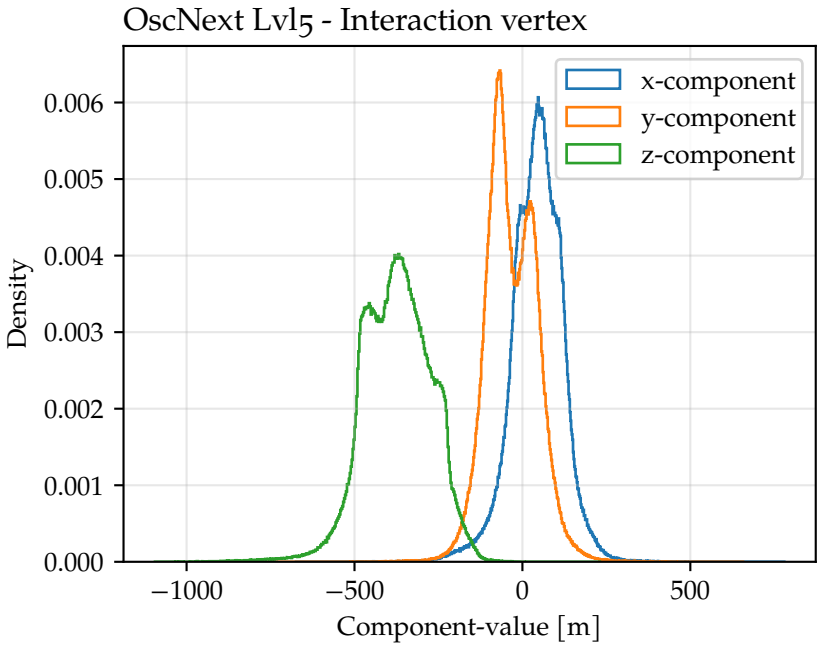



Figure 59: Distributions of $x-$, $y-$ and $z$-components of the neutrino interaction vertices.

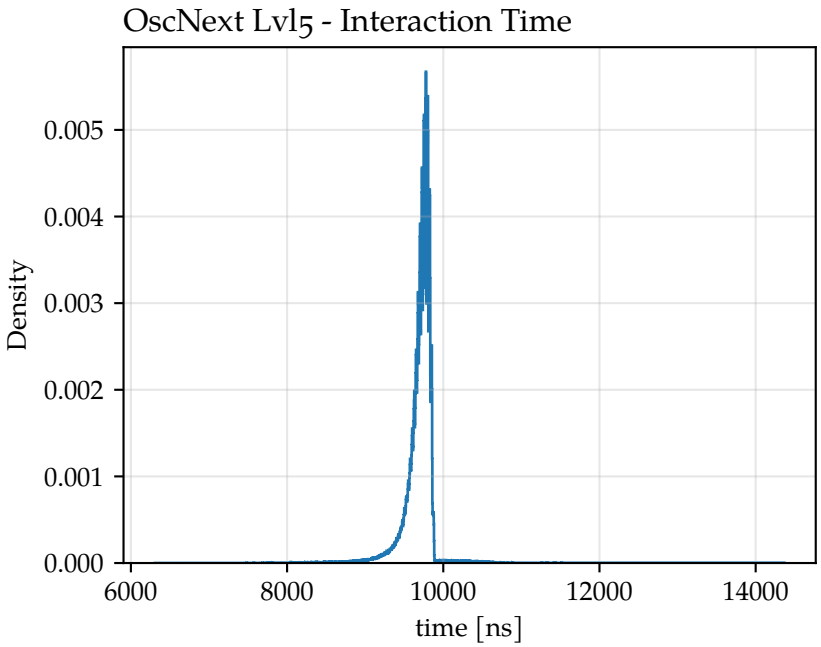



Figure 60: Distribution of interaction time.

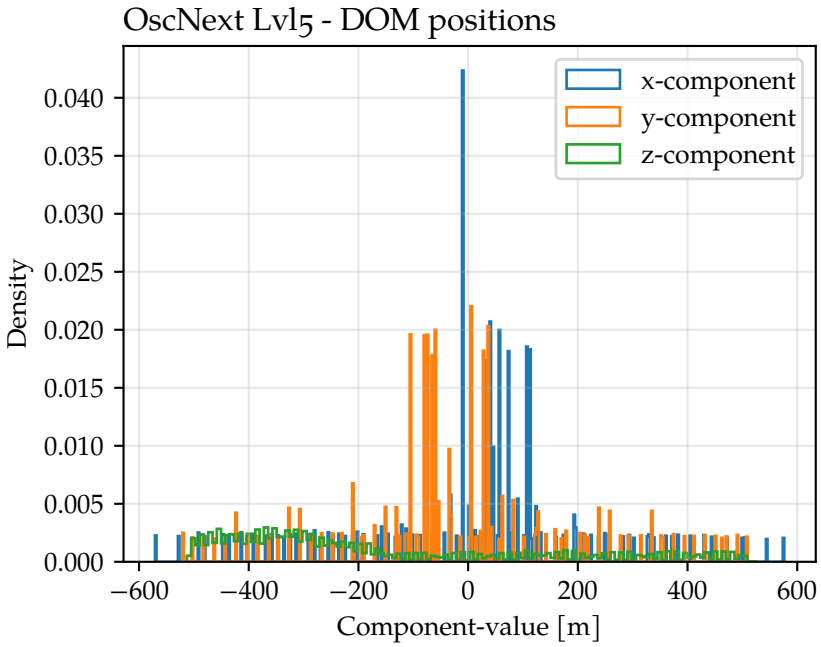



Figure 61: Distributions of $x-$, $y-$, and $z$-coordinates of the DOMs.

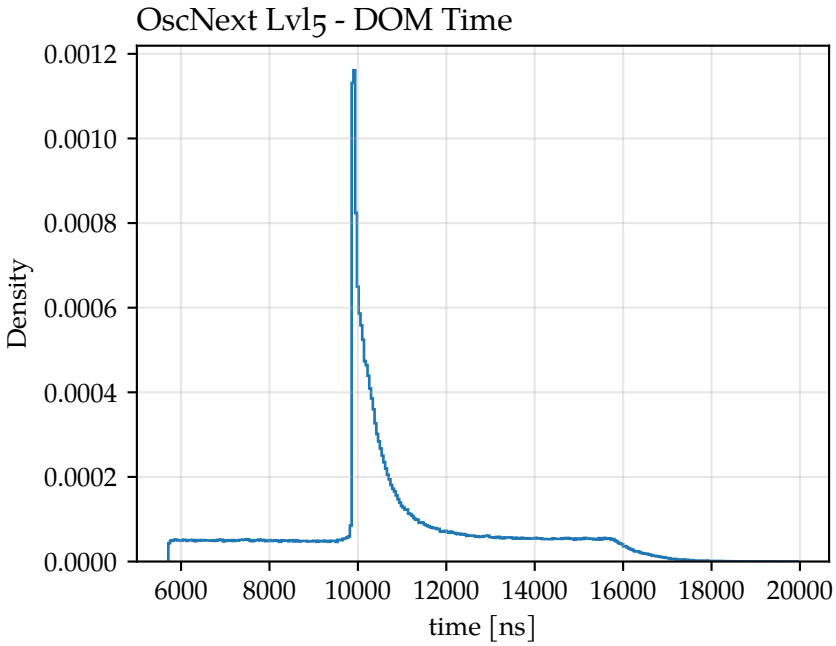



Figure 62: Distribution of DOM trigger times.

## A.2 MODEL SUMMARY

```
Model(
  (mods): ModuleList(
    (0): Sequential(
      (0): ResBlock(
        (linear0): Linear(in_features=7, out_features=64, bias=True)
        (norm1): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
        (non_lin1): LeakyReLU(negative_slope=0.01)
        (linear1): Linear(in_features=64, out_features=64, bias=True)
        (norm2): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
        (non_lin2): LeakyReLU(negative_slope=0.01)
        (linear2): Linear(in_features=64, out_features=64, bias=True)
      )
    )
    (1): RnnBlock(
      (par_RNNs): ModuleList(
        (0): GRU(64, 256, num_layers=3, batch_first=True, bidirectional=True)
      )
      (init_hidden_states): ParameterList(
            (0): Parameter containing: [
                torch.cuda.FloatTensor of size 1536 (GPU 0)
            ]
      )
    )
    (2): Sequential(
      (0): ResBlock(
        (norm1): BatchNorm1d(
            512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
        )
        (non_lin1): LeakyReLU(negative_slope=0.01)
        (linear1): Linear(in_features=512, out_features=512, bias=True)
        (norm2): BatchNorm1d(
            512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
        )
        (non_lin2): LeakyReLU(negative_slope=0.01)
        (linear2): Linear(in_features=512, out_features=512, bias=True)
      )
    )
    (3): Sequential(
      (0): Linear(in_features=512, out_features=8, bias=True)
    )
  )
)
```

## A.3 PERFORMANCE IMPROVEMENT

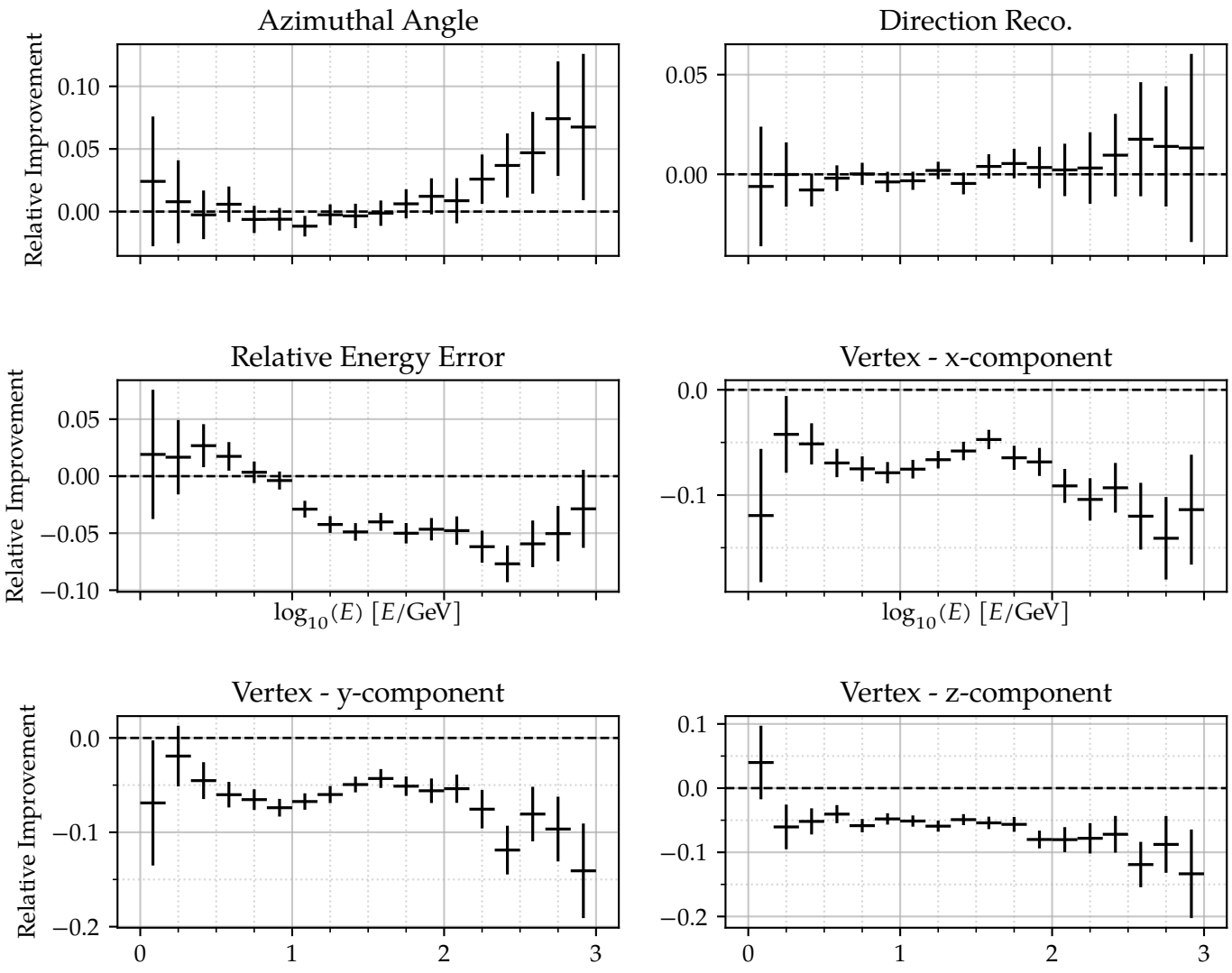



Figure 63: Additional plots of relative improvements in performance w.r.t. $RNN_{all}(3, 256, 2)$, when the regression tasks are divided among 3 models. All architectures are identical, but the loss functions have been altered.

## A.4 PREDICTION DISTRIBUTIONS

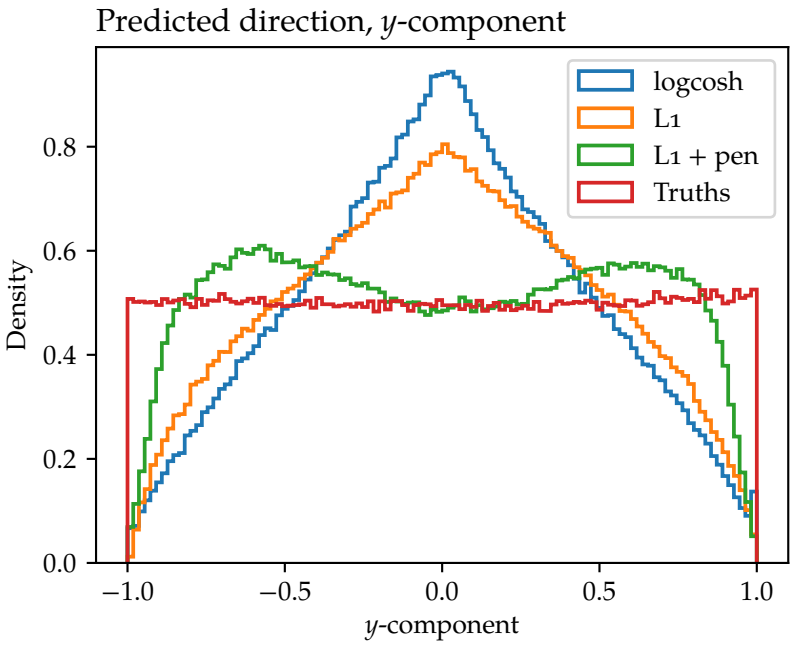



Figure 64: Distributions of y-components when different loss-functions are used in direction regression.

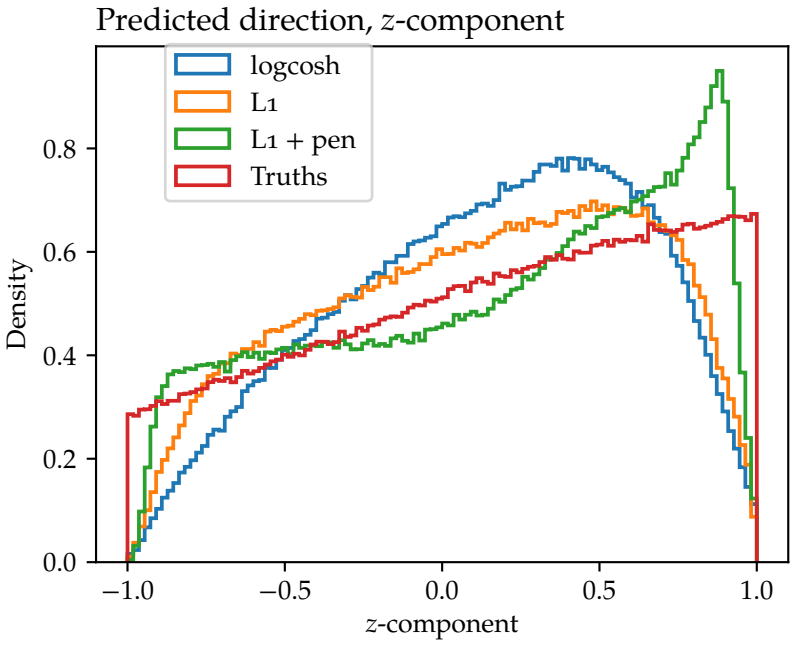



Figure 65: Distributions of z-components when different loss-functions are used in direction regression.

## A.5 BASELINE PERFORMANCE PLOTS

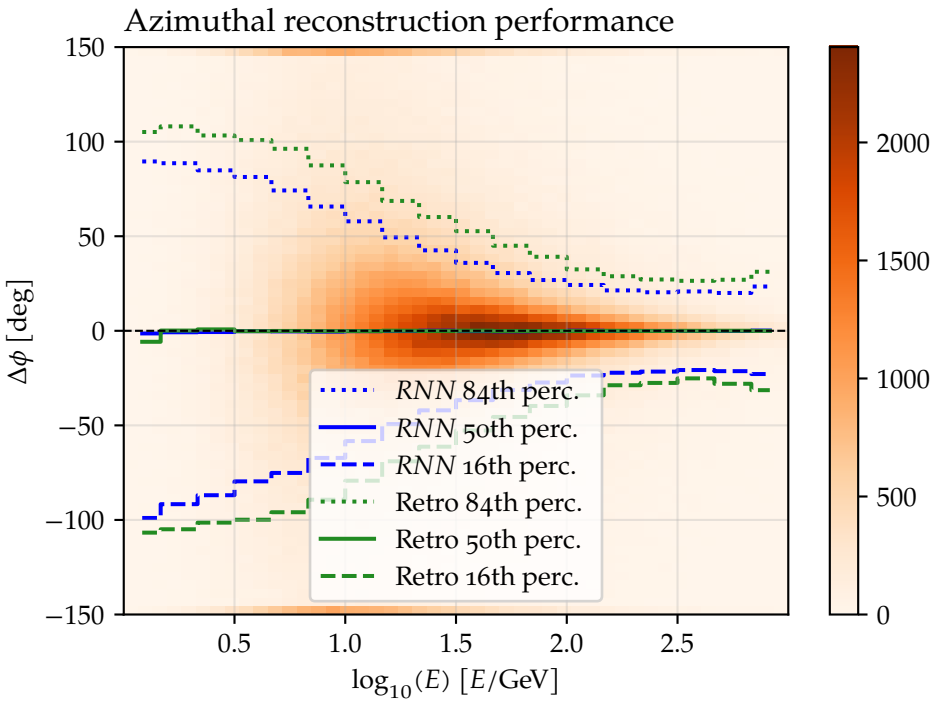



Figure 66: Error distribution of azimuthal reconstruction for the baseline model.

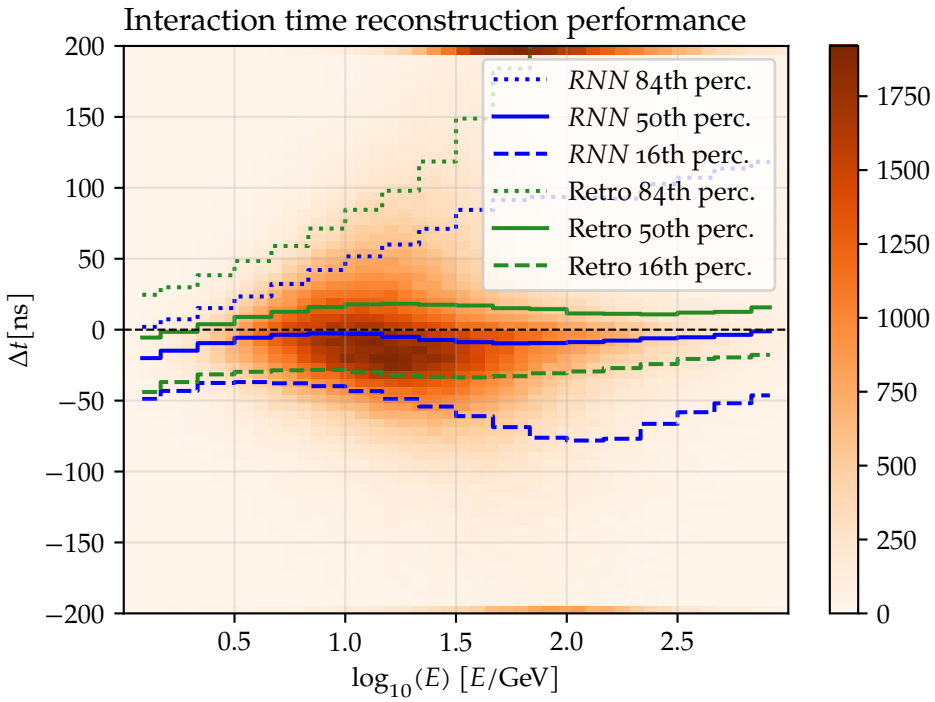



Figure 67: Error distribution of interaction time reconstruction for the baseline model.

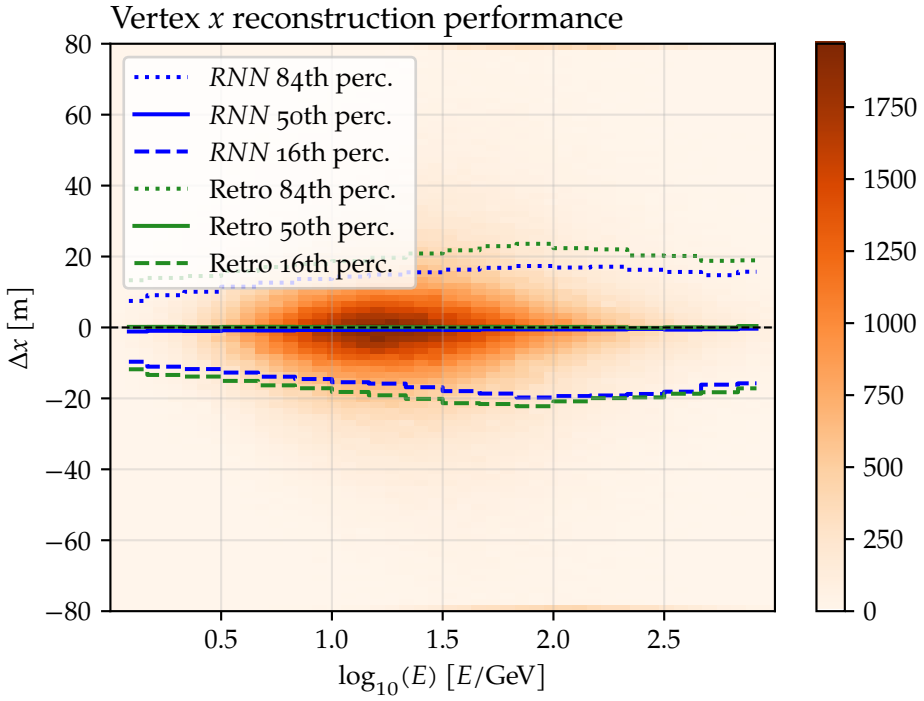



Figure 68: Error distribution of the x-component of the interaction vertex reconstruction for the baseline model.

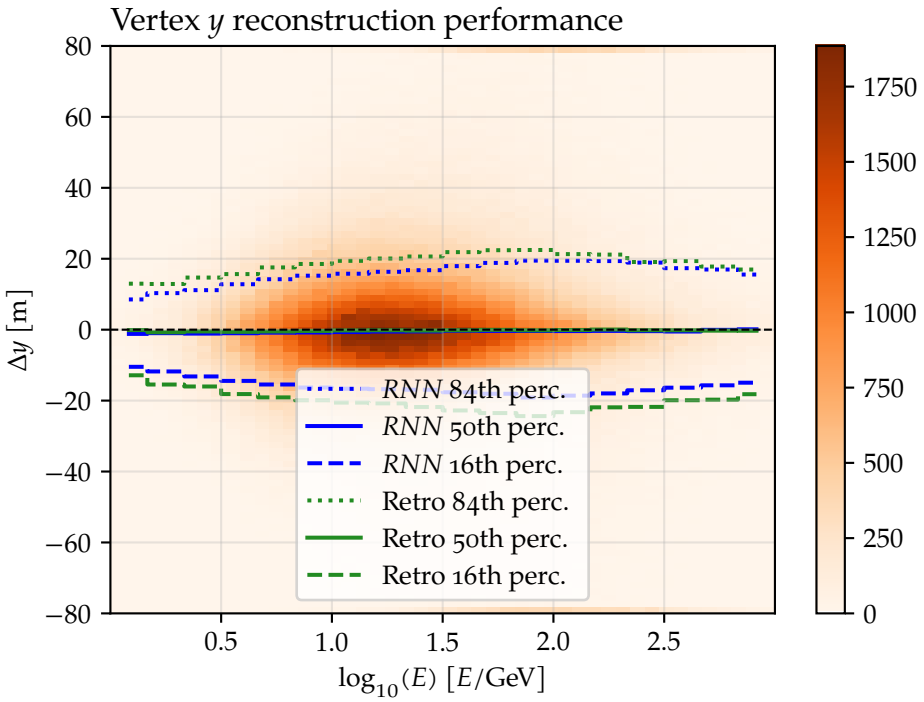



Figure 69: Error distribution of the y-component of the interaction vertex reconstruction for the baseline model.

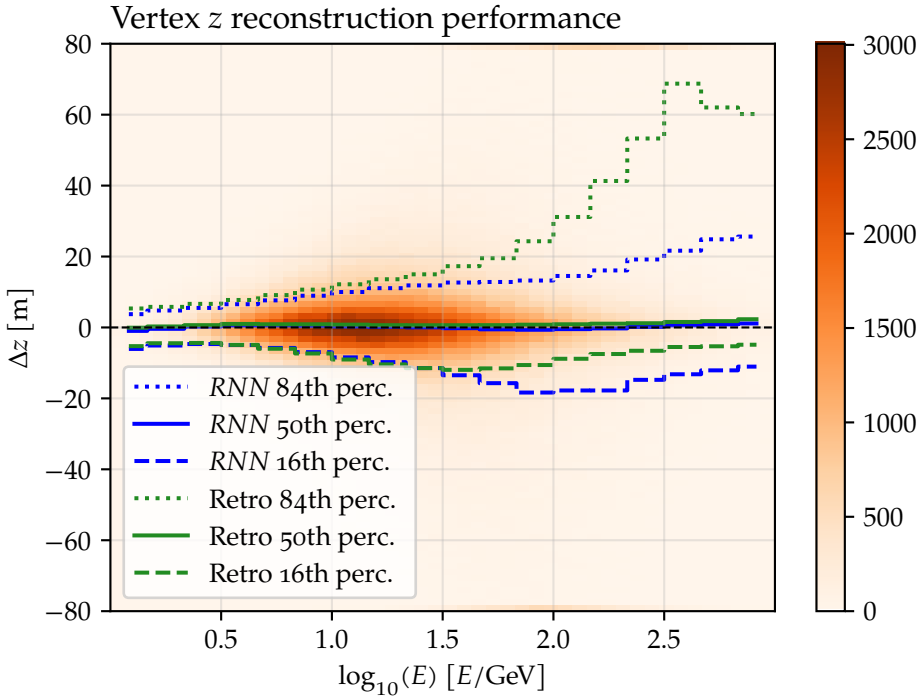



Figure 70: Error distribution of the z-component of the interaction vertex reconstruction for the baseline model.

## A.6 ENSEMBLE PERFORMANCE PLOTS

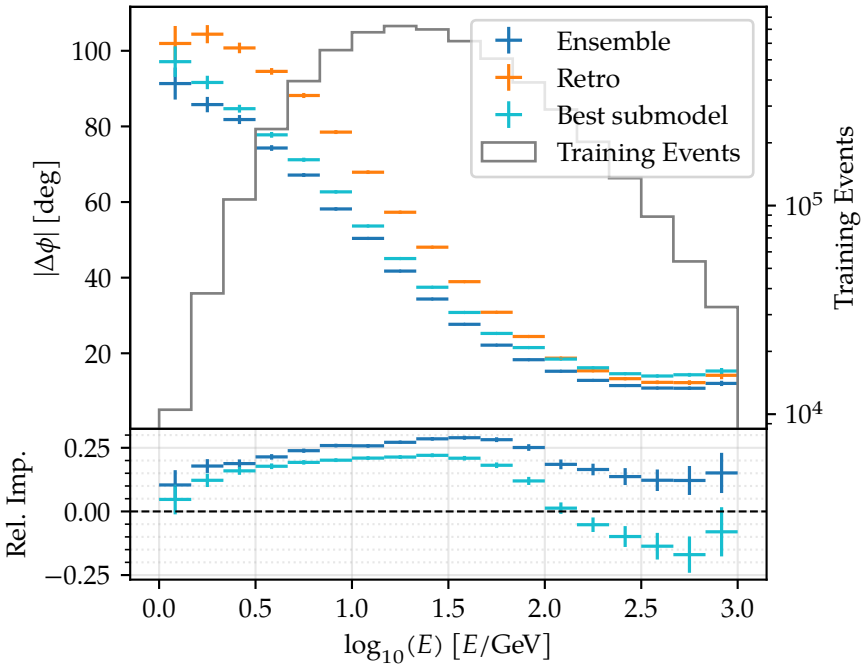



Figure 71: Ensemble model azimuthal reconstruction performance.

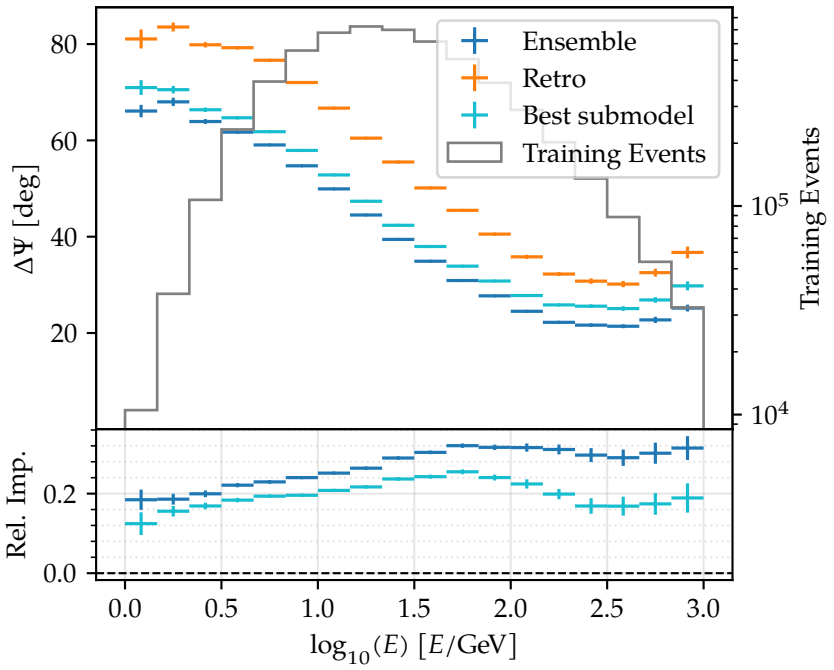



Figure 72: Ensemble model direction reconstruction performance.

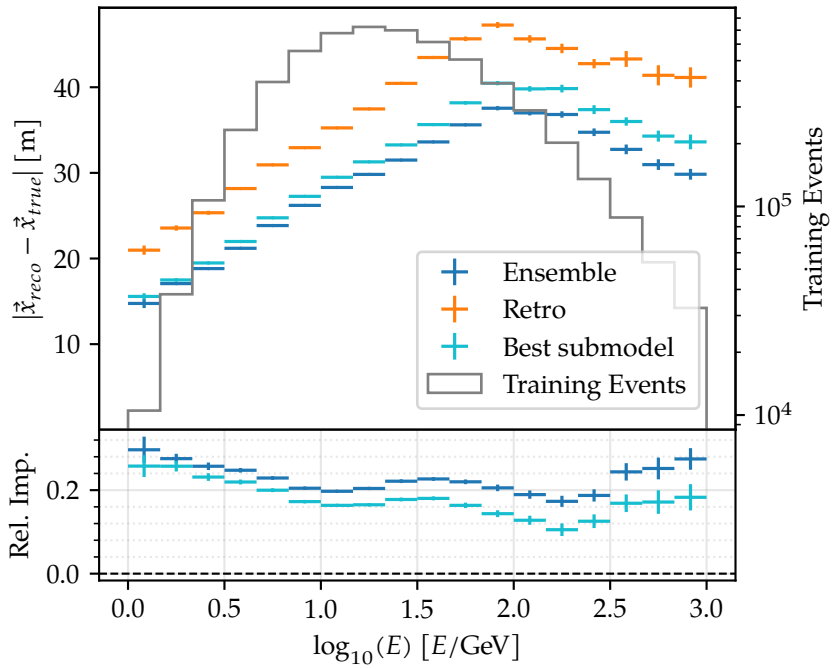



Figure 73: Ensemble model interaction vertex reconstruction performance.

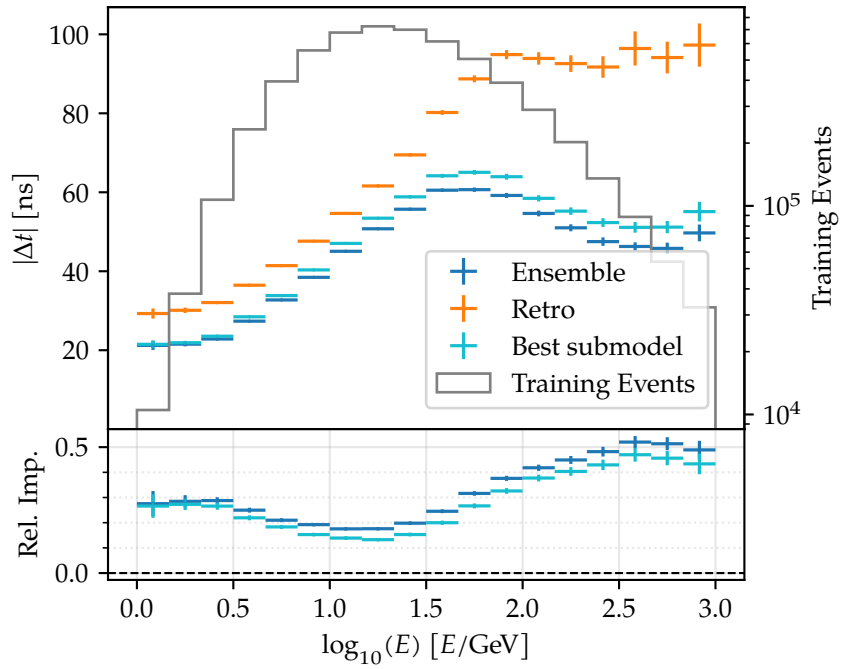



Figure 74: Ensemble model interaction time reconstruction performance.

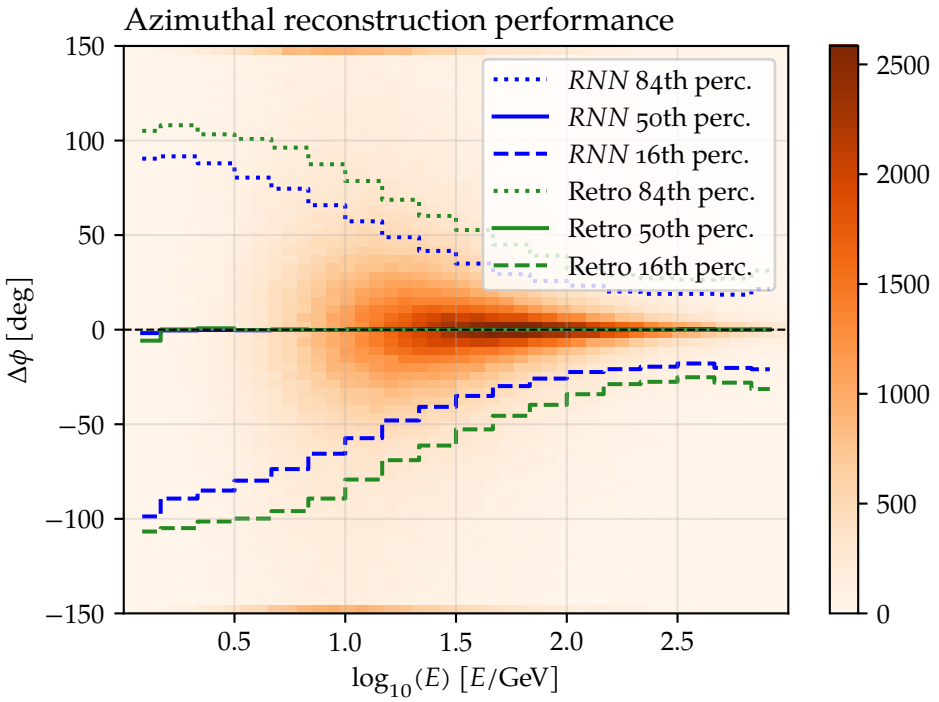



Figure 75: Ensemble model azimuthal error distribution.

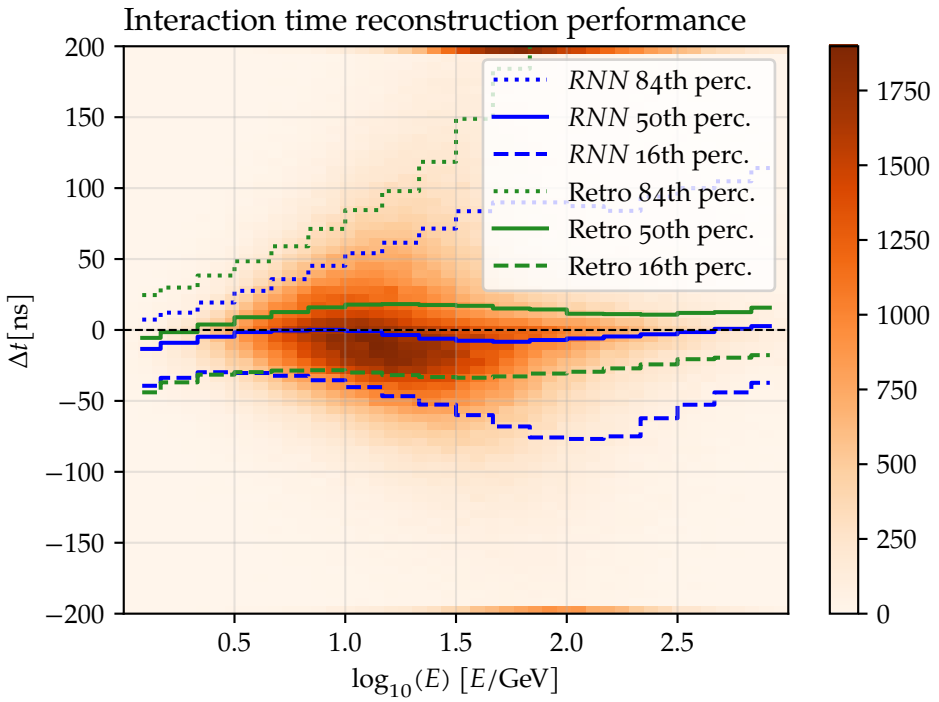



Figure 76: Ensemble model interaction time error distribution.

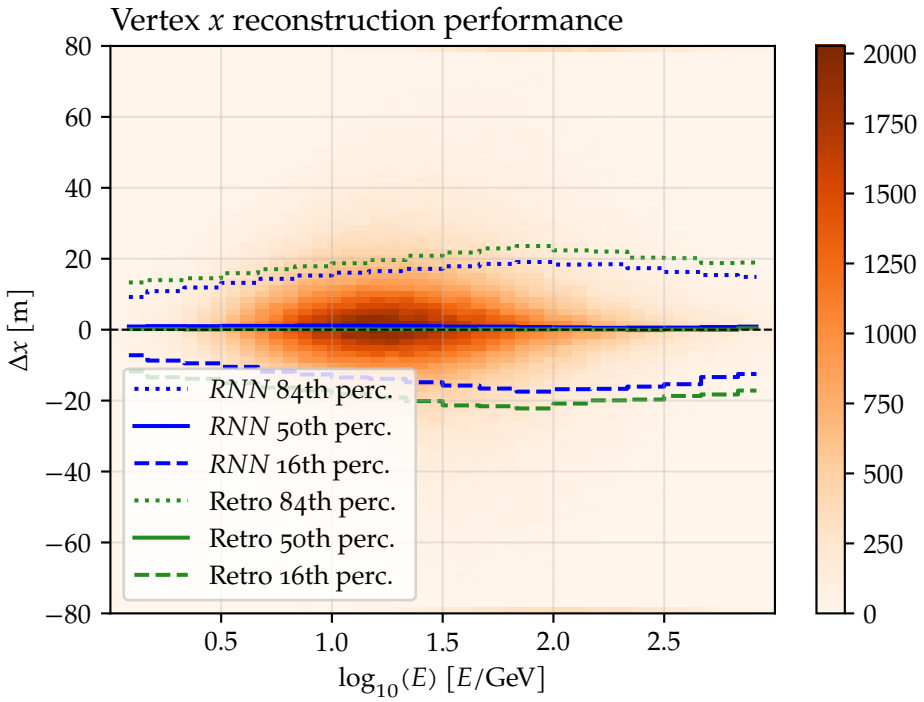



Figure 77: Ensemble model interaction vertex x-coordinate error distribution.

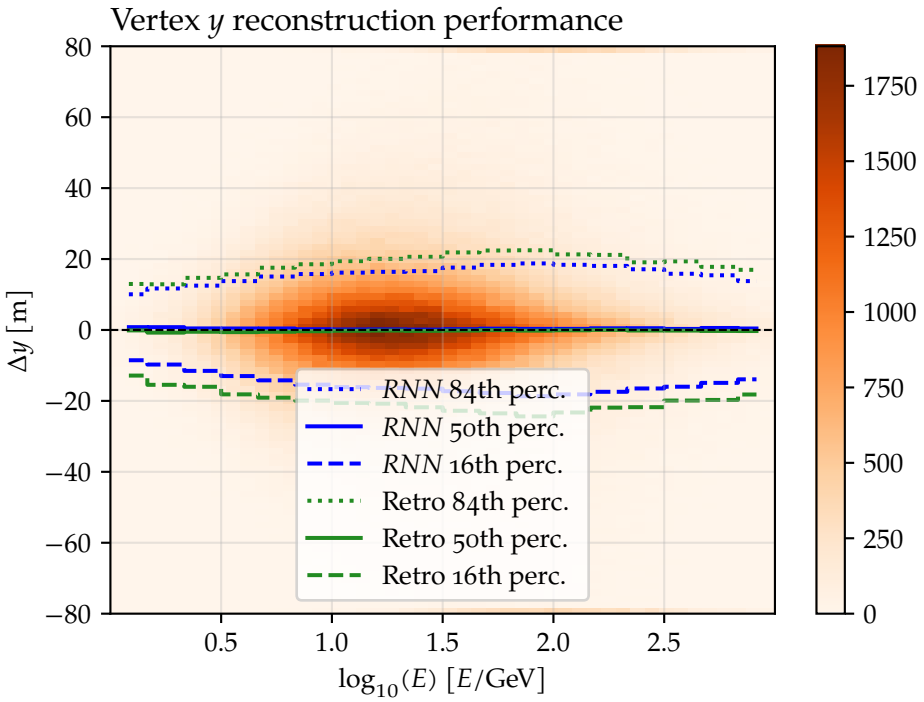



Figure 78: Ensemble model interaction vertex y-coordinate error distribution.

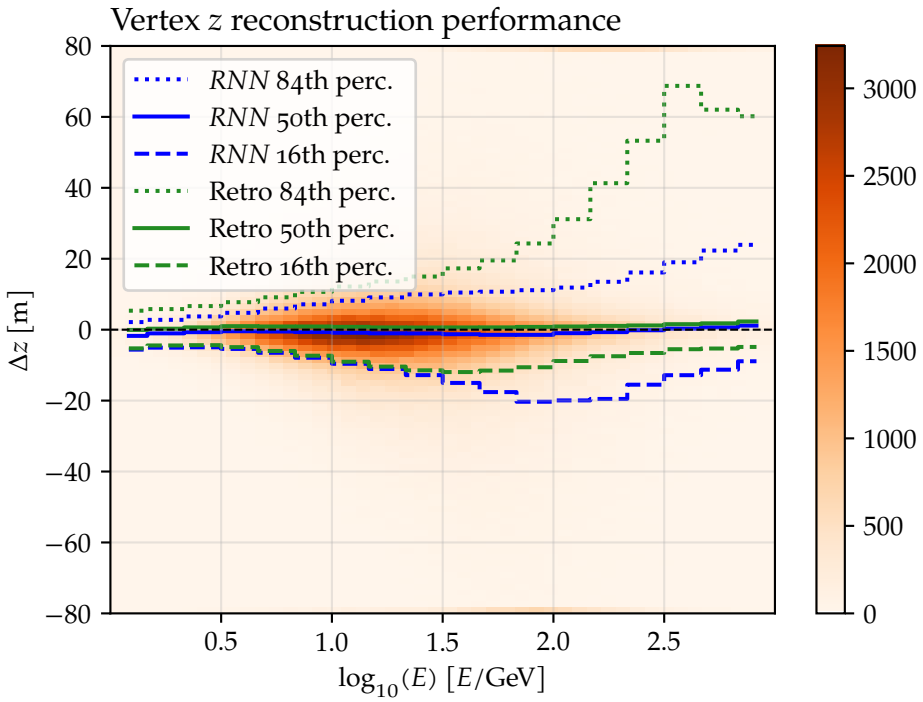



Figure 79: Ensemble model interaction vertex z-coordinate error distribution.

## A.7 ENSEMBLE Z-SCORE DISTRIBUTIONS

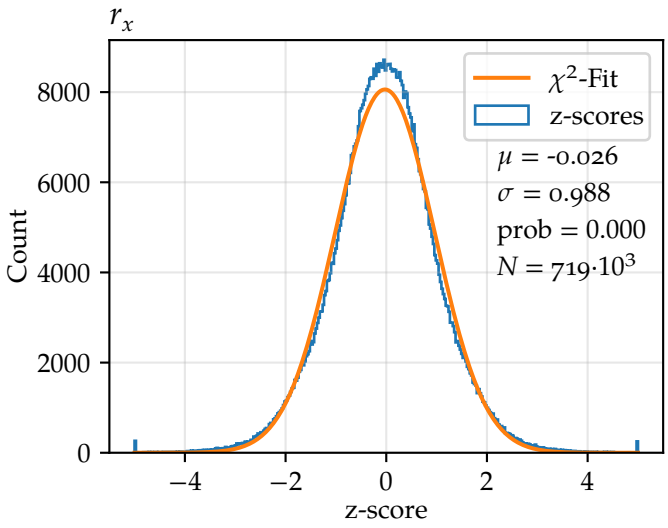



Figure 80: Z-score distribution for reconstructed direction x-component.

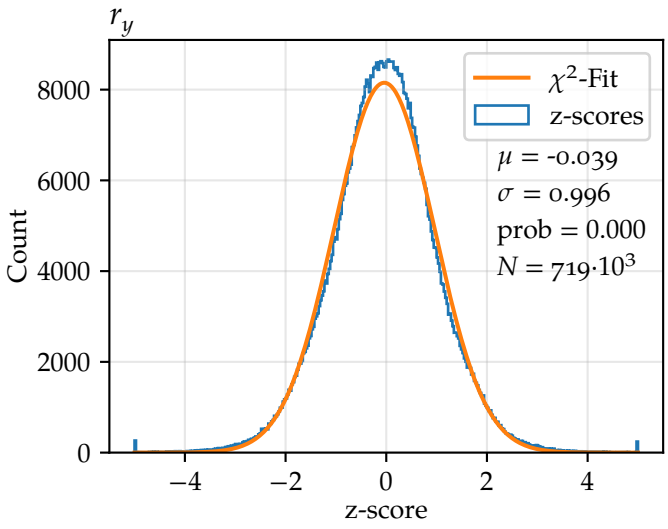



Figure 81: Z-score distribution for reconstructed direction y-component.

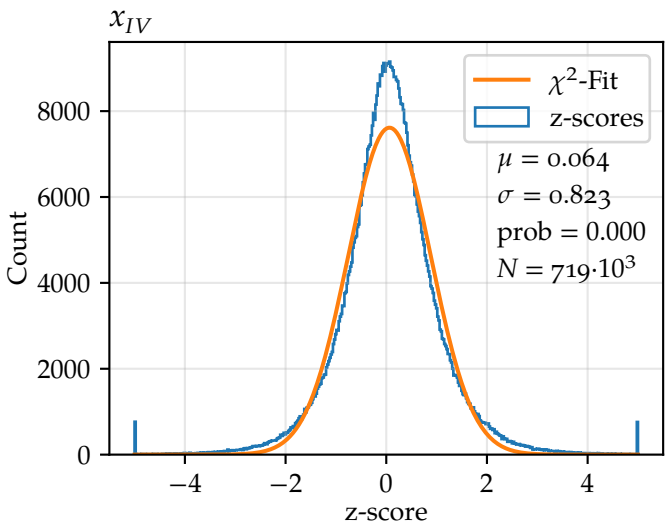



Figure 82: Z-score distribution for reconstructed interaction vertex x-component.

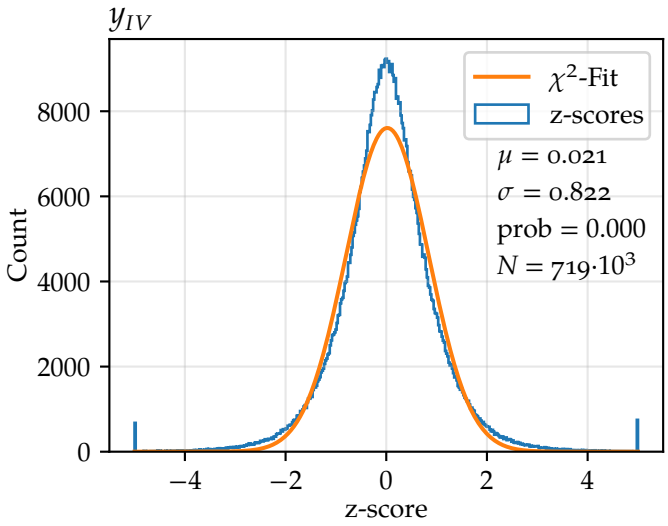



Figure 83: Z-score distribution for reconstructed interaction vertex y-component.

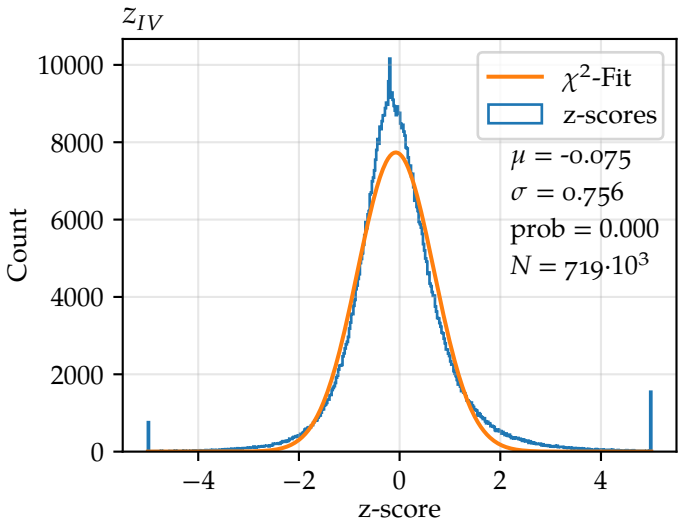



Figure 84: Z-score distribution for reconstructed interaction vertex z-component.

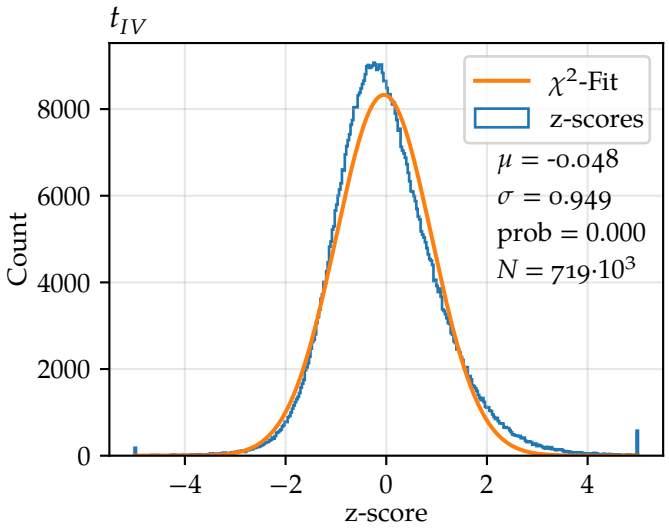



Figure 85: Z-score distribution for reconstructed interaction time.

# BIBLIOGRAPHY


[1] Pierre Radvanyi and Jacques Villain. "The Discovery of Radioactivity". In: *Comptes Rendus Physique* 18.9 (2017), pp. 544–550. ISSN: 1631-0705. DOI: 10.1016/j.crhy.2017.10.008 (cit. on p. 1).

[2] Thomas Corona. "Simulation Tools for the 2KM Detector at the T2K Experiment". In: (Apr. 2020) (cit. on p. 1).

[3] *Dear Radioactive Ladies and Gentlemen - INSPIRE*. https://inspirehep.net/literature/45177 (cit. on p. 1).

[4] J. Chadwick. "Possible Existence of a Neutron". en. In: *Nature* 129.3252 (Feb. 1932), pp. 312–312. ISSN: 1476-4687. DOI: 10.1038/129312a0 (cit. on p. 1).

[5] C. L. Cowan et al. "Detection of the Free Neutrino: A Confirmation". en. In: *Science* 124.3212 (July 1956), pp. 103–104. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.124.3212.103 (cit. on p. 2).

[6] *The Nobel Prize in Physics 1995*. en-US. https://www.nobelprize.org/prizes/physics/1995/summary/ (cit. on p. 2).

[7] *The Standard Model – Modelling The Invisible*. en-GB (cit. on p. 2).

[8] The ATLAS Collaboration. "Observation of a New Particle in the Search for the Standard Model Higgs Boson with the ATLAS Detector at the LHC". In: *Physics Letters B* 716.1 (Sept. 2012), pp. 1–29. ISSN: 03702693. DOI: 10.1016/j.physletb.2012.08.020. arXiv: 1207.7214 (cit. on p. 3).

[9] C. S. Wu et al. "Experimental Test of Parity Conservation in Beta Decay". en. In: *Physical Review* 105.4 (Feb. 1957), pp. 1413–1415. ISSN: 0031-899X. DOI: 10.1103/PhysRev.105.1413 (cit. on p. 3).

[10] SNO Collaboration et al. "Combined Analysis of All Three Phases of Solar Neutrino Data from the Sudbury Neutrino Observatory". In: *Physical Review C* 88.2 (Aug. 2013), p. 025501. ISSN: 0556-2813, 1089-490X. DOI: 10.1103/PhysRevC.88.025501. arXiv: 1109.0763 (cit. on p. 4).

[11] John N. Bahcall. "Solar Neutrinos. I. Theoretical". In: *Physical Review Letters* 12.11 (Mar. 1964), pp. 300–302. DOI: 10.1103/PhysRevLett.12.300 (cit. on p. 5).

[12] Raymond Davis. "Solar Neutrinos. II. Experimental". In: *Physical Review Letters* 12.11 (Mar. 1964), pp. 303–305. DOI: 10.1103/PhysRevLett.12.303 (cit. on p. 5).

[13] Bruce Cleveland et al. "Measurement of the Solar Electron Neutrino Flux with the Homestake Chlorine Detector". In: *The Astrophysical Journal* 496 (Jan. 2009), p. 505. DOI: 10.1086/305343 (cit. on p. 5).

[14] SNO Collaboration et al. "Direct Evidence for Neutrino Flavor Transformation from Neutral-Current Interactions in the Sudbury Neutrino Observatory". In: *Physical Review Letters* 89.1 (June 2002), p. 011301. DOI: 10.1103/PhysRevLett.89.011301 (cit. on p. 5).

[15] *Inverse Beta Processes and Nonconservation of Lepton Charge - INSPIRE*. https://inspirehep.net/literature/42736 (cit. on p. 5).

[16] Ziro Maki, Masami Nakagawa, and Shoichi Sakata. "Remarks on the Unified Model of Elementary Particles". en. In: *Progress of Theoretical Physics* 28.5 (Nov. 1962), pp. 870–880. ISSN: 0033-068X. DOI: 10.1143/PTP.28.870 (cit. on p. 5).

[17] J. W. F. Valle. "Neutrino Physics Overview". In: *Journal of Physics: Conference Series* 53 (Nov. 2006), pp. 473–505. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/53/1/031. arXiv: hep-ph/0608101 (cit. on p. 5).

[18] Yat-Long Chan et al. "Wave-Packet Treatment of Neutrino Oscillations and Its Implications on Determining the Neutrino Mass Hierarchy". In: *The European Physical Journal C* 76.6 (June 2016), p. 310. ISSN: 1434-6044, 1434-6052. DOI: 10.1140/epjc/s10052-016-4143-4. arXiv: 1507.06421 (cit. on p. 6).

[19] *NuFIT | NuFIT*. http://www.nu-fit.org/?q=node/8 (cit. on pp. 8, 9).

[20] L. Wolfenstein. "Neutrino Oscillations in Matter". en. In: *Phys.Rev.D* 17 (Nov. 1977), pp. 2369–2374. DOI: 10.1103/PhysRevD.17.2369 (cit. on p. 8).

[21] S. P. Mikheyev and A. Yu. Smirnov. "Resonant Amplification of $\nu$ Oscillations in Matter and Solar-Neutrino Spectroscopy". en. In: *Il Nuovo Cimento C* 9.1 (Jan. 1986), pp. 17–26. ISSN: 0390-5551. DOI: 10.1007/BF02508049 (cit. on p. 8).

[22] Ivan Esteban et al. "Global Analysis of Three-Flavour Neutrino Oscillations: Synergies and Tensions in the Determination of Theta_23, Delta_CP, and the Mass Ordering". In: *Journal of High Energy Physics* 2019.1 (Jan. 2019), p. 106. ISSN: 1029-8479. DOI: 10.1007/JHEP01(2019)106. arXiv: 1811.05487 (cit. on p. 9).

[23] "Precision Electroweak Measurements on the Z Resonance". en. In: *Physics Reports* 427.5 (May 2006), pp. 257–454. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2005.12.006 (cit. on p. 10).

[24] *(PDF) The LEP Legacy*. en. https://www.researchgate.net/publication/1982871_The_LEP_legacy. DOI: http://dx.doi.org/10.1142/9789812701893_0016 (cit. on p. 10).

[25] A. Boyarsky et al. "Sterile Neutrino Dark Matter". In: *Progress in Particle and Nuclear Physics* 104 (Jan. 2019), pp. 1–45. ISSN: 01466410. DOI: 10.1016/j.ppnp.2018.07.004. arXiv: 1807.07938 (cit. on p. 10).

[26] IceCube Collaboration et al. "The IceCube Neutrino Observatory: Instrumentation and Online Systems". In: *Journal of Instrumentation* 12.03 (Mar. 2017), P03012–P03012. ISSN: 1748-0221. DOI: 10.1088/1748-0221/12/03/P03012. arXiv: 1612.05093 (cit. on pp. 12–14, 21).

[27] IceCube Collaboration et al. "Measurement of South Pole Ice Transparency with the IceCube LED Calibration System". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 711 (May 2013), pp. 73–89. ISSN: 01689002. DOI: 10.1016/j.nima.2013.01.054. arXiv: 1301.5361 (cit. on p. 12).

[28] The IceCube Collaboration. "The Design and Performance of IceCube DeepCore". In: *Astroparticle Physics* 35.10 (May 2012), pp. 615–624. ISSN: 09276505. DOI: 10.1016/j.astropartphys.2012.01.004. arXiv: 1109.6096 (cit. on pp. 13, 15, 38, 39).

[29] The IceCube Collaboration et al. "Calibration and Characterization of the IceCube Photomultiplier Tube". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 618.1-3 (June 2010), pp. 139–152. ISSN: 01689002. DOI: 10.1016/j.nima.2010.03.102. arXiv: 1002.2442 (cit. on pp. 13, 14).

[30] The IceCube Collaboration. "The IceCube Data Acquisition System: Signal Capture, Digitization, and Timestamping". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 601.3 (Apr. 2009), pp. 294–316. ISSN: 01689002. DOI: 10.1016/j.nima.2009.01.001. arXiv: 0810.4930 (cit. on p. 14).

[31] Victor Hess. "On the Observations of the Penetrating Radiation during Seven Balloon Flights". In: *arXiv:1808.02927 [astro-ph, physics:physics]* (Aug. 2018). arXiv: 1808.02927 [astro-ph, physics:physics] (cit. on p. 16).

[32] *The Nobel Prize in Physics 1936*. en-US. https://www.nobelprize.org/prizes/-physics/1936/summary/ (cit. on p. 16).

[33] M Tanabashi et al. "Review of Particle Physics: Particle Data Group". In: *Physical Review D* 98 (Aug. 2018) (cit. on pp. 16, 17, 19, 20).

[34] Mark Thomson. "Modern Particle Physics". en. In: (2013) (cit. on p. 17).

[35] Takaaki Kajita. *Atmospheric Neutrinos*. en. https://www.hindawi.com/journals/ahep/2012/504715/. Review Article. 2012. DOI: 10.1155/2012/504715 (cit. on p. 18).

[36] ANTARES Collaboration. "A Deep Sea Telescope for High Energy Neutrinos". In: *arXiv:astro-ph/9907432* (July 1999). arXiv: astro-ph/9907432 (cit. on p. 18).

[37] *Classical Electrodynamics | BibSonomy*. https://www.bibsonomy.org/bibtex/2baac05176a92886bbe1eae5ee72cf234/cernlibrary (cit. on p. 19).

[38] P. A. Čerenkov. "Visible Radiation Produced by Electrons Moving in a Medium with Velocities Exceeding That of Light". In: *Physical Review* 52.4 (Aug. 1937), pp. 378–379. DOI: 10.1103/PhysRev.52.378 (cit. on p. 20).

[39] Hadiseh Alaeian. *An Introduction to Cherenkov Radiation*. http://large.stanford.edu/courses/2014/ph241/alaeian2/ (cit. on p. 20).

[40] Ava Ghadimi, Marcos Santander, and VERITAS Collaboration. "The Search for Sources of High Energy Astrophysical Neutrinos with VERITAS". In: (Jan. 2017), B4.006 (cit. on p. 22).

[41] Summer Blot. *oscNext (V00.04)*. Apr. 2020 (cit. on pp. 23, 24, 37, 38, 40).

[42] *IceCubeOpenSource/Retro*. IceCube Open Source Software. Apr. 2020 (cit. on p. 23).

[43] Murray Campbell, A. Joseph Hoane, and Feng-hsiung Hsu. "Deep Blue". en. In: *Artificial Intelligence* 134.1 (Jan. 2002), pp. 57–83. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(01)00129-1 (cit. on p. 25).

[44] G. Cybenko. "Approximation by Superpositions of a Sigmoidal Function". en. In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1989), pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274 (cit. on p. 26).

[45] Olivier Delalleau and Yoshua Bengio. "Shallow vs. Deep Sum-Product Networks". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 666–674 (cit. on p. 26).

[46] *Understanding LSTM Networks – Colah's Blog*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (cit. on pp. 27, 28).

[47] Y. Bengio, P. Simard, and P. Frasconi. "Learning Long-Term Dependencies with Gradient Descent Is Difficult". In: *IEEE Transactions on Neural Networks* 5.2 (Mar. 1994), pp. 157–166. ISSN: 1941-0093. DOI: 10.1109/72.279181 (cit. on p. 27).

[48] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 27).

[49] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *arXiv:1412.3555 [cs]* (Dec. 2014). arXiv: 1412.3555 [cs] (cit. on p. 27).

[50] Klaus Greff et al. "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (Oct. 2017), pp. 2222–2232. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2016.2582924. arXiv: 1503.04069 (cit. on p. 28).

[51] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv:1706.03762 [cs]* (Dec. 2017). arXiv: 1706.03762 [cs] (cit. on pp. 29, 48).

[52] Wassily Hoeffding. "Probability Inequalities for Sums of Bounded Random Variables". In: *Journal of the American Statistical Association* 58.301 (1963), pp. 13–30. ISSN: 0162-1459. DOI: 10.2307/2282952 (cit. on p. 30).

[53] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". en. In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0 (cit. on p. 31).

[54] Dominic Masters and Carlo Luschi. "Revisiting Small Batch Training for Deep Neural Networks". In: *arXiv:1804.07612 [cs, stat]* (Apr. 2018). arXiv: 1804.07612 [cs, stat] (cit. on p. 32).

[55] Nitish Shirish Keskar et al. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *arXiv:1609.04836 [cs, math]* (Feb. 2017). arXiv: 1609.04836 [cs, math] (cit. on p. 32).

[56] *Loss Landscape | A.I Deep Learning Explorations of Morphology & Dynamics*. en-US. https://losslandscape.com/ (cit. on p. 32).

[57] Sebastian Ruder. "An Overview of Gradient Descent Optimization Algorithms". In: *arXiv:1609.04747 [cs]* (June 2017). arXiv: 1609.04747 [cs] (cit. on p. 32).

[58] Yurii Nesterov. *A Method for Unconstrained Convex Minimization Problem with the Rate of Convergence o(1/Kˆ2)*. en. /paper/A-method-for-unconstrained-convex-minimization-with-Nesterov/ed910d96802212c9e45d956adaa27d
1983 (cit. on p. 32).

[59] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980 [cs] (cit. on p. 32).

[60] "Overfitting". en. In: *Wikipedia* (Apr. 2020) (cit. on p. 35).

[61] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv:1502.03167* [*cs*] (Mar. 2015). arXiv: 1502.03167 [cs] (cit. on p. 35).

[62] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *arXiv:1607.06450* [*cs, stat*] (July 2016). arXiv: 1607.06450 [cs, stat] (cit. on p. 36).

[63] C. Andreopoulos et al. "The GENIE Neutrino Monte Carlo Generator". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 614.1 (Feb. 2010), pp. 87–104. ISSN: 01689002. DOI: 10.1016/j.nima.2009.12.009. arXiv: 0905.2517 (cit. on p. 37).

[64] Claudio Kopper. *Claudiok/Clsim*. Dec. 2019 (cit. on p. 37).

[65] *IceCube Dust Map*. http://icecube.berkeley.edu/~bay/dustmap/ (cit. on p. 38).

[66] Mike Schuster and Kuldip Paliwal. "Bidirectional Recurrent Neural Networks". In: *Signal Processing, IEEE Transactions on* 45 (Dec. 1997), pp. 2673–2681. DOI: 10.1109/78.650093 (cit. on p. 47).

[67] *Neural Networks, Types, and Functional Programming – Colah's Blog*. http://colah.github.io/posts/2015-09-NN-Types-FP/ (cit. on p. 47).

[68] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385* [*cs*] (Dec. 2015). arXiv: 1512.03385 [cs] (cit. on p. 47).

[69] Kaiming He et al. "Identity Mappings in Deep Residual Networks". In: *arXiv:1603.05027* [*cs*] (July 2016). arXiv: 1603.05027 [cs] (cit. on p. 48).

[70] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 978-0-262-03561-3 (cit. on p. 49).

[71] Leslie N. Smith. "Cyclical Learning Rates for Training Neural Networks". In: *arXiv:1506.01186* [*cs*] (Apr. 2017). arXiv: 1506.01186 [cs] (cit. on p. 49).

[72] Leslie N. Smith and Nicholay Topin. "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates". In: *arXiv:1708.07120* [*cs, stat*] (May 2018). arXiv: 1708.07120 [cs, stat] (cit. on p. 50).

[73] Liyuan Liu et al. "On the Variance of the Adaptive Learning Rate and Beyond". In: *arXiv:1908.03265* [*cs, stat*] (Apr. 2020). arXiv: 1908.03265 [cs, stat] (cit. on p. 50).

[74] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. "Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization". In: *arXiv:1109.5647* [*cs, math*] (Dec. 2012). arXiv: 1109.5647 [cs, math] (cit. on p. 50).

[75] Sepp Hochreiter et al. *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*. 2001 (cit. on p. 51).

[76] Andrew L. Maas. *Rectifier Nonlinearities Improve Neural Network Acoustic Models*. en. /paper/Rectifier-Nonlinearities-Improve-Neural-Network-Maas/367f2c63a6f6a10b3b64b8729d601e69337ee3cc. 2013 (cit. on p. 51).

[77] Diganta Misra. "Mish: A Self Regularized Non-Monotonic Neural Activation Function". In: *arXiv:1908.08681 [cs, stat]* (Oct. 2019). arXiv: 1908.08681 [cs, stat] (cit. on p. 51).

[78] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (June 2014), pp. 1929–1958 (cit. on p. 52).

[79] Xavier Glorot and Yoshua Bengio. "Understanding the Difficulty of Training Deep Feedforward Neural Networks". en. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Mar. 2010. Chap. Machine Learning, pp. 249–256 (cit. on p. 53).

[80] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *arXiv:1502.01852 [cs]* (Feb. 2015). arXiv: 1502.01852 [cs] (cit. on p. 53).

[81] *First Course in Probability, A, 9th Edition*. en. /content/one-dot-com/one-dot-com/us/en/higher-education/program.html (cit. on p. 53).

[82] Frederick Mosteller. "On Some Useful "Inefficient" Statistics". EN. In: *Annals of Mathematical Statistics* 17.4 (Dec. 1946), pp. 377–408. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177730881 (cit. on p. 54).

[83] "Receiver Operating Characteristic". en. In: *Wikipedia* (June 2020) (cit. on p. 59).

[84] David J. Hand. "Measuring Classifier Performance: A Coherent Alternative to the Area under the ROC Curve". In: *Machine Learning* (2009). DOI: 10.1007/s10994-009-5119-5 (cit. on p. 60).

[85] H. B. Mann and D. R. Whitney. "On a Test of Whether One of Two Random Variables Is Stochastically Larger than the Other". EN. In: *Annals of Mathematical Statistics* 18.1 (Mar. 1947), pp. 50–60. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177730491 (cit. on p. 60).

[86] Corinna Cortes and Mehryar Mohri. "Confidence Intervals for the Area Under the ROC Curve". In: *Advances in Neural Information Processing Systems 17*. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. MIT Press, 2005, pp. 305–312 (cit. on p. 60).

[87] E. L. Lehmann. "A General Concept of Unbiasedness". EN. In: *Annals of Mathematical Statistics* 22.4 (Dec. 1951), pp. 587–592. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729549 (cit. on p. 60).

[88] Peter I. Frazier. "A Tutorial on Bayesian Optimization". In: *arXiv:1807.02811 [cs, math, stat]* (July 2018). arXiv: 1807.02811 [cs, math, stat] (cit. on p. 63).

[89] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *arXiv:1912.01703 [cs, stat]* (Dec. 2019). arXiv: 1912.01703 [cs, stat] (cit. on p. 64).

[90] *GeForce GTX 1080 Graphics Cards from NVIDIA GeForce*. en-sg. https://www.nvidia.com/en-sg/geforce/products/10series/geforce-gtx-1080/ (cit. on p. 71).

[91] *ImprovedLinefit — Metaproject 'icerec.Releases.V05-01-02' at R177505*. https://docs.icecube.aq/icerec/V05-01-02/projects/linefit/improved.html (cit. on p. 75).

[92] Peter J. Huber. "Robust Estimation of a Location Parameter". EN. In: *Annals of Mathematical Statistics* 35.1 (Mar. 1964), pp. 73–101. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177703732 (cit. on p. 75).

[93] *Tensor of Inertia — Metaproject 'combo.Trunk' at R180534*. https://-docs.icecube.aq/combo/trunk/projects/tensor-of-inertia/index.html?highlight=tensor%20inertia (cit. on p. 75).

[94] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *arXiv:1506.02142 [cs, stat]* (Oct. 2016). arXiv: 1506.02142 [cs, stat] (cit. on p. 78).

[95] Leo Breiman. "Random Forests". en. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324 (cit. on p. 78).

[96] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the Difficulty of Training Recurrent Neural Networks". In: *arXiv:1211.5063 [cs]* (Feb. 2013). arXiv: 1211.5063 [cs] (cit. on p. 89).

[97] David H. Wolpert. "Stacked Generalization". en. In: *Neural Networks* 5.2 (Jan. 1992), pp. 241–259. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(05)80023-1 (cit. on p. 91).

[98] IceCube Collaboration et al. "Observation of the Cosmic-Ray Shadow of the Moon with IceCube". In: *Physical Review D* 89.10 (May 2014), p. 102004. ISSN: 1550-7998, 1550-2368. DOI: 10.1103/PhysRevD.89.102004. arXiv: 1305.6811 (cit. on p. 97).

[99] Aya Ishihara. "The IceCube Upgrade – Design and Science Goals". In: *arXiv:1908.09441 [astro-ph, physics:physics]* (Aug. 2019). arXiv: 1908.09441 [astro-ph, physics:physics] (cit. on p. 97).

# LIST OF FIGURES

# LIST OF TABLES