Sensitivity studies on tau neutrino appearance with KM3NeT/ORCA using Deep Learning Techniques

Sensitivitätsstudien zur Messung von Tau Neutrinos mit dem Neutrinoteleskop KM3NeT/ORCA unter Verwendung von Deep Learning Methoden

Der Naturwissenschaftlichen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg

zur
Erlangung des Doktorgrades Dr. rer. nat.

vorgelegt von

Michael Moser

aus Nürnberg

Als Dissertation genehmigt
von der Naturwissenschaftlichen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung: 29.07.2020

Vorsitzender des Promotionsorgans: Prof. Dr. Georg Kreimer

Gutachter: Prof. Dr. Gisela Anton
Dr. Piotr Mijakowski

„The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way
Where many paths and errands meet.
And whither then? I cannot say.“

— Bilbo Baggins, in *The Lord of the Rings* by J. R. R. Tolkien —

Contents

Chapter 1 Abstract

In the last few decades, it has been experimentally verified that neutrinos can change their flavor while propagating through space by so-called neutrino oscillations. The oscillation probabilities of neutrinos to oscillate from one flavor into another flavor are described by the neutrino mixing matrix.

An important open question in particle physics is if the neutrino mixing matrix is unitary or not. Currently, the uncertainties on several matrix elements are too large in order to draw significant conclusions on the unitarity of the matrix. This is mostly due to the low experimental statistics in the tau neutrino sector. KM3NeT/ORCA is a water Cherenkov neutrino detector under construction in the Mediterranean Sea with several megatons of instrumented volume. Its main objective is the determination of the neutrino mass ordering. However, it will also observe about 4000 tau neutrino events per year, which will significantly improve the available tau neutrino statistics. In KM3NeT/ORCA, tau neutrinos will be identified by observing a statistical excess of charged-current-induced, cascade-like events with respect to the atmospheric electron neutrino expectation.

The reconstruction of the low-level detector data consists of several stages. First, the detector background consisting of atmospheric muons and randomly correlated noise by ^{40}K decays in seawater and by bioluminescence needs to be distinguished from the expected neutrino signals by using a classification algorithm. After this, track-like (ν_μ^{CC}) events are distinguished from cascade-like (ν_e^{CC}, ν_e^{NC}) events based on another classifier. At last, neutrino properties like the energy and the direction of the neutrinos need to be reconstructed. Until now, maximum likelihood-based reconstruction algorithms accompanied by shallow machine learning techniques like Random Forests have been employed in the standard KM3NeT/ORCA reconstruction pipeline to tackle all of these tasks.

A novel technique for the reconstruction of the detector data is to employ deep artificial neural networks. In this approach, simulated, low-level experimental data is used for the training of a deep neural network. During the training process, the network builds a representation of the typical event properties that then allows for the reconstruction of the events. Within the scope of this thesis, deep convolutional neural networks (CNNs) have been designed for each of the aforementioned tasks, leading to a complete, CNN-based reconstruction chain. It is shown that this first application of a CNN-based event reconstruction to large-volume neutrino telescope data of KM3NeT/ORCA yields competitive results and performance improvements with respect to classical approaches.

For the background classification, the CNN-based method leads to a significant improvement in the ability to distinguish atmospheric muon events from neutrino events. Additionally, a significant gain in performance for the track-shower classification can be observed. And for the reconstruction of the neutrino properties, an improvement in the energy reconstruction of track-like events is made.

Applying the CNN-based reconstruction to an analysis on the sensitivity of KM3NeT/ORCA to the appearance of tau neutrinos shows that the sensitivity can be improved by more than 10% with respect to the currently employed reconstruction techniques.

Zusammenfassung

Innerhalb der letzten Jahrzehnte wurde experimentell nachgewiesen, dass Neutrinos durch sogenannte Neutrinooszillationen ihre Familie wechseln können. Die Oszillationswahrscheinlichkeiten von Neutrinos, von einer Familie in eine andere Familie zu wechseln, werden mit der Neutrino-Mischungsmatrix beschrieben.

Eine wichtige offene Frage in der Teilchenphysik ist, ob die Neutrino-Mischungsmatrix unitär ist. Aktuell sind die Unsicherheiten auf diverse Matrixelemente zu groß, um signifikante Rückschlüsse auf die Unitarität der Matrix zu ziehen. Dies ist vor allem auf die geringe experimentelle Statistik im Tau Neutrino Bereich zurückzuführen. KM3NeT/ORCA ist ein im Mittelmeer im Bau befindliches Wasser-Tscherenkow-Neutrinoteleskop mit mehreren Megatonnen Meerwasser an instrumentiertem Volumen. Das Hauptziel von KM3NeT/ORCA ist die Bestimmung der Neutrinomassenhierarchie. Gleichzeitig wird KM3NeT/ORCA aber auch etwa 4000 Tau Neutrino Ereignisse pro Jahr beobachten, was die verfügbare Tau Neutrino Statistik deutlich verbessern wird. In KM3NeT/ORCA werden Tau Neutrinos durch Beobachtung eines statistischen Überschusses an geladenen-strom-induzierten, kaskadenartigen Ereignissen gegenüber der atmosphärischen Elektron Neutrino Erwartung allein identifiziert.

Die Rekonstruktion der Rohdaten des KM3NeT/ORCA Detektors besteht aus mehreren Stufen. Zunächst muss der überwältigende Detektoruntergrund, bestehend aus atmosphärischen Myonen und zufällig korreliertem Rauschen durch ^{40}K Zerfälle im Meerwasser und durch Biolumineszenz, mithilfe eines Klassifikationsalgorithmus von den erwarteten Neutrinosignalen unterschieden werden. Danach werden spurähnliche (ν_μ^{CC}) Ereignisse von kaskadenähnlichen (ν_e^{CC}, ν_e^{NC}) Ereignissen auf der Basis eines weiteren Klassifikators unterschieden. Zuletzt müssen Neutrinoeigenschaften wie die Energie und die Richtung der Neutrinos rekonstruiert werden. Bislang wurden in der Standard Rekonstruktionspipeline von KM3NeT/ORCA Maximum Likelihood Verfahren in Kombination mit Shallow Machine Learning Techniken wie Random Forests eingesetzt, um all diese Aufgaben zu bewältigen.

Eine neuartige Methode zur Rekonstruktion der Detektordaten ist der Einsatz von tiefen, künstlichen neuronalen Netzen. Bei diesem Ansatz werden simulierte Rohdaten des Detektors für das Training eines neuronalen Netzes verwendet. Während des Trainingsprozesses baut das Netz eine interne Darstellung der typischen Ereigniseigenschaften auf, welches dann die Rekonstruktion der Ereignisse ermöglicht. Im Rahmen dieser Arbeit wurden für jede der oben genannten Aufgaben sogenannte Convolutional Neural Networks (CNNs) entworfen, mit dem Resultat einer vollständigen, CNN-basierten Rekonstruktionskette. Es wird gezeigt, dass diese erste Anwendung einer CNN-basierten Ereignisrekonstruktion auf simulierte Detektordaten des KM3NeT/ORCA Neutrinoteleskops zu konkurrenzfähigen Ergebnissen und teilweise auch zu Verbesserungen gegenüber klassischen Ansätzen führt.

Für die Untergrundklassifizierung führt die CNN-basierte Methode zu einer signifikanten Verbesserung der Fähigkeit, atmosphärische Myonen Ereignisse von Neutrino Ereignissen zu unterscheiden. Zusätzlich kann ein signifikante Verbesserung für die Unterscheidung von spurartigen und kaskadenartigen Ereignissen beobachtet werden. Außerdem wird für die Rekonstruktion der Neutrinoeigenschaften eine Verbesserung bei der Energierekonstruktion von spurähnlichen Ereignissen erreicht.

Die Anwendung der CNN-basierten Rekonstruktion auf eine Analyse der Empfindlichkeit von KM3NeT/ORCA auf das Auftreten von Tau Neutrinos zeigt, dass die Empfindlichkeit um mehr als 10% gegenüber den derzeit verwendeten Rekonstruktionstechniken verbessert werden kann.

CHAPTER 2 Introduction

I have done a terrible thing, I have postulated a particle that cannot be detected.
— *Wolfgang Pauli* —

2.1 Basic neutrino properties

SINCE their first postulation by Wolfgang Pauli in 1930 [1], neutrinos have played a puzzling role in the Standard Model of particle physics, as some of their properties are significantly different to those of the other elementary particles. Neutrinos only interact via the weak force, since they carry no electric charge or color charge. There are three confirmed generations of neutrinos, each associated to one charged lepton within the *flavors* $\alpha = e, \mu, \tau$:

$$\begin{bmatrix} e \\ \nu_e \end{bmatrix}, \begin{bmatrix} \mu \\ \nu_\mu \end{bmatrix}, \begin{bmatrix} \tau \\ \nu_\tau \end{bmatrix} . \tag{2.1}$$

A distinct property of neutrinos is their small mass. The current upper limit on the absolute mass scale of the neutrinos, derived by the beta decay of Tritium, is $m_\nu < 1.1\,\mathrm{eV}\,c^{-2}$ at a 90% confidence limit [2]. The next lightest particle in the Standard Model, the electron, is significantly heavier with a mass of $m_e = 0.511 \times 10^6\,\mathrm{eV}\,c^{-2}$ [3]. In addition, only (anti) neutrinos with a (right-handed) left-handed helicity, i.e. the spin direction is (parallel) antiparallel to the momentum direction, interact. This is in contrast to the electron, muon and tau leptons, which can interact as both left-handed as well as right-handed particles.

Unlike the neutrinos' doublet partners like the electron, neutrinos can change their flavor through so-called *neutrino oscillations*.

2.2 Neutrino oscillations

While neutrino oscillations were first considered by Bruno Pontecorvo in 1957 [4], it was not until 2001 that the existence could be proven by the SNO experiment [5]. Since then, neutrino oscillations have been observed by atmospheric, solar, reactor and accelerator experiments [6]. As will be shown in the next paragraph, the neutrino oscillation mechanism requires massive neutrinos. Thus, these observations also established that the neutrino has a non-zero mass, which is in contrast to the Standard Model of particle physics.

For the neutrino oscillation mechanism, it is assumed that there are two different kinds of neutrino eigenstates: the flavor eigenstates $\nu_{e,\mu,\tau}$ and the mass eigenstates $\nu_{1,2,3}$. The flavor eigenstates only take part in interactions, while the mass eigenstates propagate through space. The relationship between the flavor eigenstates and the mass eigenstates is described by a 3×3 matrix U, similar to the CKM matrix [7] in the quark sector:

$$\nu_\alpha = U^*_{\alpha i} \nu_i \,, \tag{2.2}$$

with $\alpha = e, \mu, \tau$, $i = 1, 2, 3$ and the PMNS (Pontecorvo–Maki–Nakagawa–Sakata) matrix U. Since the PMNS matrix is not the identity matrix, the flavor eigenstates and the mass eigenstates differ from each other. Thus, it can be shown that the probability for measuring a particular flavor of a neutrino created at a time t_0 differs from the probability to measure the same flavor after a certain amount of time $t_0 + \Delta t$. Hence, neutrinos can change their flavor during their propagation through space.

For the propagation of neutrinos through vacuum, the probability of the transition $\nu_\alpha \to \nu_{\alpha'}$ is given by the following formula [8]:

$$\begin{aligned} P_{\nu_\alpha \to \nu_{\alpha'}} = \delta_{\alpha\alpha'} - 4 \sum_{i>k} \mathrm{Re}[U^*_{\alpha i} U_{\alpha' i} U_{\alpha k} U^*_{\alpha' k}] \sin^2 \left(\frac{\Delta m^2_{ki} L}{4E} \right) \\ + 2 \sum_{i>k} \mathrm{Im}[U^*_{\alpha i} U_{\alpha' i} U_{\alpha k} U^*_{\alpha' k}] \sin \left(\frac{\Delta m^2_{ki} L}{2E} \right) , \end{aligned} \tag{2.3}$$

where E is the neutrino energy, L is the propagation length, $\Delta m^2_{ki} = m^2_i - m^2_k$ is the mass difference between the eigenstates i and k, and α is the neutrino flavor index. Therefore, the absolute masses m_k cannot be measured in neutrino oscillation experiments if the neutrinos travel through a vacuum-like medium, since only the mass differences Δm_{ki} are part of the transition probability, cf. equation 2.3. Thus, it is not known yet, which mass eigenstate is the heaviest one and which mass eigenstate is the lightest one. Currently, there are two possible configurations of the neutrino mass eigenstates, the *normal* and the *inverted* ordering. They are commonly also referred to as the normal hierarchy (NH) and the inverted hierarchy (IH), cf. figure 2.1.

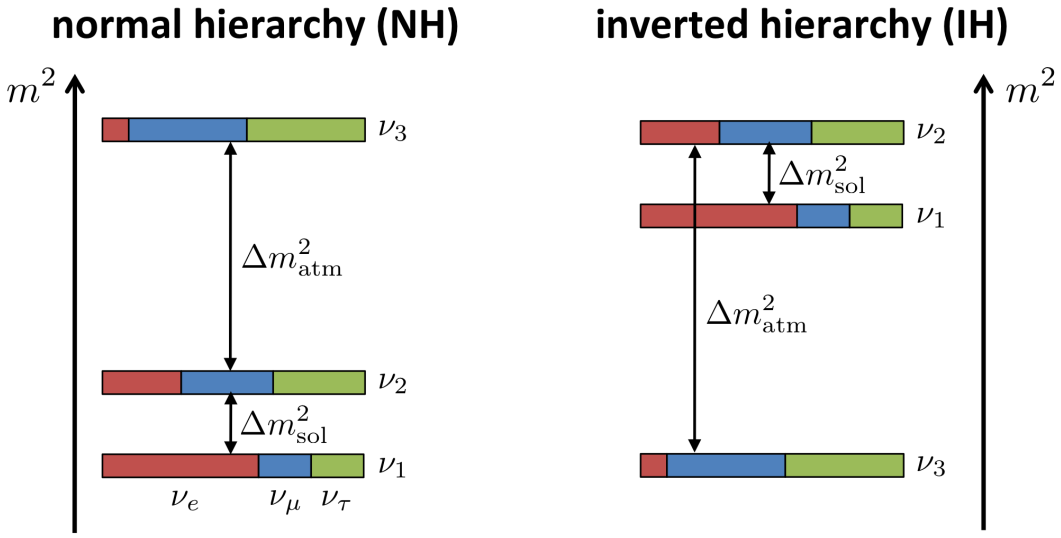

Fig. 2.1: Normal (left) and inverted (right) hierarchy of the neutrino mass eigenstates. $\Delta m^2_{\mathsf{sol}}$ and $\Delta m^2_{\mathsf{atm}}$ are synonyms for Δm^2_{21} and Δm^2_{32} respectively. The absolute mass scale (i.e. m^2_{lightest}) is not known. [9]

However, the oscillation probabilities have to be modified in order to describe neutrinos propagating through matter [10, 11]. For electron containing matter, the presence of electrons can lead to charged current, elastic scatterings of electron neutrinos. Based on this effect, future experiments like KM3NeT/ORCA [12] will determine the neutrino mass hierarchy in this decade. As of now, the normal hierarchy is favored with a significance of $2.5\,\sigma$ [13, 14].

Other than this, it is not known if the PMNS matrix U is unitary or not. On the one hand, the PMNS matrix must be unitary, since it is contained in the time evolution operator of the neutrinos. On the other hand, a violation of the unitarity of the 3×3 matrix U can be a hint of new physics beyond the Standard Model. For example, in the case of the existence of right-handed neutrinos proposed by the seesaw model [15], additional parameters are necessary to describe the neutrino mixing. Then, the original 3×3 PMNS matrix must not be unitary anymore.

In order to prove the unitarity experimentally, the parameters in the PMNS matrix must be probed with high precision. The specific structure of the PMNS mixing matrix is as follows:

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_{23} & \sin\theta_{23} \\ 0 & -\sin\theta_{23} & \cos\theta_{23} \end{pmatrix} \begin{pmatrix} \cos\theta_{13} & 0 & \sin\theta_{13}e^{-i\delta_{\mathrm{CP}}} \\ 0 & 1 & 0 \\ -\sin\theta_{13}e^{-i\delta_{\mathrm{CP}}} & 0 & \cos\theta_{13} \end{pmatrix} \begin{pmatrix} \cos\theta_{12} & \sin\theta_{12} & 0 \\ -\sin\theta_{12} & \cos\theta_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (2.4)$$

where θ_{12}, θ_{13} and θ_{23} are the three mixing angles $\theta_{ij} \in [0, \pi/2)$ and $\delta_{\mathrm{CP}} \in [0, 2\pi]$ is a CP violating [16] phase factor. In this notation, the currently unknown Majorana phases [17] are neglected, since they do not have an effect on neutrino oscillation experiments. For the phase δ_{CP}, the uncertainties are too large in order to draw significant conclusions [13, 14]:

$$\delta_{\mathrm{CP}} = 222^{+38}_{-28}\ (\mathrm{NH}) \text{ or } \delta_{\mathrm{CP}} = 285^{+24}_{-26}\ (\mathrm{IH})\,. \quad (2.5)$$

Equation 2.5 indicates that the value of δ_{CP} is dependent on the neutrino mass hierarchy. Other than this and apart from θ_{23}, the mixing angles have been determined precisely [13, 14]:

$$\theta_{12} = 33.82^{+0.78}_{-0.76}\,, \quad (2.6)$$

$$\theta_{13} = 8.61^{+0.13}_{-0.13}\ (\mathrm{NH}) \text{ or } \theta_{13} = 8.65^{+0.13}_{-0.12}\ (\mathrm{IH})\,, \quad (2.7)$$

$$\theta_{23} = 48.3^{+1.1}_{-1.9}\ (\mathrm{NH}) \text{ or } \theta_{23} = 48.6^{+1.1}_{-1.5}\ (\mathrm{IH})\,. \quad (2.8)$$

As for the CP violating phase δ_{CP}, the best-fit values of the mixing angles θ_{23} and θ_{13} are dependent on the neutrino mass ordering. Additionally, the current best-fit value of θ_{23} is close to a maximal mixing, where $\theta_{23} = \pi/4$. However, the uncertainties on the value of θ_{23} are still large. For example, assuming normal ordering, the $3\,\sigma$ range of θ_{23} is 40.8° to 51.3°. The parameter region of $\theta_{23} < 45°$ is commonly referred to as the *first octant* solution and $\theta_{23} > 45°$ is commonly referred to as the *second octant* solution. At the same time, the sensitivity of future neutrino mass hierarchy experiments is highly dependent on the octant of θ_{23}.

Consequently, future experiments will have to probe both θ_{23} as well as δ_{CP} such that the fundamental structure of the PMNS matrix will finally be known. In the case of θ_{23}, its precise determination can be used to probe the unitarity of the neutrino mixing matrix. As already mentioned, deviations from the 3×3 neutrino oscillation scenario can be a first sign of new physics beyond the Standard Model.

Aim of this thesis: ν_τ appearance with KM3NeT/ORCA

As shown in the previous paragraphs, θ_{23} is currently the least constrained mixing angle in the PMNS matrix. At the present time, its value is mainly based on ν_μ disappearance measurements in $\nu_\mu \to \nu_\tau$ oscillations. However, it is also possible to probe the appearance of tau neutrinos directly.

KM3NeT/ORCA is a water Cherenkov neutrino detector under construction, which will be able to observe about 4000 tau neutrinos per year. Compared to past experiments, this will improve the statistics on measured tau neutrinos by more than a factor of one hundred. In this work, ν_τ appearance is studied with KM3NeT/ORCA by employing novel machine learning techniques based on deep neural networks.

2.3 Electroweak unification

In the Standard Model, the weak force is unified with the electromagnetic force at high energies, resulting in the electroweak force. The associated symmetry group describing the electroweak interaction is a composition of the weak isospin $SU(2)_L$ symmetry group, assumed to be only coupling to left-handed particles, and the weak hypercharge $U(1)_Y$ symmetry group:

$$SU(2)_L \otimes U(1)_Y \,. \tag{2.9}$$

Since the Lagrangian must be invariant under local $SU(2)_L \otimes U(1)_Y$ gauge transformations, four massless gauge bosons are induced. Three W_μ^i $(i = 1, 2, 3)$ bosons arise due to the three generators $T^k = \frac{1}{2}\sigma^k$ of the $SU(2)_L$ group and a single B_μ boson arises due to the single generator Y of the $U(1)_Y$ group. Adding mass terms to the Lagrangian would violate the gauge symmetry, so there must be another mechanism in order to generate the masses of the observed $\mathrm{W}^\pm$ and Z^0 bosons. In the electroweak theory, the massive $\mathrm{W}^\pm$ and Z^0 bosons as well as the massless photon are created by spontaneous symmetry breaking due to the Higgs mechanism [18]:

$$SU(2)_L \otimes U(1)_Y \to U(1)_{\mathrm{EM}} \tag{2.10}$$

The two massive charged $W^\pm$ bosons are then a combination of W^1 and W^2:

$$W_\mu^\pm = \frac{1}{\sqrt{2}} \left(W_\mu^1 \mp i W_\mu^2 \right) , \tag{2.11}$$

while the field of the Z^0 boson, Z_μ, and the field of the photon, A_μ, are related to the W_3 and B_μ fields by the Weinberg electroweak mixing angle θ_W [19]:

$$\begin{pmatrix} W_\mu^3 \\ B_\mu \end{pmatrix} = \begin{pmatrix} \cos\theta_W & \sin\theta_W \\ -\sin\theta_W & \cos\theta_W \end{pmatrix} \begin{pmatrix} Z_\mu \\ A_\mu \end{pmatrix} . \tag{2.12}$$

Neutrinos can only interact with matter by the exchange of a Z^0 boson, called a *neutral current* (NC) interaction, or a $\mathrm{W}^\pm$ boson, called a *charged current* (CC) interaction. The $\mathrm{W}^\pm$ bosons only couple to left-handed particles and right-handed antiparticles [18]:

$$\mathcal{L}_{\mathrm{CC}} = \frac{e}{\sqrt{2}\sin\theta_w} \left(W_\mu^+ J_+^\mu + W_\mu^- J_-^\mu \right) . \tag{2.13}$$

Here, J_+^μ is a left-handed current and J_-^μ is the corresponding hermitian conjugate. For a single quark and lepton family, the current is as follows:

$$J_+^\mu = V_{ij}^{\mathrm{CKM}} \bar{u}_L \gamma^\mu d_L + \bar{\nu}_{eL} \gamma^\mu e_L \,, \tag{2.14}$$

where V_{ij} is the CKM matrix and γ^μ are the gamma matrices. On the other hand, the Z^0 boson, mediating the neutral current, couples to both left-handed and right-handed particles, although not with the same magnitude. The neutral current Lagrangian is given as [19]:

$$\mathcal{L}_{\mathrm{NC}} = \mathcal{L}_{\mathrm{QED}} + \mathcal{L}_{\mathrm{NC}}^Z \,. \tag{2.15}$$

Here, $\mathcal{L}_{\mathrm{QED}}$ is the quantum electrodynamics Lagrangian [18]:

$$\mathcal{L}_{\mathrm{QED}} = eA_\mu \sum_i Q_i \left(\bar{\psi}_i^L \gamma^\mu \psi_i^L + \bar{\psi}_i^R \gamma^\mu \psi_i^R \right) = eA_\mu J_{\mathrm{em}}^\mu \,, \tag{2.16}$$

for a given doublet ψ_i and with the electromagnetic current J_{em}^μ and the electromagnetic charge operator $Q = T^3 + Y\mathbb{1}$, where T^3 is the weak isospin and Y is the weak hypercharge.

Thus, the electromagnetic current couples to left-handed and right-handed particles. $\mathcal{L}^Z_{\mathrm{NC}}$ contains the interaction of the Z^0 boson [18, 19]:

$$\mathcal{L}^Z_{\mathrm{NC}} = \frac{e}{2\sin\theta_W \cos\theta_W}\left(J^\mu_3 - 2\sin^2\theta_W J^\mu_{\mathrm{em}}\right) Z_\mu \ , \ J^\mu_3 = \sum_i \bar{\psi}^L_i \gamma^\mu T^3 \psi^L_i \, , \tag{2.17}$$

where J^μ_3 is the neutral weak current and Z_μ is the field of the Z^0 boson. Only due to the electromagnetic current J^μ_{em}, the Z^0 boson can couple to both left- and right-handed particles.

2.4 Interaction of neutrinos with matter

Both charged current as well as neutral current interactions of neutrinos feature an inherently low cross section compared to the electromagnetic interactions of other, charged fermions for example. And in addition, the cross section itself is highly dependent on the second scattering constituent, i.e. the type of fermion. The following sections, which introduce the main interaction types of neutrinos with matter, are based on [20], [21] and [22].

2.4.1 Neutrino scattering off free fermions

In this section, the scattering of neutrinos off free fermions, i.e. leptons or quarks, is discussed. This scattering process can occur in two different ways, either due to a charged current or a neutral current interaction. For both types it is now assumed that the four-momentum q of the mediating boson is small compared to its mass, i.e. $|q^2| \ll M^2_{\mathrm{Z,W}}$. At first, charged current interactions are introduced:

$$\nu_l + f \to l + f' \, , \tag{2.18}$$

where ν_l is a neutrino of given lepton flavor l, f is a fermion, l' is a lepton of the same flavor as in ν_l and f' is another fermion different from f. Here, we assume that the center of mass energy in the interaction is high enough that all masses can be safely neglected. The cross section is given as follows [20]:

$$\sigma_{\mathrm{CC}}\left(\nu f\right) = \frac{G^2_F s}{\pi} = 17.2 \times 10^{-42}\,\mathrm{cm}^2 \times E_\nu/\mathrm{GeV} \, , \tag{2.19}$$

with G_F as the Fermi constant and s as the center-of-mass energy. Thus, the cross section increases linearly with energy. The fraction of the neutrino energy that is transferred to the target system, here the fermion f, is called the *reaction inelasticity* y, often referred to as the *bjorken-y*:

$$y = \frac{E_\nu - E_f}{E_\nu} \, . \tag{2.20}$$

It can be shown that in the case of interacting antineutrinos, the cross section is three times lower due to helicity constraints [21]:

$$\sigma_{\mathrm{CC}}\left(\bar{\nu} f\right) = \frac{G^2_F s}{3\pi} \, . \tag{2.21}$$

For the total cross section, the neutral current interaction needs to be taken into account as well. An example is a muon neutrino scattering off an electron:

$$\nu_\mu + e^- \to \nu_\mu + e^- \, . \tag{2.22}$$

Contrarily to the charged current interaction, the neutral current interaction can couple to both the left-handed as well as the right-handed electron. And since the scattering of a muon neutrino off an electron can only be mediated by the neutral current, the total cross section for $\nu_\mu + e^- \to \nu_\mu + e^-$ scattering is equal to the NC cross section [20]:

$$\sigma\left(\nu_\mu + e^- \to \nu_\mu + e^-\right)_{\mathrm{TOT}} = \frac{G^2_F s}{\pi}\left(\frac{1}{4} - \sin^2\theta_W + \frac{4}{3}\sin^4\theta_W\right) \, . \tag{2.23}$$

In the case of $\nu_e + e^- \to \nu_e + e^-$ scattering though, the total cross section is composed of charged current as well as neutral current interactions [20]:

$$\sigma\left(\nu_e + e^- \to \nu_e + e^-\right)_{\mathrm{TOT}} = \frac{G_F^2 s}{\pi}\left(1 - 2\sin^2\theta_W + \frac{4}{3}\sin^4\theta_W\right), \tag{2.24}$$

which is significantly larger than for the neutral current only $\nu_\mu + e^- \to \nu_\mu + e^-$ interaction.

2.4.2 Neutrino scattering off nucleons

For neutrino energies at the GeV scale, which is the region of interest for KM3NeT/ORCA, the cross section of neutrinos is dominated by the scattering of neutrinos off nucleons. At this energy scale, there are three main processes that contribute to the total cross section and the type of interaction depends on the four momentum Q that is transferred during the interaction. For higher Q, the interaction gets more and more inelastic.

The first scattering process is the scattering of the neutrino off an entire nucleon. In the case of a charged current interaction this is called *quasielastic* scattering, while it is called *elastic* scattering in the case of neutral current interactions [21]. As Q is increasing, the neutrino can also excite the nucleon, which then decays, in a *resonance* production process [21]. And with even higher Q, the neutrino can resolve the single quark constituents instead of the whole nucleon, which is called *deep inelastic* scattering. This breaks up the nucleon and produces a hadronic shower cascade. For charged current, deep inelastic scatterings, the ratio of the neutrino to the antineutrino cross section is about two [22]. Additionally, the cross section of charged current deep inelastic scatterings is about a factor of three higher than those of neutral current, deep inelastic scatterings [22]. It can be shown that the charged current, neutrino-nucleon cross section is approximately independent of the reaction inelasticity y, while the charged current, antineutrino-nucleon cross section has a $(1-y)^2$ dependence [22]. Thus, the distribution of the reaction inelasticity for deep inelastic, charged current, antineutrino-nucleon interactions is shifted to a lower value compared to neutrino-nucleon interactions. The neutrino and antineutrino charged current cross sections per nucleon for each of these processes are shown in figure 2.2.

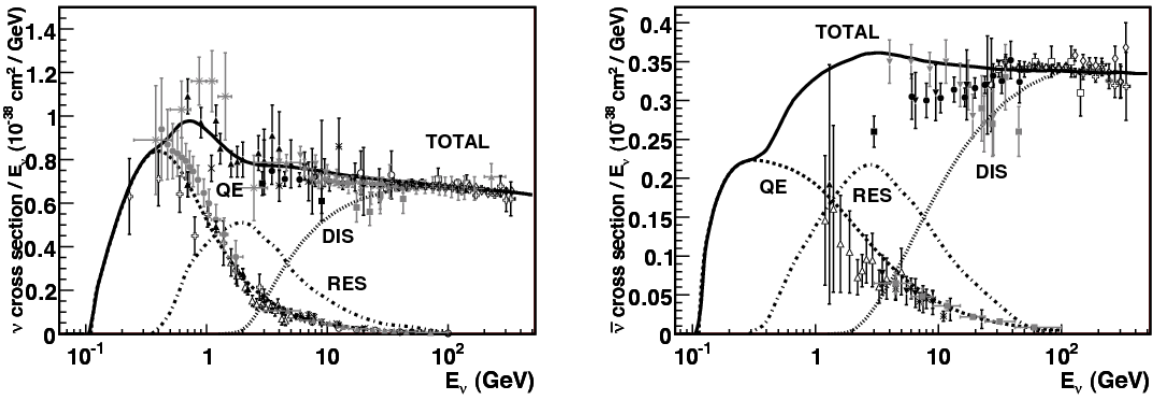

Fig. 2.2: Cross section of neutrinos (left) and antineutrinos (right) per nucleon for an isoscalar target. An isoscalar target is defined as a nucleus that contains the same number of protons and neutrons. Shown are the measurements by several experiments and predictions based on the NUANCE generator [23]. QE stands for quasi elastic scattering, RES for resonance production and DIS for deep inelastic scattering. [21]

It can be seen that for neutrino, charged current interactions the resonance production plays a major role in the energy range of about 1 GeV to 4 GeV. Above that energy scale, the deep inelastic scattering mechanism starts to dominate more and more. At the same time, quasi elastic scattering is the least dominant process in the energy range above a few single GeV,

which KM3NeT/ORCA is interested in. Additionally, the cross section increases linearly with energy above a few GeV. For tau neutrinos, the case is different, due to the high rest mass of the tau lepton. Figure 2.3 shows the total charged current cross section for tau neutrinos compared to muon neutrinos. Only at higher energies above a few hundred GeV, the charged

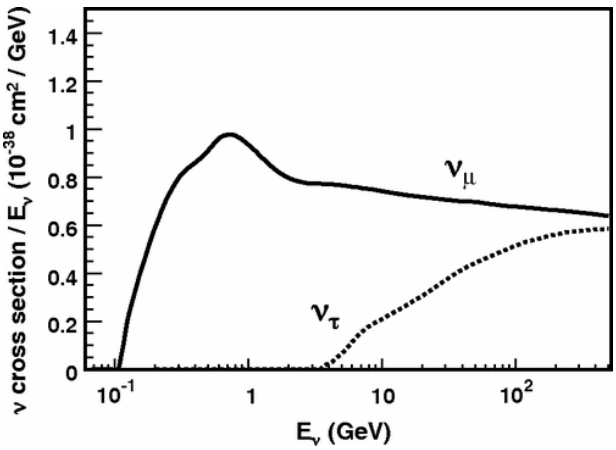

Fig. 2.3: Total charged current cross section per nucleon of muon neutrinos and tau neutrinos divided by neutrino energy as a function of neutrino energy. [21]

current cross section of tau neutrinos gets close to the one of muon neutrinos. For energies significantly below 100 GeV, the cross section of tau neutrinos is significantly lower than that of muon neutrinos.

2.5 Overview of other tau neutrino appearance experiments

Apart from KM3NeT/ORCA, there are also other experiments that study the appearance of tau neutrinos. In general, tau neutrino appearance detectors can be classified into two categories: atmospheric neutrino experiments, like KM3NeT/ORCA, and accelerator-based experiments.

Atmospheric neutrino detectors

Atmospheric neutrino detectors are typically Cherenkov detectors. When charged particles travel through a dielectric medium faster than the speed of light in this medium, they will emit electromagnetic radiation known as *Cherenkov radiation*. In the case of water, the Cherenkov light is emitted in a 42° cone. By measuring the Cherenkov light of the secondary charged particles in the neutrino interactions the properties of the initial neutrino can be reconstructed. Usually, these experiments employ a large, but sparsely instrumented target like water or ice. Hence, the detection of tau neutrinos is only possible on a statistical basis and not on an event-by-event one.

Till now, the highest tau neutrino statistics have been collected with the Super Kamiokande detector (SK). It is operating at the Kamioka Observatory, next to Hida-city, Gifu Prefecture, in Japan. More specifically, it is located inside the Mozumi mine and thus shielded by about 1000 m [24] of concrete rock (2700 m water equivalent [25]). Therefore, the atmospheric muon flux is reduced by about 1/200.000 [26], which greatly reduces the experimental background. SK is a water Cherenkov detector comprised by a 50 kiloton water tank, which is surrounded by more than 11,000 20-inch photomultipliers. The detector has been operating since 1996 and thus, more than two decades of data has been gathered. With the latest SK study from November 2017, the hypothesis of no ν_τ appearance has been rejected with a significance of $4.6\,\sigma$ [27]. The data sample includes 5,326 days of atmospheric neutrino data, where 338.1 ± 72.7 tau neutrinos have been measured. This translates into about 20 tau neutrinos per year.

Another experiment, which is similar to KM3NeT/ORCA, is the proposed future PINGU experiment within the IceCube collaboration. IceCube is a neutrino telescope located at the South Pole, which uses ice instead of water as the detector medium. IceCube has been operating since 2010, but it's too sparsely instrumented in order to efficiently detect tau neutrinos. PINGU is a proposed low-energy extension of IceCube, which would add a dense instrumentation to the central part of the IceCube array. According to the Letter of Intent, 3,000 tau neutrinos per year could be detected with PINGU [28].

Accelerator-based detectors

Other than observing the natural atmospheric neutrino flux, an accelerator-based neutrino beam can also be used in order to detect ν_τ appearance. Due to the high flux and the directionality of the neutrino beam, a high resolution detector with a small amount of target mass can be employed. One of these experiments is OPERA, which is located in the Gran Sasso Underground Laboratory, Italy. With OPERA, it has been possible to detect tau neutrinos on an event-by-event basis due to the high spatial resolution of the detector. Here, the tau neutrino has a very distinct signature due to the characteristic *kink* at the decay vertex of the secondary τ^- particle in the interaction, cf. figure 2.4. Thus, the background is significantly reduced and five ν_τ-like events result in a significance of $5.1\,\sigma$ on the ν_τ appearance [29]. However, five tau-neutrino-like events in four years of data are too less to probe the oscillation into ν_τ.

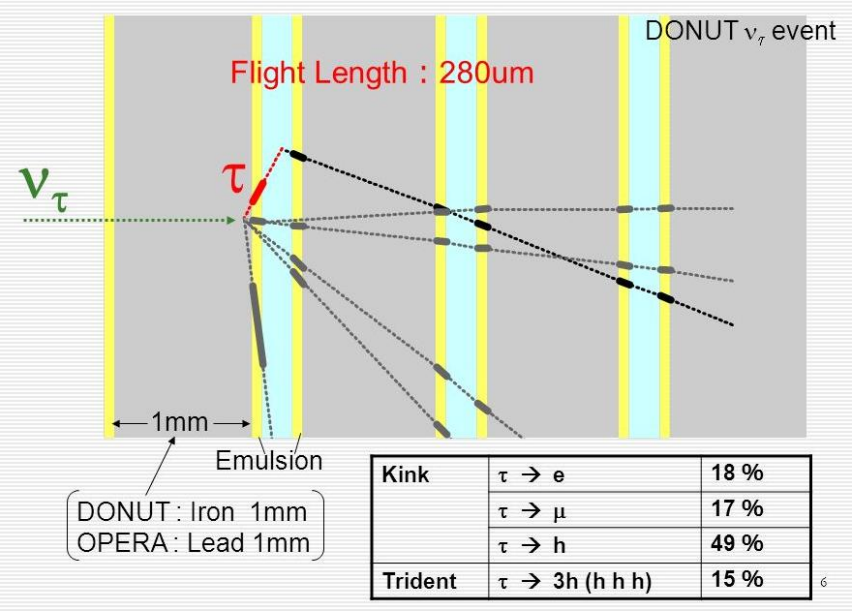

Fig. 2.4: Tau neutrino detection principle of the DONUT experiment, which is similar to the OPERA experiment. For the OPERA experiment, 1 mm lead sheets are used as a target and the emulsion in between the sheets can detect the charged secondary particles of a tau neutrino interaction. The secondary tau lepton in the tau neutrino interaction can be identified by the characteristic kink between the primary tau and its secondaries. [30]

CHAPTER 3

KM3NeT/ORCA: a deep sea neutrino detector

What is a scientist after all? It is a curious man looking through a keyhole, the keyhole of nature, trying to know what's going on.

— *Jacques-Yves Cousteau* —

KM3NeT [12] is a European neutrino telescope research infrastructure under construction in two locations of the deep Mediterranean Sea. KM3NeT/ARCA (**A**stroparticle **R**esearch with **C**osmics in the **A**byss) is the high-energy part of KM3NeT that aims to detect cosmic neutrinos with energies between several tens of GeV and PeV. For this purpose, the detector features a large volume of about one cubic kilometer. It is located about 100 kilometers off-shore of Sicily, Italy at a depth of about 3500 m. One of the first goals of ARCA is to measure the cosmic neutrino flux that has been observed by the IceCube neutrino telescope [31] in less than one year.

On the other hand, ORCA (**O**scillation **R**esearch with **C**osmics in the **A**byss) is the low-energy counterpart of ARCA, which aims to investigate fundamental properties of neutrino physics. The detector is located at a depth of about 2450 m, 40 km off-shore of Toulon in the south of France. ORCA is optimized for the detection of GeV atmospheric neutrinos with a lower neutrino energy threshold of a few GeV. Thus, the instrumentation of ORCA is significantly more dense compared to ARCA. The main goal of ORCA is to determine the neutrino mass hierarchy. Additionally, its large instrumented volume allows for precision measurements of the oscillation parameters. Likewise, the large instrumented volume of about 6 megatons of seawater makes it possible to collect an unprecedented amount of tau neutrino statistics.

3.1 Layout of the detector

Both KM3NeT/ARCA as well as KM3NeT/ORCA are water Cherenkov detectors and thus the Cherenkov light of secondary particles from neutrino interactions has to be detected in the deep sea. To that end, the detector volume of ORCA is instrumented with 115 Detection Units (DUs), also called *lines*. The DUs are string-like structures, which are anchored to the seabed and held upright by buoys at the top of the DUs. As of late 2019, four DUs are currently installed and operational. Each DU carries 18 Digital Optical Modules (DOMs) and each DOM, cf. figure 3.1 (right), is equipped with 31, 3" photomultiplier tubes (PMTs) in order to maximize effective photon detection area of a DOM. The PMTs are used to measure the arrival time and the time range that the anode output signal remains above a tunable threshold (time-over-threshold, ToT) with a time resolution on the nanosecond scale. The ToT is a proxy for the amount of light registered by the PMT. A scheme of the KM3NeT/ORCA detector is shown in figure 3.1 (left). The vertical spacing between the DOMs on a single detection unit is about 9 m, while the average horizontal spacing between the lines is about 20 m.

The main background of KM3NeT/ORCA consists of atmospheric muons, reaching the detector from the top, and randomly correlated noise by ^{40}K beta decays or bioluminescent animals. Atmospheric muons are created, when highly energetic, cosmic particles impinge on the atmosphere, cf. chapter 4. Most of them are stopped by the seawater, but some still reach the detector, producing light signatures very similar to ν_μ^{CC} interactions. Additionally, the expected rate of

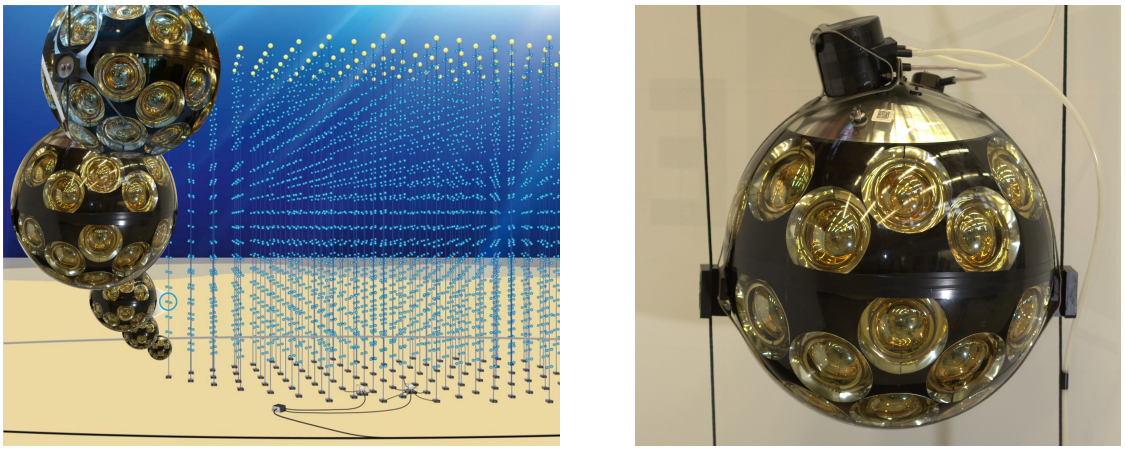

Fig. 3.1: Left: scheme of the KM3NeT/ORCA detector. Right: a KM3NeT DOM equipped with 31, 3" PMTs.

atmospheric muons is about 10^5 times higher than the expected interaction rate of GeV neutrinos within the detector volume [22]. Since only neutrinos can enter the detector from the bottom, the atmospheric muon background can be efficiently reduced by a selection on the reconstructed vertex position and the reconstructed direction of the emerging particle trajectory. Randomly correlated noise by ^{40}K decays or bioluminescence, also called random noise from now on, accounts for about 10 kHz uncorrelated single noise per PMT with a rate of two-fold coincidences of about 500 Hz per DOM [12]. By employing dedicated triggering algorithms, cf. section 3.1.1, and classifier algorithms the background by random noise can be reduced very efficiently. The fully funded prototype phase of KM3NeT/ORCA, Phase 1, will employ six DUs in total and is foreseen to be finished in early 2020. With these first strings, the performance of the detector will be demonstrated. Subsequently, Phase 2, which is still subject to be funded, aims to construct the full 115 DU layout.

3.1.1 Monte Carlo simulations and trigger algorithms

Detailed Monte Carlo simulations (MC) of the detector response have been performed for three distinct types of triggered data, namely atmospheric muons, randomly correlated noise and neutrinos. A detailed introduction to the KM3NeT simulation package and the trigger algorithms can be found in the KM3NeT Letter of Intent [12]. For neutrinos, all three possible charged current interactions with nucleons and nuclei (ν_e^{CC}, ν_μ^{CC}, ν_τ^{CC}) have been simulated, while NC interactions are represented by ν_e^{NC} only, since the photon distribution in the detector does not depend on the flavor of the neutrino in a neutral current interaction.

For all three types of triggered data, the detector response has been simulated in detail. The horizontal distance between the DUs for the simulations used in this work is on average 23 m and hence 3 m larger than for the simulations used in [12]. The vertical inter-DOM spacing is set to 9 m, which was identified as optimal for the determination of the neutrino mass hierarchy in [12]. The final level of the simulated data consists of a list of time stamps and the measured ToTs for all PMTs in the detector. A signal in a PMT with a measured ToT is called a *hit*. In addition to the *signal* hits induced by the interactions of neutrinos and atmospheric muons in the sensitive detector volume, simulated *background* hits due to random noise are added such that the simulated, triggered data matches the real conditions as close as possible.

After the hit simulation, several trigger algorithms that rely on causality conditions are applied. For example, these can be coincident hits on the same DOM during a certain amount of time. After a trigger has fired, which defines a triggered *event*, all hits that have led to the triggering decision are labeled as *triggered* hits. Since the trigger algorithm is not fully efficient at identifying

all signal hits, all hits in a larger time window than defined by the triggered hits are stored for further analysis. Assuming a triggered event with t_{first} as the time of the first triggered hit and t_{last} as the time of the last triggered hit, all photon hits in the single PMTs are recorded in between this time range $[t_{\text{first}}, t_{\text{last}}]$. After this, the time window is enlarged by a fixed time margin: $[t_{\text{first}} - t_{\text{marg}}, t_{\text{last}} + t_{\text{marg}}]$. This time margin is defined by the maximum amount of time that a particle at the speed of light would need to traverse the whole detector. As a result, the total time window of triggered neutrino events in KM3NeT/ORCA is about 3 µs.

CHAPTER 4

Tau neutrino detection principle

The experimenter dealing with nature faces an outside and often hard world.
Natures' curriculum cannot be changed.

— *Martin Lewis Perl* —

THIS chapter gives an introduction about how tau neutrinos can be detected in KM3NeT/ORCA. ORCA is designed for the detection of GeV neutrinos. Figure 4.1 shows the main sources of the total neutrino flux over the various energy scales. Hence, the only significant origin of neutrinos at the GeV energy scale is the atmospheric neutrino flux, which is caused by cosmic rays impinging on the atmosphere.

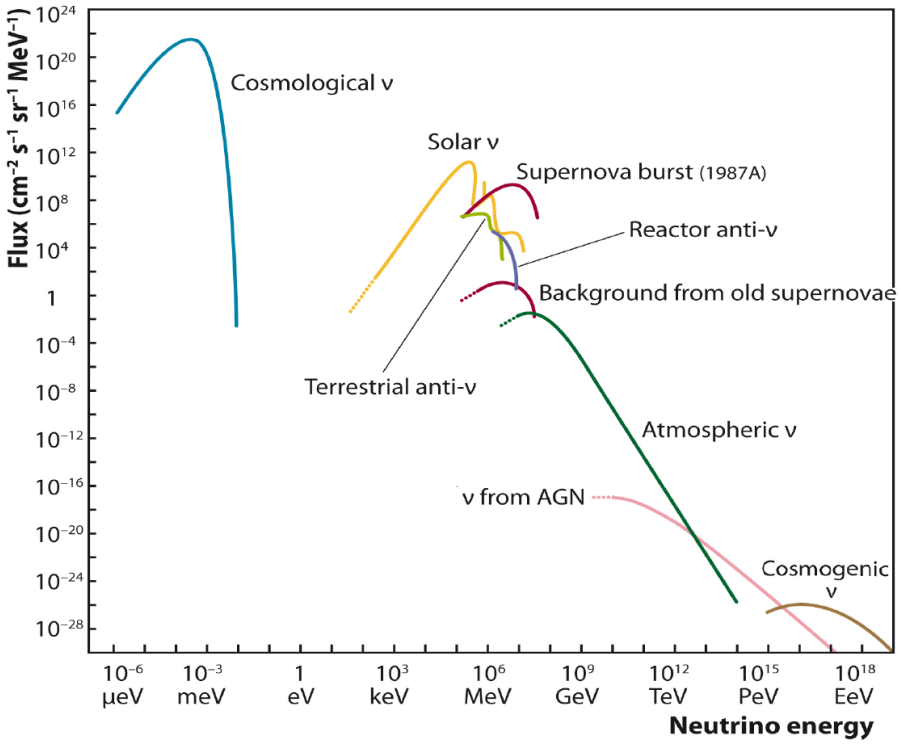

Fig. 4.1: Measured and expected fluxes of natural and reactor neutrinos [32].

4.1 The atmospheric neutrino flux as a source of tau neutrinos

Every day, the upper atmosphere of the Earth is impinged by a multitude of highly energetic cosmic ray particles. Cosmic rays primarily consist of protons, but also of a small amount of electrons and ions. When these particles collide with atoms in the atmosphere, the interactions produce hadronic and electromagnetic particle showers with a large amount of secondary particles. Among these secondaries are charged pions and kaons, whose decays create an atmospheric neutrino flux, cf. figure 4.2.

$$
\begin{aligned}
\pi^- &\to \mu^- + \bar{\nu}_\mu \to e^- + \bar{\nu}_e + \nu_\mu + \bar{\nu}_\mu && \text{(BR 1st: 99.9\%)} \\
\mathrm{K}^- &\to \mu^- + \bar{\nu}_\mu \to e^- + \bar{\nu}_e + \nu_\mu + \bar{\nu}_\mu && \text{(BR 1st: 63.6\%)} \\
&\to \pi^- + \pi^0 && \text{(BR: 20.7\%)} \\
&\to \pi^- + \pi^- + \pi^+ && \text{(BR: 5.6\%)} \\
&\to \pi^0 + e^- + \bar{\nu}_e && \text{(BR: 5.1\%)} \\
&\to \pi^0 + \mu^- + \bar{\nu}_\mu && \text{(BR: 3.4\%)} \\
&\to \pi^- + \pi^0 + \pi^0 && \text{(BR: 1.8\%)}
\end{aligned}
$$

Fig. 4.2: Decay channels of charged pions and kaons. The decays for the positively charged pions or kaons are the charge conjugated ones and the abbreviation "BR" specifies the branching ratio for the i-th decay. [33]

KM3NeT/ORCA can detect atmospheric neutrinos secondaries with energies from about 1 GeV to 100 GeV. Since intergalactic and interstellar magnetic fields can deflect charged cosmic ray particles on their way through the cosmos, it can be assumed that the cosmic ray flux incident on the Earth is for the most part isotropically distributed in this energy regime [33]. There are multiple models that predict the atmospheric neutrino flux. For this work, the models by the HKKM group [34] have been used. Figure 4.3 (left) shows the predicted atmospheric neutrino flux of muon and electron neutrinos for three different models in the GeV energy range.

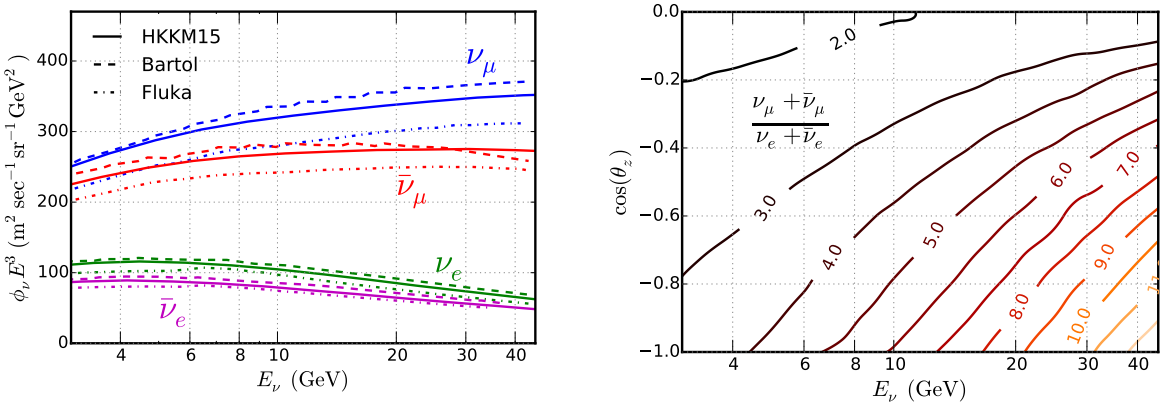

Fig. 4.3: Left: predictions of the atmospheric neutrino flux for electron and muon neutrinos as a function of the neutrino energy for different models [35]. Right: the total ratio of muon and electron neutrinos for neutrinos that cross the earth as a function of the neutrino energy and the cosine of the zenith angle [35].

It can be seen that the flux for neutrinos is slightly higher than for antineutrinos, especially in the case of muon neutrinos. Additionally, the ratio of the number of muon and electron neutrinos is between 2.5 and 3, similar to what one would expect by the decays shown in figure 4.2. However, the atmospheric neutrino flux that arrives at the KM3NeT/ORCA detector is different, since neutrino oscillations need to be factored in. This includes both neutrino oscillations in the air, where the oscillation probabilities in vacuum can be used as an approximation, as well as the neutrino oscillations in matter for neutrinos that travel through the Earth. Thus, the atmospheric neutrino flux that arrives at the KM3NeT/ORCA detector is highly dependent on the zenith angle and the neutrino energy. This can also be seen in figure 4.3 (right), which shows the total muon neutrino to electron neutrino ratio for neutrinos that cross the Earth based on the neutrino energy and the zenith angle. It can be seen that the ratio changes significantly based on the neutrino energy and the zenith angle, which is related to the oscillation length.

Other than that, no tau neutrinos are produced in the cosmic ray interactions introduced in figure 4.2. Likewise, the tau production rate of other, not mentioned atmospheric processes is negligible. Hence, the origin of atmospheric tau neutrinos must be based on neutrino oscillations from ν_e and ν_μ. Thus, an atmospheric neutrino flux model like the HKKM model needs to be combined with an additional neutrino oscillation model in order to estimate the atmospheric neutrino flux for the KM3NeT/ORCA detector site.

4.2 Modeling of neutrino oscillations for atmospheric neutrino fluxes

In order to derive a neutrino oscillation model, atmospheric oscillations have to be taken into account, as well as oscillations in matter from neutrinos that traverse the Earth. For atmospheric neutrino oscillations, the influence of the atmosphere for neutrinos above 1 GeV on the oscillation probabilities is sufficiently small, such that the oscillation probabilities in vacuum can be used. This also applies to the case of neutrino oscillations in water. For neutrino oscillations in solid Earth matter, the density profile of the Earth needs to be considered. The density can be estimated with the PREM model [36] for example. Then, the probabilities for electron and muon neutrinos oscillating into tau neutrinos can be calculated, cf. figure 4.4.

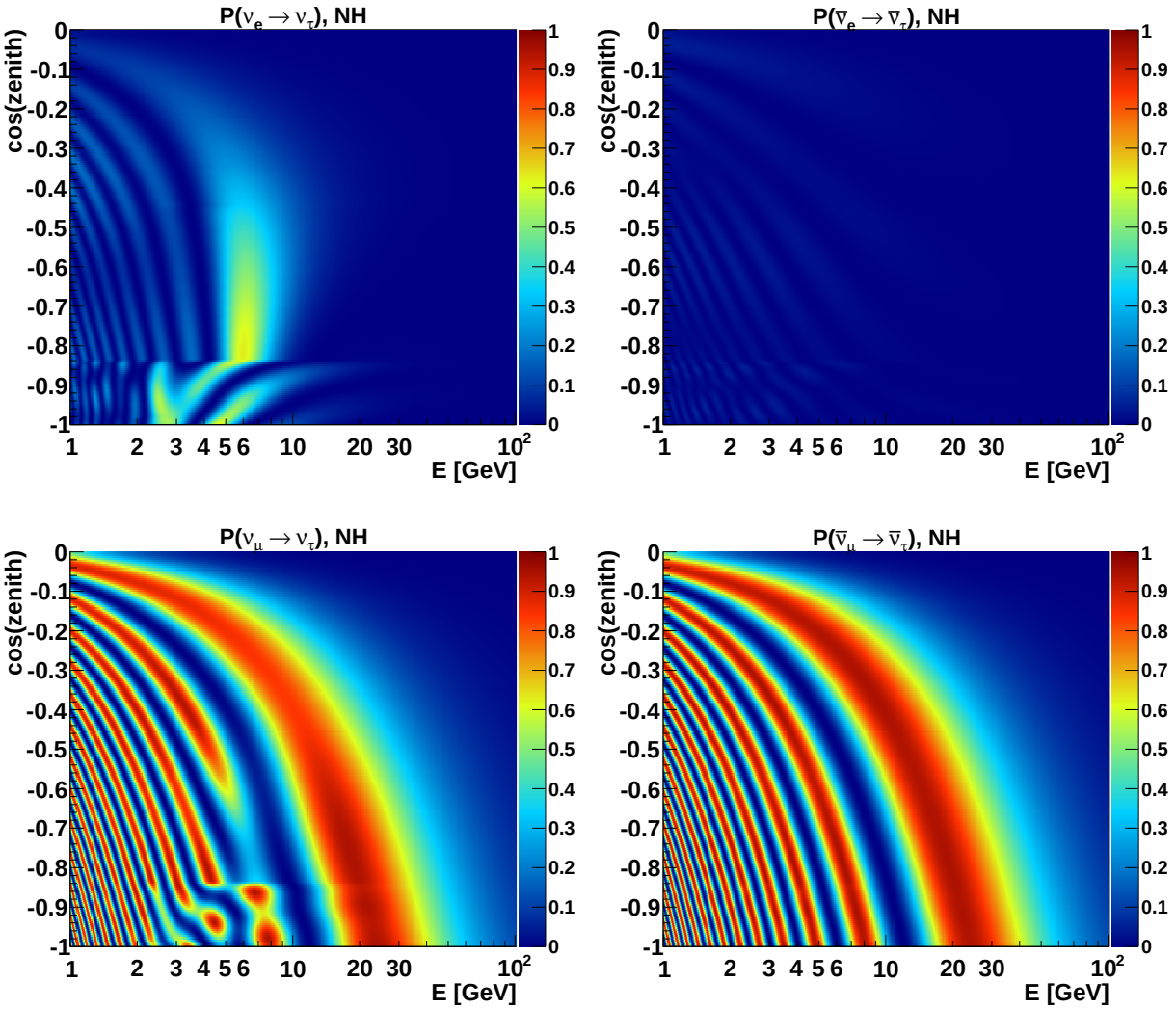

Fig. 4.4: Oscillation probabilities for (anti-) electron and (anti-) muon neutrinos into tau neutrinos for the normal mass hierarchy scenario. The probabilities are shown for up-going events in the KM3NeT/ORCA detector (cos(zenith) = 0 → horizontal, cos(zenith) = -1 → straight up-going). [37]

It can be seen that the oscillation probabilities for muon neutrinos into tau neutrinos are significantly larger than the ones for electron neutrinos. As already introduced, the ratio of atmospheric muon to atmospheric electron neutrinos is about 2.5 to 3. Thus, the main source of tau neutrinos in ORCA are $\nu_\mu \rightarrow \nu_\tau$ oscillations. Additionally, the oscillation probabilities for (anti-) muon neutrinos into (anti-) tau neutrinos are especially large in the 20 GeV to 30 GeV range for straight up-going neutrinos. Due to the high rest mass of the secondary tau lepton in ν_τ^{CC} interactions, the energy threshold for tau neutrino interactions is large, about 3.5 GeV. Thus, tau neutrinos with lower energies than this threshold can principally not be detected.

4.3 Topology of neutrino interactions in KM3NeT/ORCA

After having identified the source of tau neutrinos in KM3NeT/ORCA, the final question is how they can be detected in the KM3NeT/ORCA detector. For this purpose, the photon signature topologies of tau neutrino interactions in the detector need to be investigated. The different interaction types of GeV neutrinos in ORCA are shown in figure 4.5.

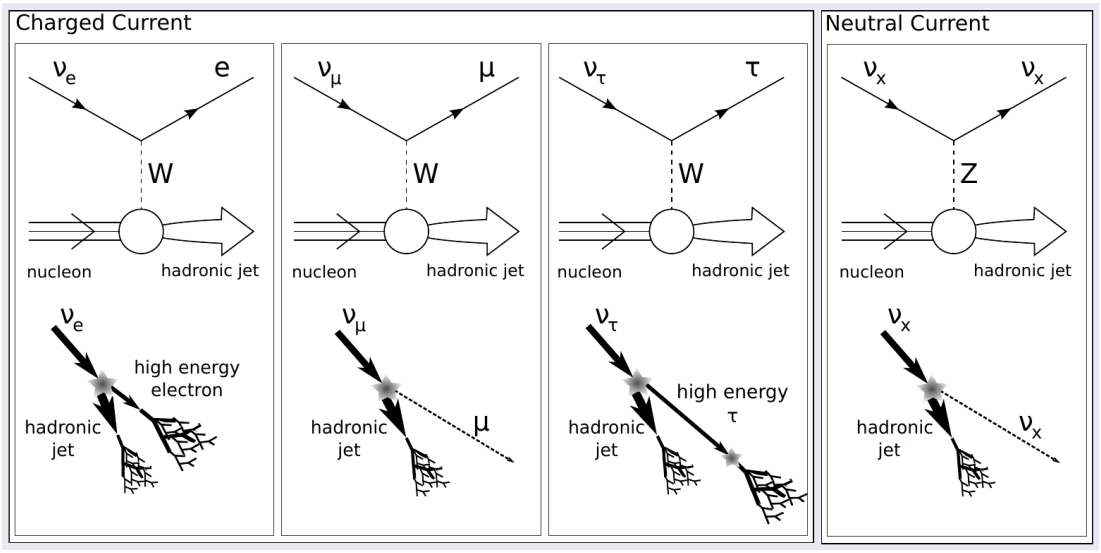

Fig. 4.5: Neutral current (NC) and charged current (CC) interactions of high energy neutrinos. A significant spatial separation of the outgoing τ lepton in the ν_τ^{CC} interaction is only given for very high neutrino energies or for detectors with a high spatial resolution. For tau leptons at the PeV scale, the tau decays after about $50\,\text{m} \times E_\nu/\text{PeV}$ [38], while the propagation length of tau leptons induced by GeV tau neutrinos is close to a millimeter. [39]

As introduced in section 2.4.2, the two dominant scattering processes are the resonance production for few GeV neutrinos and otherwise the deep inelastic scattering for neutrinos with higher energies. Neutral current interactions result in hadronic particle showers, while the outgoing, scattered neutrino cannot be detected. On the other hand, the secondary hadronic shower can be detected due to the Cherenkov light of the charged secondary particles. Since the cross section of the particles in the hadronic shower is large, the whole interaction is confined within a small volume. Hence, neutral current interactions of neutrinos in KM3NeT/ORCA look point-like, or also called shower-like. In charged current interactions, a hadronic shower is produced as well, but instead of an escaping neutrino a secondary lepton of the same family is created, carrying a part of the initial neutrino's energy. The fraction of the neutrino energy that is deposited in the hadronic shower can be described by the already introduced reaction inelasticity y. The topology of such an event is largely related to the mean free path of its charged secondary particle. For example, the attenuation length of a highly energetic electron is about 36 cm [40]. Compared to the average spacing between the KM3NeT/ORCA DUs, about 20 m, the path of the electron is confined within a very small volume. This also applies to the hadronic shower from the original electron

neutrino interaction as well as to the electromagnetic shower that is induced by the interaction of the secondary electron. Thus, charged current electron neutrino interactions appear as shower-like events in the detector.

On the other hand, muons created in charged current muon neutrino interactions have a track length of about $4.25\,\mathrm{m\,GeV^{-1}}$ [41]. Hence, dependent on the energy of the muon neutrino and the energy transfer in the interaction, the secondary muon can travel through large parts of the detector. As a result, ν_μ^{CC} events with muons as secondary particles can look shower-like, for a low energy muon, as well as track-like, for a high energy muon. Thus, the induced photon signatures of neutrinos in the KM3NeT/ORCA detector can be classified into two categories: events with a shower-like topology and events with a track-like topology. For tau neutrino interactions, the situation is more complicated, due to the different decay channels of the secondary tau lepton. A scheme of the different neutrino interactions in KM3NeT/ORCA, grouped into the shower-like and track-like categories, is shown in figure 4.6.

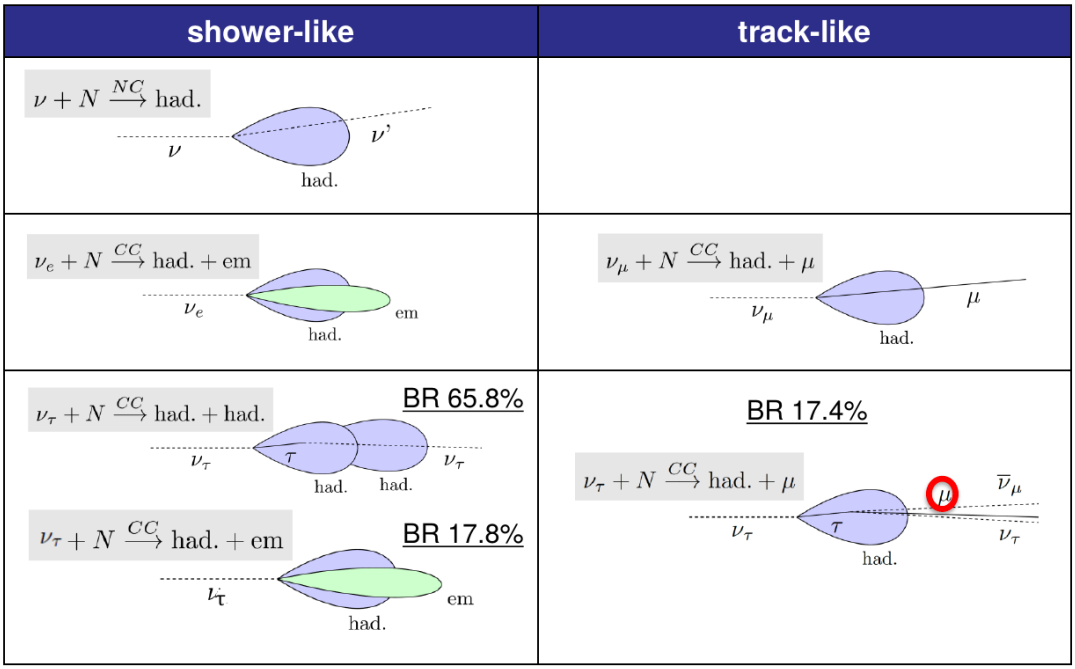

Fig. 4.6: Track-shower composition of the different neutrino interactions in KM3NeT/ORCA.

It can be seen that all charged current tau neutrino interactions produce a hadronic shower, similar to electron and muon neutrino charged current interactions. Additionally, a tau lepton with a short lifetime is created. For KM3NeT/ORCA, the tau lepton from a GeV tau neutrino interaction decays instantly due to the temporal and spatial resolution of the detector. The photon signature that the tau neutrino leaves in the detector depends on the decay channel of the tau lepton. The most prominent decay of the tau lepton is a hadronic one with a branching ratio of about 65.8%:

$$\tau^- \rightarrow \text{hadronic} \quad (\text{BR } 65.8\%)\,.$$

Thus, these tau neutrino interactions consist of two hadronic showers. In principle, one could identify this so-called *Double Bang* event due to the two individual hadronic showers, but this is not possible with the spatial resolution of KM3NeT/ORCA. Aside from this, the tau can also decay into an electron with a branching ratio of 17.8%:

$$\tau^- \rightarrow e^- + \bar{\nu}_e + \nu_\tau \quad (\text{BR } 17.8\%)\,.$$

The resulting signature is similar to the one of a ν_e^{CC} interaction. Hence, this interaction looks

shower-like as well. The situation is different though, if the tau lepton decays into a muon:

$$\tau^- \rightarrow \mu^- + \bar{\nu}_\mu + \nu_\tau \quad (\text{BR } 17.4\%)\,.$$

This tau decay has a branching ratio of 17.4% and due to the outgoing muon, the event can look track-like. Based on these three different decay channels of the tau, it can be seen that at least 80% of all tau neutrino interactions appear as shower-like events in the detector. Keeping in mind that most of the tau neutrinos in KM3NeT/ORCA are also coming from muon to tau neutrino oscillations, leads to the conclusion that tau neutrinos can be detected as a statistical excess of shower-like events compared to the non-appearance scenario.

As a result, a classification algorithm which can distinguish track-like from shower-like events, based on their different light topologies in the detector, needs to be developed. Before this can be done, the background consisting of atmospheric muons and randomly correlated noise needs to be filtered with a background classification algorithm. And at last, the properties of the recorded neutrinos, like the energy or the direction, need to be reconstructed. All of these tasks can be tackled by employing machine learning techniques, which will be introduced in the next chapter.

CHAPTER 5

Introduction to Machine Learning

May not machines carry out something which ought to be described as thinking but which is very different from what a man does?

— *Alan Turing* —

Statistical learning, or also called *machine learning*, is a subset of computer science that has been existing for a long time. Already in 1951, Marvin Lee Minsky and Dean Edmonds built the first artificial neural network machine, the "Stochastic Neural Analog Reinforcement Calculator" [42]. Since then, major computational advances and new statistical methods have allowed for huge leaps in the field. Nowadays, machine learning is present in many domains of human life, from self-driving cars and language processing to medicine and even board games.

Machine learning techniques can be classified into two categories [43]:

- **Supervised learning**
  The aim of supervised learning is the prediction of specific properties, called *labels*, for a given dataset using a machine learning technique. Thus, a function $f(x_i)$ needs to be derived that maps the input space x_i to the output space, the labels y_i. For this purpose, a set of *training* data with already known labels is needed.
- **Unsupervised learning**
  In unsupervised learning, no labels y_i are handed to the algorithm. Thus, the task is to find certain patterns and structures inside the input space. Consequently, this approach alleviates the problem of the need for labeled data.

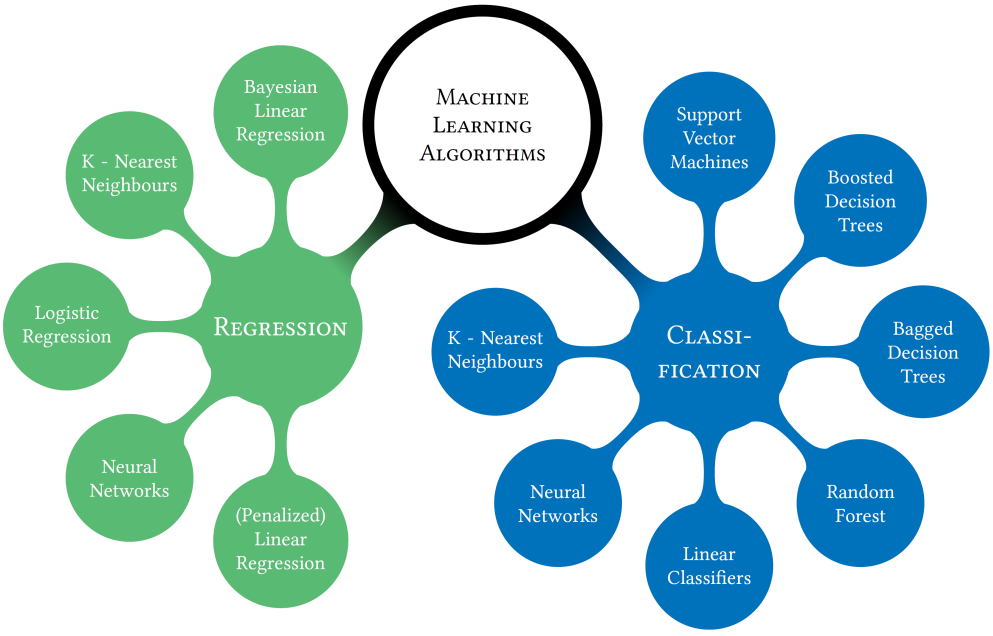

Fig. 5.1: Overview of various supervised learning algorithms. [44]

In supervised learning, the need for labels can pose a huge problem, since most data is unlabeled. Labeling a large enough training data set, e.g. with human-based labeling, can be very costly or even impracticable. In contrast to many application fields of computer sciences, physics experiments, like KM3NeT/ORCA, often have access to elaborate simulations that alleviate the problem. For this approach though, it needs to be demonstrated that the differences between the simulated data and the real data are small enough such that they can be controlled sufficiently well.

In unsupervised learning, it proves to be difficult to make the algorithm learn a specific property of the data, since no labels that should be reconstructed are given to the algorithm. In addition, it is nontrivial to extract the learned information out of the unsupervised learning algorithm. Thus, both techniques have their use cases depending on the application.

5.1 Shallow Learning

In order to study tau neutrino appearance with KM3NeT/ORCA, machine learning techniques can be employed to distinguish background from signal, to derive the track-shower composition of neutrino events and to reconstruct the energy and direction of neutrinos. It has not been clarified till now that most conventional machine learning algorithms cannot be fed with the raw data as inputs x_i. Therefore, it is necessary to design various high-level *features*, which can be extracted from the raw data. Afterwards, these *handcrafted* features can be used as an input for a machine learning algorithm, cf. figure 5.2.

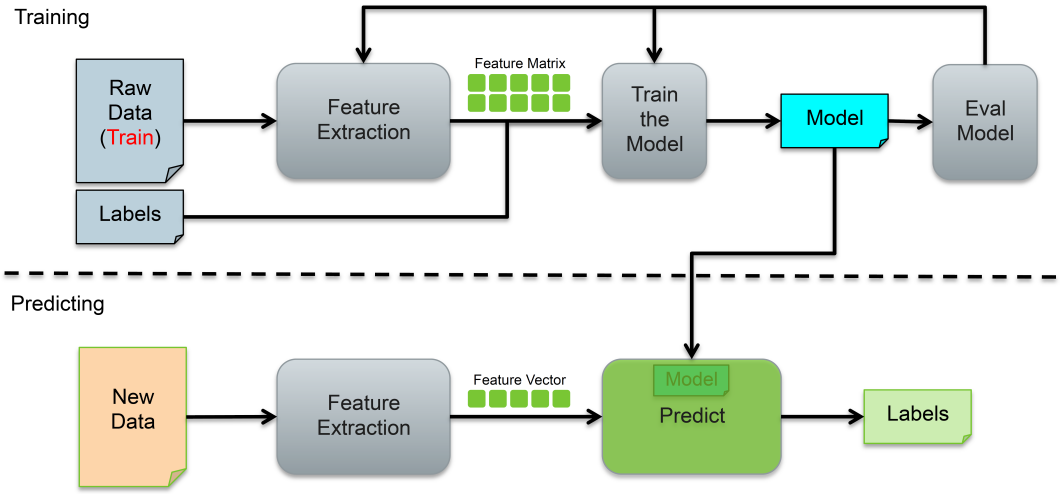

Fig. 5.2: Typical workflow in supervised learning for a shallow machine learning approach. [45]

From now on, this approach will be referred to as *shallow learning*. There is no rigid definition of this term, but in this work it will be used in order to distinguish between shallow machine learning approaches with handcrafted features and the to be introduced deep learning approaches, where the features are learned by the algorithm itself.

In both cases, there is a multitude of machine learning techniques that can be employed. Suppose that we have a training set with data points $x_i \in X$ and labels $y_i \in Y$. Then, the goal is to find an approximation $\hat{f}(x)$ to the function $f(x)$ that describes the relationship between the inputs x_i and the outputs y_i. In most scenarios though, the relationship between (X,Y) is not a fully deterministic one, i.e. $y = f(x)$ does not hold. Instead, there will be some kind of noise ϵ with $E(\epsilon) = 0$. If we then make the assumption that the error is independent and identically distributed, we get:

$$y = f(x) + \epsilon\,. \tag{5.1}$$

Now, the purpose of a machine learning algorithm is to approximate the function $f(x)$ with $\hat{f}(x)$ as good as possible. For example, this can be done by measuring the mean squared error (MSE) and thus minimizing $(y - \hat{f}(x))^2$. Since the error ϵ has a variance σ_ϵ^2, the approximation will never be perfect. However, we can derive a relation for the prediction error of the fit $\hat{f}(x)$ [43]:

$$\begin{aligned}
\mathrm{Err}(x) &= E\left[\left(y - \hat{f}(x)\right)^2\right] \\
&= \sigma_\epsilon^2 + \left[E\hat{f}(x) - f(x)\right]^2 + E\left[\hat{f}(x) - E\hat{f}(x)\right]^2 \\
&= \sigma_\epsilon^2 + \mathrm{Bias}\left(\hat{f}(x)\right)^2 + \mathrm{Var}\left(\hat{f}(x)\right) \\
&= \text{Irreducible Error} + \mathrm{Bias}^2 + \text{Variance}\,.
\end{aligned}$$

This is called the *bias-variance decomposition*, where both the bias and the variance prevent a learning algorithm from generalizing beyond the training dataset. Thus, apart from the training dataset, it is necessary to evaluate a machine learning algorithm on a separate set of data, which is commonly called the *validation* dataset. Since the algorithm is typically optimized over time and thus tailored to the validation dataset, the final performance is assessed on a different dataset, the *test* dataset, which is only being investigated after all optimization has been finished.

5.1.1 Decision Trees, Random Forests and Gradient Tree Boosting

For the approximation of $f(x)$, a multitude of machine learning methods exist. A particularly popular concept in shallow learning are so-called decision trees. A decision tree consists of multiple *nodes*. The first node inside a tree is called the *root*, nodes with at least one child are called *internal nodes* and nodes without children are called *leafs*. In a simple decision tree, each internal node acts on a single input feature and forms a decision based on the value of the feature. The tree is then traversed from top to bottom, until a leaf is reached that contains the result of the decision tree algorithm. An example of this procedure can be found in figure 5.3.

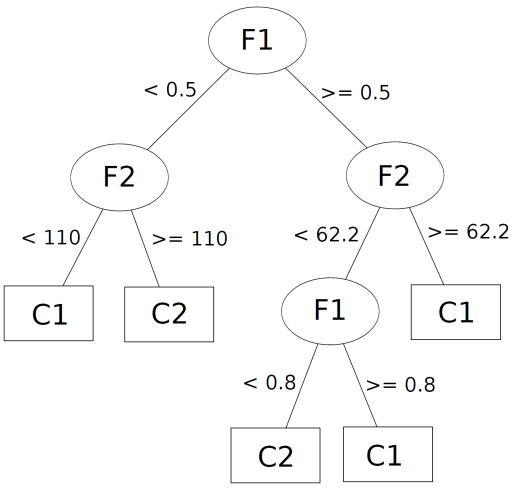

Fig. 5.3: Scheme of a simple decision tree with two features F1 and F2 and two classes C1 and C2. Each node takes one feature as an input and makes a single decision. The tree is traversed from top to bottom until a leaf with the result C1 or C2 is reached. [46]

Similar to many other machine learning techniques, decision trees cannot simultaneously reduce the bias and the variance down to zero. A large decision tree may reduce the bias, but at the same time this increases the variance due to *overfitting* on the training data sample, i.e. the performance of the decision tree is then better on the training dataset compared to the validation dataset. Thus, decision trees are often categorized as a low-bias but high-variance method.

There are multiple approaches in order to improve upon the decision tree learning scheme and one of them are random forests. Random forests [47] feature two main changes to the decision tree scheme in order to make the model less prone to overfitting, which reduces the variance of the predictive error. The first step in getting a more robust model is to use the bootstrap aggregating technique, also called *bagging*. Instead of training a single decision tree, random forests employ an ensemble of multiple decision trees during the training. With this procedure, the training dataset (X,Y) is bootstrapped into b_i smaller data samples (X_b, Y_b), and for each of these samples a tree $f_b(x)$ is trained. Then, the final learning result can be formed by averaging, called *majority voting*, over all the single trees f_b:

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x) \,. \tag{5.2}$$

With this training scheme though, there is the risk that some features which are highly dominant over other features can lead to highly correlated trees f_b. Hence, only a random subset of all features is available for the tree f_b during each training step. This is commonly refereed to as *feature bagging*. Thus, each tree f_b overfits the data in a different way, such that these differences can be averaged out during the voting on the final result.

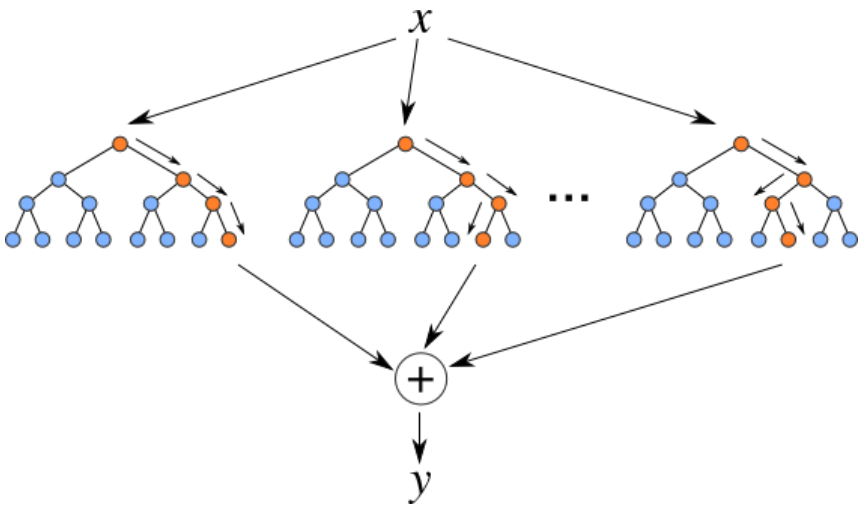

Fig. 5.4: Simplified random forest training scheme. The final result can be derived by majority voting of the single decision trees which have been trained with the bootstrap aggregating method. [48]

These random decision forests are also currently used in KM3NeT/ORCA as the standard technique for distinguishing between track- and shower-like neutrino events and for separating backgrounds from the neutrino signal. For example, features could be the sphericity of an event or the time distribution of the photon hits in the detector.

Another strategy in order to improve upon decision trees is the *boosting* method. Similar to the random forest, the boosting technique is an ensemble learning method, where the outputs of various trees are combined for the final prediction. This time though, the trees are trained in a sequential way, i.e. one by one, and not in parallel as for random forests. In addition, instead of using fully sized decision trees, like in random forests, small trees are used. Hence, the single learners are initially weak, with high bias and low variance. The main idea is to improve these weak learners and thus to lower the bias by telling each learner the errors of the previous learner. This approach is called *boosting*, due to the boosting of the initially high bias of the trees to a lower one. For this method, it is vital that the initial learners are weak and not already overfitting, since otherwise, there won't be any errors for the subsequent learners to build upon.

For any of these shallow machine learning techniques though, the performance of the algorithm is limited by the quality of the input features. However, most of the times the well thought out features created by a human do not represent the full feature space that is needed for a perfect approximation of $f(x)$. Fortunately, computational advances in the last few years have made it possible to employ deep learning approaches, in which the features are also learned by a machine learning algorithm.

5.2 Deep Learning

Similar to the term "Shallow Learning", there is no strict definition of the term "Deep Learning". Usually, the deepness refers to artificial neural networks with many hierarchical layers, though the main discrimination between shallow learning and deep learning is the feature learning process in deep learning models. In the past, neural networks were mainly employed with shallow architectures, which was partly caused by computational limitations. In the last years though, the huge performance leap by general-purpose computing on graphics processing units (GPGPU) has brought the computational power of supercomputers to desktop PCs. This leap made it possible to employ deep neural networks.

5.2.1 Neural networks

Neural networks, often also called *feedforward neural networks* or *multilayer perceptrons*, serve as the basis for most of the deep learning algorithms. Like any machine learning approach, neural networks are used in order to approximate a function $f(x)$ based on a certain number of inputs $x_i \in X$. Contrarily to shallow models, deep neural networks can take the low-level, unprocessed data as an input and not only handcrafted features.

Neural networks are based on the concept of artificial *neurons* that are arranged in layers. For a fully connected neural network, each neuron in layer l is connected to each neuron in layer $l-1$. The number of neurons inside of a layer determines the *width* of the network and the number of layers the *depth* of a network. Stacking multiple layers of neurons can be interpreted as multiple, chained functions that are acting on the input X. For a two layer model with a depth of one and thus two functions $f^{(1)}$ and $f^{(2)}$ we get (the $\hat{}$ of $\hat{f}$ is neglected from this point on):

$$f(x) = f^{(2)}(f^{(1)}(x))\,.$$

Here, $f^{(1)}$ refers to the first layer in the network and $f^{(2)}$ to the second one. The first layer of a neural network is called the *input* layer, the intermediate layers are called the *hidden* layers and the last layer is called the *output* layer. An exemplary neural network with two hidden layers and a width of four is shown in figure 5.5.

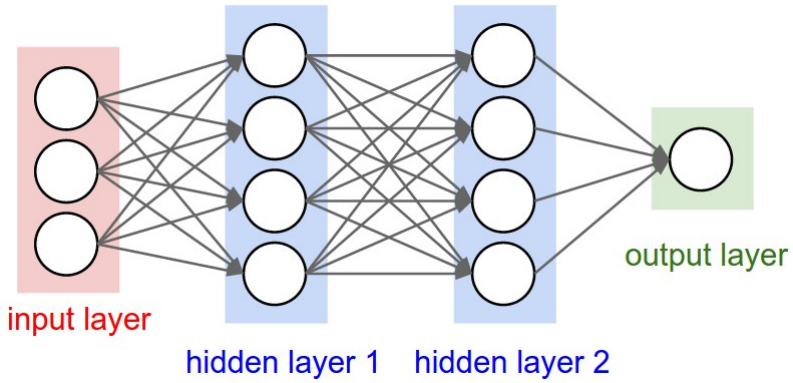

Fig. 5.5: Scheme of a fully connected neural network with two hidden layers and four neurons per hidden layer. [49]

In order to learn the relationship between (X, Y) during the training process of the neural network, learnable weights are used for each neuron. Suppose we have a single neuron with inputs x_i, then each input x_i of the neuron has a weight w_i associated to the input. Additionally, a single, learnable *bias* parameter is added in order to increase the flexibility of the model to fit the data.

This process in a neuron, consisting of the weights w_i and the bias b is called the *transfer function*:

$$f_\Sigma = \sum_{i=1}^{n} w_i x_i + b\,. \tag{5.3}$$

This equation models a linear neuron response with respect to the input data, whereas many processes in nature are inherently nonlinear. For this reason, the output of the transfer function can be wrapped in another, nonlinear function. Such functions are called *activation* functions. Nowadays, the most commonly used activation function is the *rectified linear unit* (ReLU):

$$f_{\mathrm{ReLU}}(x) = \max(0, x)\,. \tag{5.4}$$

The ReLU has the advantage of being it is computationally very efficient and that it does not saturate in the positive region. However, the downside to any nonlinear model is that most cost functions, like the mean squared error, become non-convex. Hence, convex optimization algorithms that guarantee the convergence of the training with any starting condition cannot be used. Thus, typically iterative, gradient descent-based optimization algorithms are used that just minimize the cost function until a low value is achieved. Altogether though, the benefits of using nonlinear activations outweigh the cons, such that they are nearly always used [50]. A scheme of a single neuron which puts together the transfer function and the activation function is shown in figure 5.6.

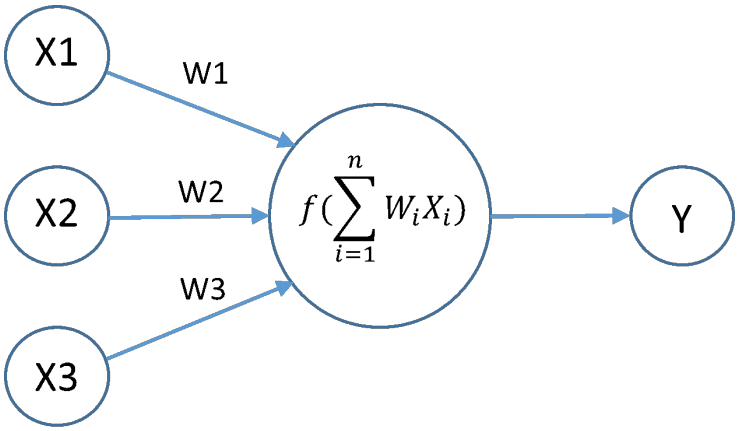

Fig. 5.6: Scheme of a single neuron in a neural network. The bias in the transfer function is neglected here and f depicts the activation function. [51]

5.2.2 The softmax classifier and its loss function

As already mentioned in chapter 5.1, a *cost* or also called *loss* function, upon which the model optimizes itself, is needed during the training. For regression tasks, a usual choice is taking the mean squared error or the mean absolute error.

For a categorical classifier, the desired output is 1 for the correct class and otherwise it is 0 for all the other classes. There are two popular classifiers: the support-vector machine (SVM) classifier and the softmax classifier. Since an SVM classifier is not used for the results of this thesis, only the softmax classifier is introduced in the following. The main concept of the softmax classifier is that the output should represent the probability distribution over n classes given some input.

Hence, the sum of the probabilities for each class should sum up to one. In fact, the softmax function is just a generalization of the logistic function, used in binomial logistic regression, to a multinomial problem. The softmax function can be defined as follows:

$$y : \mathbb{R}^K \to (0,1)^K \,, \tag{5.5}$$

$$y(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, ..., K \,. \tag{5.6}$$

It takes a K-dimensional vector $\mathbf{z}$ and maps it to a K-dimensional vector $y(\mathbf{z})_j \in (0,1)$, so that all the single elements y_j add up to one. In the case of a neural network, the softmax function can be interpreted as a layer, i.e. the output layer, with K neurons. The probability that the class is $t = j$ for a given input $\mathbf{z}$ is then as follows [52]:

$$\begin{bmatrix} P(t=1|\mathbf{z}) \\ \vdots \\ P(t=K|\mathbf{z}) \end{bmatrix} = \begin{bmatrix} y(\mathbf{z})_1 \\ \vdots \\ y(\mathbf{z})_K \end{bmatrix} = \frac{1}{\sum_{k=1}^{K} e^{z_k}} \begin{bmatrix} e^{z_1} \\ \vdots \\ e^{z_K} \end{bmatrix} , \tag{5.7}$$

where $P(t = j|\mathbf{z})$ is the probability that the class is j given the input $\mathbf{z}$. Based on these network outputs, we now need a method which can derive functions $\hat{f}$ that are good estimators for certain models. The following paragraphs are based on [52]. A common approach that can be used is the maximum likelihood principle. Suppose we have a set of parameters θ in our model. We would then like to maximize the likelihood that the parameters θ of the model result in the true prediction of the correct class $\mathbf{t}$ given some input $\mathbf{z}$:

$$\theta_{\text{ML}} = \underset{\theta}{\operatorname{argmax}} \, \mathcal{L}(\theta|\mathbf{t}, \mathbf{z}) \,. \tag{5.8}$$

The likelihood can be rewritten as the joint probability of generating $\mathbf{t}$ and $\mathbf{z}$ given the set of parameters θ [52]:

$$\mathcal{L}(\theta|\mathbf{t}, \mathbf{z}) = P(\mathbf{t}, \mathbf{z}|\theta) = P(\mathbf{t}|\mathbf{z}, \theta) P(\mathbf{z}|\theta) \,. \tag{5.9}$$

We are not interested in the probability of the data $\mathbf{z}$ and as such, this can be reduced to

$$\mathcal{L}(\theta|\mathbf{t}, \mathbf{z}) = P(\mathbf{t}|\mathbf{z}, \theta) \,. \tag{5.10}$$

Writing this for a fixed θ further simplifies the probability to $P(\mathbf{t}|\mathbf{z})$. Now, if we keep in mind that the t_i are dependent on the full $\mathbf{z}$ and that only a single class can carry a value of one in the $\mathbf{t}$, we get the following:

$$P(\mathbf{t}|\mathbf{z}) = \prod_{i=j}^{K} P(t_j|\mathbf{z})^{t_j} = \prod_{i=j}^{K} y(\mathbf{z})_j^{t_j} \,. \tag{5.11}$$

At last, maximizing the likelihood is equivalent to minimizing the negative logarithm of the likelihood, also called the log-likelihood, which is done for practical reasons:

$$-\log \mathcal{L}(\theta|\mathbf{t}, \mathbf{z}) = -\log \prod_{i=j}^{K} y_j^{t_j} = -\sum_{i=j}^{K} t_j \cdot \log(y_j) \,. \tag{5.12}$$

This function is called the *cross entropy* loss of the softmax classifier. It is possible to interpret this maximum likelihood procedure as a minimization of the differences between the true distribution t_j and the model distribution y_j. The justification for this can be derived by investigating the general cross entropy between a true distribution p and an estimated distribution q:

$$H(p, q) = -\sum_{x} p(x) \log q(x) \,. \tag{5.13}$$

This cross entropy can be rewritten in terms of the entropy and the so-called Kullback-Leibler (KL) divergence:

$$H(p, q) = H(p) + D_{\mathrm{KL}}(p \parallel q) \,. \tag{5.14}$$

The KL divergence gives a handle on the dissimilarity between the two discrete probability distributions p and q:

$$D_{\mathrm{KL}}(p \parallel q) = \sum_i p_i \log \frac{p_i}{q_i} \,. \tag{5.15}$$

Since $H(p)$ is independent of the model distribution q, minimizing the cross entropy is equivalent to minimizing the KL divergence between the true and the modeled distribution.

5.2.3 The backpropagation algorithm

At this point, we have derived our cost function C in the case of a regression task as well as in the case of a classification task. During the training, we now need to be able to calculate the gradients on the weights in our fully connected network based on the error of the model, which is measured by the cost function. The calculation of the gradient can be done by applying an algorithm called the *backpropagation* algorithm, which performs a *gradient descent*. The following paragraphs are based on [53]. Let w^l_{jk} be the weight for the connection between the neuron j in the layer l and the neuron k in the layer $l-1$.

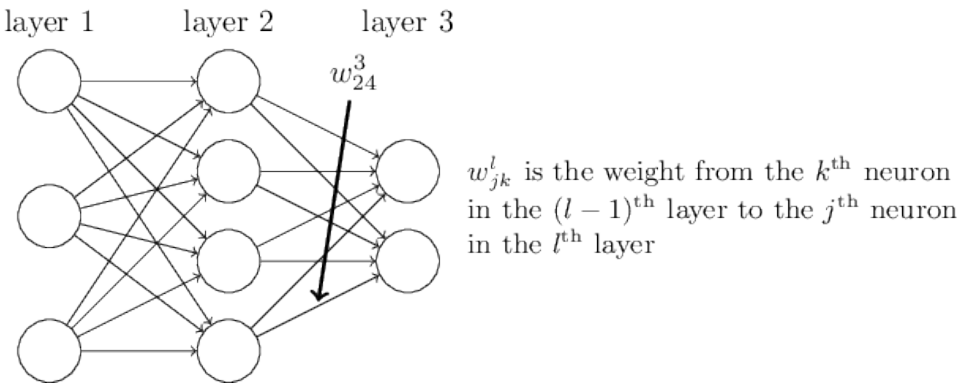

Fig. 5.7: Scheme of a fully connected network and terminology that is used for the explanation of the backpropagation algorithm. [53]

Then, the activation a^l_j of the neuron j in the layer l is given as follows:

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right) \,, \tag{5.16}$$

where σ is the activation function that is used. This can be rewritten in a matrix form:

$$a^l = \sigma\left(z^l\right) = \sigma\left(w^l a^{l-1} + b^l\right) \,, \tag{5.17}$$

where a^l is a vector with the components a^l_j, w^l is a weight matrix connecting to the l^{th} layer of neurons, b^l is a bias vector with the components b^l_j and z^l is called the *weighted input* to the neurons in layer l. The goal of the backpropagation algorithm is to find an expression for the partial derivatives of the cost with respect to the weights and biases:

$$\frac{\partial C}{\partial w} = ? \,, \quad \frac{\partial C}{\partial b} = ? \,. \tag{5.18}$$

For this, two assumptions have to be made. First, the cost function can be defined as an average

over the cost functions C_x for the individual training samples x.

$$C = \frac{1}{n} \sum_x C_x \,. \tag{5.19}$$

The reason is that the backpropagation algorithm only allows to compute the partial derivatives of the cost with respect to the weights and the bias for a single training example x. Second, it must be possible to write the cost as a function of the neural network's outputs:

$$C = C(a^L) \,, \tag{5.20}$$

where L is the total number of layers. As an intermediate step, it is useful to define the error δ_j^l of a neuron j in the layer l and δ^l represents the vector of errors for the layer l:

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} \,, \tag{5.21}$$

where z_j^l is the weighted input of the neuron j in layer l. The error in the output layer L can be calculated as follows:

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \sigma^{'}(z_j^L) \,, \tag{5.22}$$

where k is the sum over all neurons in the output layer and $\partial a_k^L / \partial z_j^L$ vanishes for $k \neq j$. Hence, the error in the output layer is a function of the dependence of how fast the cost is changing with respect to the output of the j^{th} output activation (first term) and how fast the activation function σ is changing dependent on z_j^L (second term). For simplicity, this can be rewritten again in a matrix form:

$$\delta^L = \nabla_{a^L} C \odot \sigma^{'}(z^L) \,, \tag{5.23}$$

where $\odot$ denotes the Hadamard product. In order to do the full backpropagation through all the layers of the network an additional expression is needed that specifies the error δ^l based on the error in the subsequent layer δ^{l+1}:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1} \frac{\partial \left(\sum_j w_{kj}^{l+1} \sigma(z_j^l) + b_k^{l+1} \right)}{\partial z_j^l} = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma^{'}(z_j^l) \,. \tag{5.24}$$

Rewritten in matrix form, this can be shortened to

$$\delta^l = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma^{'}(z^l) \,. \tag{5.25}$$

An intuitive interpretation of this formula is that by multiplying the error δ^{l+1} with the transpose weight matrix $\left(w^{l+1} \right)^T$, we move the error backwards through the network. After this, the Hadamard product is applied, moving the error backwards through the activation function in layer l, which gives the error for the layer l as a result. By combining equation 5.23 with equation 5.25, we can see that we can now compute the error for any layer l in the network. Finally, the derivative of the cost function with respect to the bias of the j^{th} neuron b_j^l in a layer l and the weights w_{jk}^l in the layer l can be related to error δ_j^l:

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \,, \tag{5.26}$$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} a_k^{l-1} = a_k^{l-1}\delta_j^l = a_{\text{in}}\delta_{\text{out}} \,. \tag{5.27}$$

All of the terms in these equations are already known and as such, the updates of the weights and biases in the network based on the cost can be calculated for every neuron.

5.2.4 Stochastic gradient descent

Now, the gradient descent mechanism can be used in order to train a neural network. Specifically, we want to minimize the cost function for the training dataset $x_i = 1, ..., n$, which is a sum over the costs for the single samples x_i:

$$C(\theta) = \frac{1}{n}\sum_{x_i=1}^{n} C_{x_i}(\theta) \,. \tag{5.28}$$

For this purpose, the weights can be iteratively updated as follows:

$$\theta_{j+1} = \theta_j + \Delta\theta_j = \theta_j - \eta \sum_{x_i=1}^{n} \frac{\nabla C_{x_i}(\theta_j)}{n} \,, \tag{5.29}$$

where θ_j is the set of model parameters after the j^{th} training iteration and η is called the *learning rate* or also *step size*. If the learning rate is set too high, the network never starts to learn, i.e. the loss never starts to decrease, or the loss even diverges. On the other hand, if the learning rate is too small, the network takes a long time to train. Moreover, the network can be trapped in a local minimum during the early stages of the training, due to the small step size. Often, the learning rate is annealed over time. In the beginning of the training, the network finds features that are easy to learn and it is found that high step sizes are tolerated. In the later stages of the training though, it can sometimes be observed that the network stops to improve if the learning rate is set too high. The reason is that the minima are more difficult to find during the later stages of the training, since the improvement in the loss is usually significantly smaller compared to the early stages of the training. If the step size is set too high then, the minimizer of the gradient descent procedure is not able to converge into a small minimum, since it will immediately jump out of the minimum in the next iteration, due to the high step size. Thus, the learning rate is a parameter, that has to be optimized based on the performance of the model. Such kind of parameters are commonly referred to as *hyperparameters*.

For large-scale machine learning problems, the calculation of the gradients for all of the training data samples that are required for a single parameter update can have a very high computational cost. To mitigate this problem, the true gradient over all samples can be approximated by calculating the weight update on a smaller *minibatch* of the data. This technique is called *stochastic gradient descent* (SGD). The size of the batch typically ranges from one to a few hundreds of samples and one iteration over the whole training dataset is called an *epoch*.

There are multiple extensions to the stochastic gradient descent method and one of them is the *momentum* concept. With this, the weight update gets modified by a constant momentum factor dependent on the weight update of the $j-1$-th iteration:

$$\theta_{j+1} = \theta_j - \eta \sum_{x_i=1}^{n} \frac{\nabla C_{x_i}(\theta_j)}{n} + \alpha\Delta\theta_{j-1} \,. \tag{5.30}$$

This method can be even further modified, by using the so-called *Nesterov* momentum [54]:

$$\theta_{j+1} = \theta_j - \eta \sum_{x_i=1}^{n} \frac{\partial C_{x_i}}{\partial\left(\theta_j + \alpha\Delta\theta_{j-1}\right)} \cdot \frac{1}{n} + \alpha\Delta\theta_{j-1} \,. \tag{5.31}$$

Due to computational considerations, the structure of equation 5.31 is slightly modified in most implementations.

5.2.5 Adam: adaptive moment estimation

Apart from this, other gradient descent algorithms for stochastic optimization similar to SGD exist, like Adam [55]. Adam is based on adaptive estimates of the first two moments of the gradient. For each timestep j and a single input x_i, the moving average of the first and the second moment is calculated:

$$m_j = \beta_1 \cdot m_{j-1} + (1 - \beta_1) \cdot g_j \,, \tag{5.32}$$

$$v_j = \beta_2 \cdot m_{j-1} + (1 - \beta_2) \cdot (g_j \odot g_j) \,, \tag{5.33}$$

$$g_j = \nabla_\theta C_j \left(\theta_{j-1}\right) . \tag{5.34}$$

m_j is the moving average estimating the first moment, i.e. the mean, v_j is the moving average estimating the second moment, the uncentered variance, and g_j is the gradient of the cost function C at timestep j with the parameters from timestep $j-1$. β_1 and β_2 are exponential decay rates of the moments with $\beta_1, \beta_2 \in [0, 1)$ and proposed default values [55] of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For the first timestep $j = 1$, the moving averages of the first and second moment of the previous timestep, m_0 and j_0, are initialized with zero. Thus, the moments are biased towards zero, especially in the first timesteps of the gradient descent. However, the bias can be corrected, yielding the bias-corrected, moving average estimates of the first and the second moment:

$$\hat{m}_j = \frac{m_j}{1 - \beta_1^j} \,, \;\; \hat{v}_j = \frac{v_j}{1 - \beta_2^j} \,. \tag{5.35}$$

With this, the rule for the update of the parameters θ can be expressed as follows:

$$\theta_{j+1} = \theta_j - \eta \cdot \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon} \,, \tag{5.36}$$

where ϵ is a constant factor with a proposed default value of 10^{-8}, preventing a division by zero. There is no theoretical reasoning into which gradient descent optimization algorithm performs best at all times. Thus, the question which optimizer performs best at a certain task has to be investigated on a case-by-case basis.

5.2.6 Weight initialization

Before the training of a neural network can start, the weights need to be initialized. For this purpose, one needs to think about the properties of the input data x_i at first. Often, the data is zero-centered before it is fed to the network, i.e. the mean of all the samples x_i is subtracted from each sample x_i. One reason for this is that the nonlinearity of many different types of activation functions is explicitly designed to be close to zero. This is especially true for the ReLU function but also for the hyperbolic tangent function, $\tanh(x)$, whose activation saturates quickly for higher $|x|$ such that the gradient vanishes. Hence, the mean of the input data is expected to be zero and, dependent on the data, it follows a specific distribution, usually a Gaussian one.

Keeping this in mind, a reasonable first approach could be to initialize all weights with zeros. However, this proves to be a mistake, since every neuron would compute the same output during the first forward pass of the data through the network. Therefore, all gradients during the backpropagation phase, and thus also all weight updates, would be the same. As a consequence, this symmetry in the network would never be broken.

However, in order to break the symmetry, the initial weights can be sampled from a zero-mean, Gaussian distribution. This works reasonably well, but choosing the correct width of the Gauss proves to be difficult. If the width is too small, it is possible that all activations might become zero

and thus the gradients would vanish. On the other hand, if the width is too large, the gradients might explode. It turns out that this issue is part of a more general problem: the distribution of the outputs from a randomly initialized neuron has a variance that grows with the number of inputs. In general, we would like that the distribution of the activations in a neural network looks similar for each layer. In this way, even for a network with many layers, the gradients would not die nor explode. A solution for this issue would be to normalize the variance of each neuron's output to one. In [56], Glorot et al. argue that a good initialization for linear activations can be derived as follows:

$$\mathrm{Var}[\theta] = \frac{2}{n_{\mathrm{in}} + n_{\mathrm{out}}}\,, \tag{5.37}$$

with n_{in} and n_{out} as the number of inputs in the previous and in the next layer. This is commonly referred to as the *Glorot* weight initialization However, the method is derived with the assumption of linear activations and not with nonlinear activations like ReLUs. A fix for this issue has been proposed by He et al. [57], who show that the initialization must be modified in the case of nonlinear activations with an additional factor of two, resulting in the so-called *He* weight initialization:

$$\mathrm{Var}[\theta] = \frac{2}{n_{\mathrm{in/out}}}\,. \tag{5.38}$$

Additionally, they show that if the initialization properly scales the backward signal, the initialization will also be properly scaled for the forward signal and vice versa. Thus, it does not matter if one uses the number of inputs n_{in} or the number of outputs n_{out} for scaling the variance.

5.2.7 Batch Normalization

Even with properly initialized weights, the change of the distributions of the activations in the network, due to the optimization of the network parameters during the training, is still an issue. This problem is also commonly referred to as the *internal covariate shift*. A novel approach to this issue to explicitly normalize the inputs that the activation function gets. For this purpose, so-called *batch normalization* [58] layers are added in between the fully connected layer and the nonlinearity, i.e. after the transfer function. For each batch B containing m samples $B = \{x_{1\ldots m}\}$ during the training, the mean and variance is calculated:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i\,, \tag{5.39}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2\,. \tag{5.40}$$

Then, the normalization can be applied:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\,, \tag{5.41}$$

where ϵ is a constant for numerical stability. At last, two learnable parameters γ and β are added, such that the network can freely shift and scale the output:

$$\mathrm{BN}_{\gamma,\beta}(x_i) = \gamma\hat{x}_i + \beta\,, \tag{5.42}$$

where $\mathrm{BN}_{\gamma,\beta}$ is the output of the batch normalization layer dependent on the inputs x_i and the parameters γ and β. Notice that the output of the batch normalization layer is dependent on the statistics of the current batch of samples. This is fine during the training stage, but when the trained network is applied to data, also referred to as *inferencing*, the output should only depend on the input. Additionally, we also want to be able to inference on single samples of data and not only on multiple ones in a batch. Thus, the batch normalization layer acts differently during

the training and the inferencing stage. The difference is that we now use the full population of the training dataset for the calculation of the mean and the variance during any inferencing. In practice, the population mean and variance can already be calculated efficiently during the training stage by storing a moving average, which is then kept for the inferencing stage. Nowadays, batch normalization layers are widely used in deep neural networks, for example since they can make the networks more robust to a bad initialization.

5.2.8 Dropout

During the training, it is often observed that a network performs better on the training data compared to the validation data. This is commonly referred to as *overfitting* on the training data. Methods which try to prevent overfitting are usually called *regularization* techniques. A simple approach to this problem is the so-called *Dropout* technique, which can be implemented as a layer in the network. Any input to the dropout layer has a probability p to be set to zero in the output of the layer. In this way, many sub-networks of the original network are created during the training process. The amount of dropped inputs can be controlled with the dropout rate p, which is a hyperparameter that is set before the training starts. A scheme of this regularization method is shown in figure 5.8.

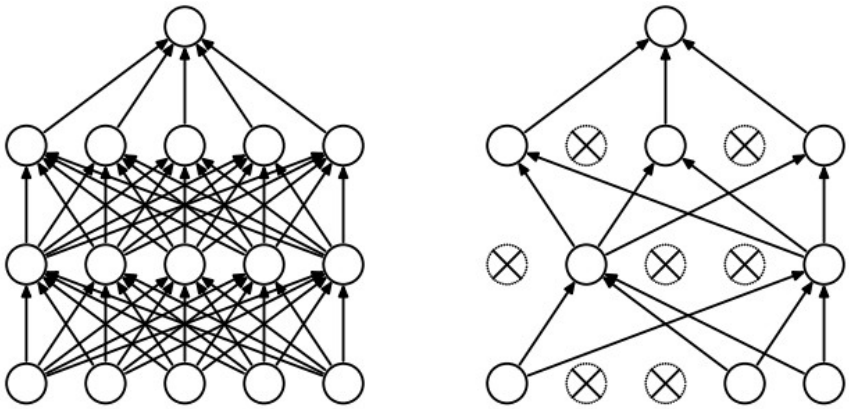

Fig. 5.8: Example of the dropout technique applied to a fully connected neural network. [59]

It is crucial that the dropout layer behaves differently during the training and the inferencing phase. For the training stage, a certain number of inputs is dropped during the forward pass of the input and the backward pass is then executed based on the forward pass with the dropped neurons. For the inferencing stage though, no neurons are dropped in order to achieve the maximum performance. However, the output of the neurons in the inferencing should be the same as their expected outputs during the training. Hence, the output of the hidden layers is scaled by a factor of p for the inferencing stage. It can be shown that this process is related to computing the ensemble prediction of all the many sub-networks that the dropout technique creates [60].

5.2.9 Convolutional Neural Networks

As already mentioned in section 5.2.1, there are more sophisticated network architectures that build upon fully connected networks. One of them are *convolutional* neural networks (CNNs). Since 2011, the error rates in popular image classification challenges have improved by nearly a factor of ten [61] due to the advent of deep learning techniques and especially due to specialized networks like convolutional neural networks. CNNs are typically used in domains, where the input can be expected to be image-like, i.e. in image or video classification. In this regard, several changes are made to the architecture of CNNs compared to fully connected neural networks. The main concepts of convolutional neural networks are based on only locally and not fully

connected networks and on parameter sharing between certain neurons in the network. The simplest convolutional neural networks are built using three different layer types: convolutional layers, pooling layers and fully connected layers.

Convolutional layers

Convolutional layers are the main building blocks of CNNs, which do most of the feature learning. The parameters of a convolutional layer consist of a set of learnable *filters*, composed of weights, that are able to learn features from the input.

Let's consider 2D images as the input of a convolutional neural network. Since most images are colored, they are actually three-dimensional: with width, height and channel information. Here, the channel dimension specifies the brightness for each red, green and blue (RGB) color channel of the image. This three-dimensional array is used as an input for the first convolutional layer. A visualization of a 4×4 RGB image is shown in figure 5.9.

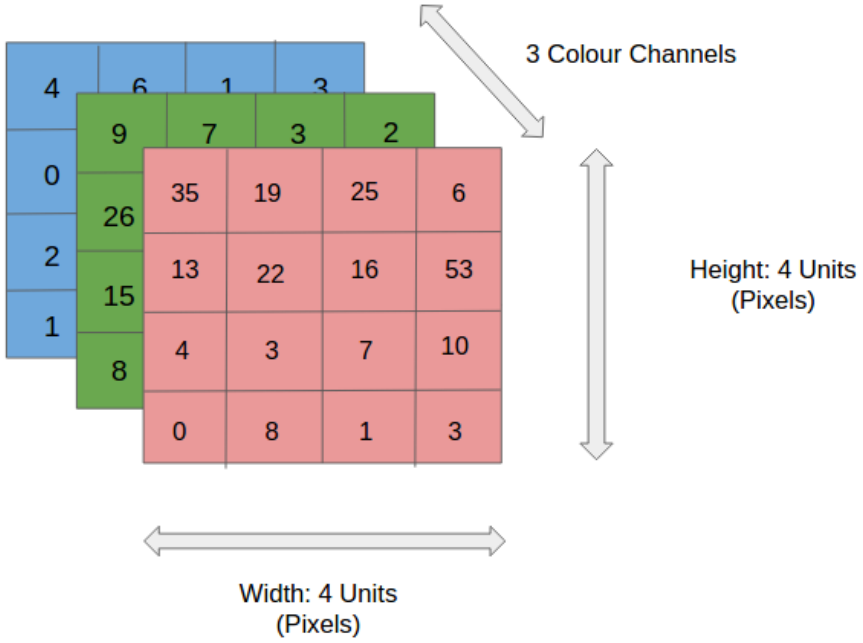

Fig. 5.9: Representation of a 4×4 RGB image as a three-dimensional array. The brightness of each channel is indicated with numbers ranging from 0 to 255. [62]

Similar to the input layer, the neurons in a convolutional layer are arranged in three dimensions, i.e. in width, height and depth. The term depth that is used here should not be confused with the depth of fully connected networks, which can be related to the number of its layers.

One of the main differences between convolutional layers and fully connected layers is that the neurons inside a convolutional layer are only connected to a local region of the input volume. This is often called the *receptive field*, the *filter size* or also the *kernel size* of the neuron. The connections of this local area to the neuron can be local in space (width, height), but they are always full along the depth of the input volume. Suppose that we have a $[32 \times 32 \times 3]$ image (width, height, channel), where each neuron in the first convolutional layer is connected to a local $[5 \times 5 \times 3]$ (width, height, depth) patch of the input. For each of these connections, a weight is assigned, such that each neuron has a $[5 \times 5 \times 3]$ weight matrix, which is often called the *kernel*. These weights are now used in performing a dot product between the receptive field of the neuron and its associated kernel. Here, the total number of parameters for the single neuron would be $5 \cdot 5 \cdot 3 + 1$ (bias) $= 76$. Additionally, each neuron at the same depth level covers a different part of the image with its receptive field. However, the question is, how far the receptive field

of one neuron should be apart from the receptive field of the adjacent neuron. This behavior is controlled with the *stride* hyperparameter. If the stride is one in one dimension, the perceptive field of the adjacent neuron is slid by one pixel. If the stride is two, it is slid by two pixels and so on. As a result, strides larger than one will produce smaller output volumes. A visualization of this mechanism is shown in figure 5.10.

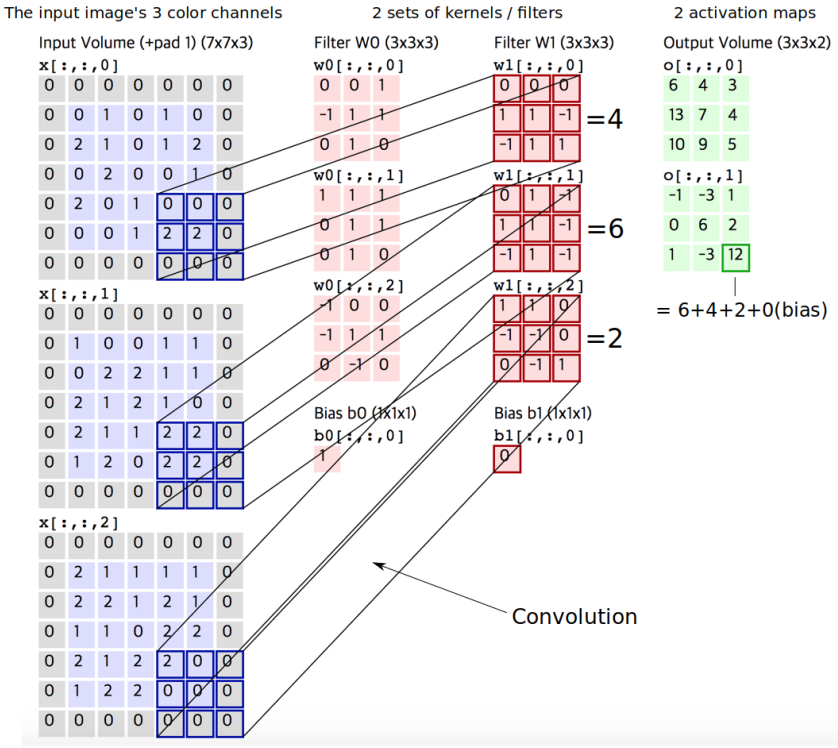

Fig. 5.10: Visualization of computing the dot product between a $[3 \times 3]$ filter (kernel) W_1 and a $[7 \times 7 \times 3]$ input volume. Here, a stride of two is used in the convolutions in order to generate the output volume. Taken from [63] and modified.

However, even for stride values of one, the output volume of the convolutional layer will be smaller than the input volume, if the kernel size is larger than one in one dimension. For example, if we have a $[4 \times 4]$ input volume and the kernel is of size $[2 \times 2]$, the output volume will be of size $[3 \times 3]$, neglecting the depth dimension. This is due to the fact that we can only slide the $[2 \times 2]$ perceptive field for three times until we reach the end of the input volume. Hence, the spatial dimensions of the original input image would be reduced with every convolutional layer, which leads to less parameters the more layers we have. In order to fix this problem, the borders of the spatial dimensions of the input can be padded with zeros. This is called *zero padding*. By padding zeros at the borders of the image, it can be achieved that the input volume has the same size as the output volume.

Based on all of these considerations, it is possible to calculate the size of the output volume O for each spatial dimension as a function of the input volume size I, the kernel size of the neurons K, the stride S and the amount of zeros that are used for the padding at each border P:

$$O = \frac{I - K + 2P}{S} + 1\,. \tag{5.43}$$

For the example in figure 5.10, with $I = 5$, $K = 3$, $P = 1$ and $S = 2$, the calculated result would be a $[3 \times 3]$ output volume, which is also shown on the right hand side of the image. Based on equation 5.43, it is possible to calculate the number of parameters in the model. Assuming more

realistic input images of size $[256 \times 256 \times 3]$, a kernel size of $[5 \times 5]$, a depth of 100, strides of one and no zero padding would result in an output volume of $[252 \times 252 \times 100]$. This translates to $252 \cdot 252 \cdot 100 \approx 6.3 \times 10^6$ neurons and each of these neurons would be connected to a perceptive field of $[5 \times 5 \times 3]$. Remembering that each neuron would then have $5 \cdot 5 \cdot 3 = 75$ weights plus one bias, yields the total number of parameters for this shallow, one layer CNN: $6.3 \times 10^6 \cdot 76 \approx 4.8 \times 10^8$. Hence, the number of parameters in the model explodes with a larger input volume.

However, for many types of images, the reasonable assumption can be made that a filter that is useful at some point (x_1, y_1) should also be useful at another point (x_2, y_2). For example, elementary features like edges are typically present in multiple areas of the image. For a convolutional layer, this assumption can be incorporated by letting all neurons in the same depth slice use the same weights. This is called *parameter sharing*. Thus, it makes sense to call the combination of all the neurons in a certain depth slice a filter. For example, a volume of $[32 \times 32 \times 10]$ neurons has a depth of ten and thus ten filters, where in each depth slice, i.e. $[32 \times 32 \times 1]$, the weights are shared across the single neurons. In the example from above, the convolutional layer would only have 100 sets of weights, one for each depth slice. Hence, the total number of parameters in the layer would be reduced to $100 \cdot (5 \cdot 5 \cdot 3 + 1) = 7600$ parameters. A visual representation of the parameter sharing technique in a depth slice can be found in figure 5.11.

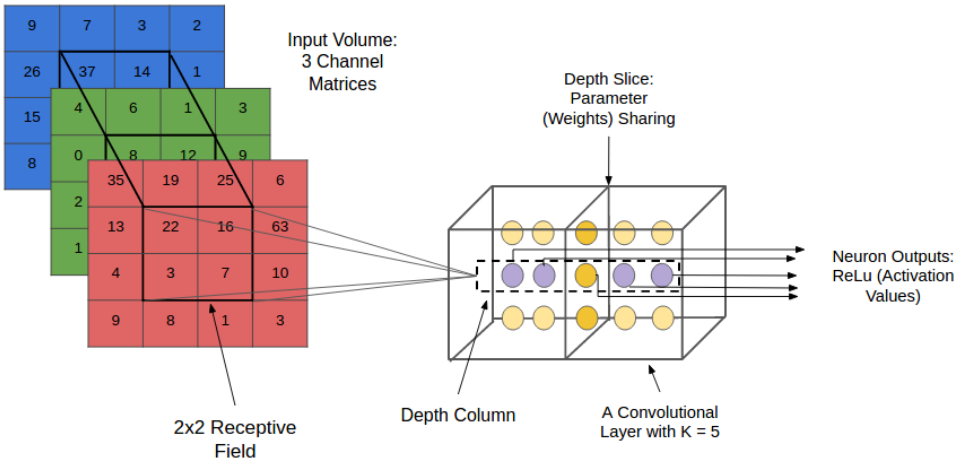

Fig. 5.11: Concept of the depth slice and the receptive field in a convolutional layer. [62]

Pooling layers

After a certain number of convolutional layers, typically a so-called *pooling* layer is added. This layer contains no parameters, since its only purpose is to reduce the spatial dimensions of an input volume. The pooling operation reduces the number of parameters in a model and additionally, it also serves as a regularization method in order to reduce overfitting.

Pooling layers have spatial filters that are slid over the single depth slices of the input volume with a certain stride, such that the output volume has smaller spatial dimensions. Nowadays, a popular type of pooling is the so-called *max* pooling. For example, if we have a 2×2 filter and stride two, the pooling filter is slid over the single depth slices of the input volume and at each step, a maximum operation is calculated over $2 \cdot 2 = 4$ numbers. Hence, four numbers are reduced to one in the final output volume, which results in throwing away 75% of the total number of activations. In the general case for 2D images, let $[W_1 \times H_1 \times D_1]$ be the input volume, F_W and F_H be the filter size for the width and the height and S be the stride.

Then, the dimensions of the output volume can be calculated as follows:

$$W_2 = \frac{W_1 - F_W}{S} + 1 \tag{5.44}$$

$$H_2 = \frac{H_1 - F_H}{S} + 1 \tag{5.45}$$

$$D_2 = D_1 \,. \tag{5.46}$$

A visualization of the pooling technique is shown in figure 5.12.

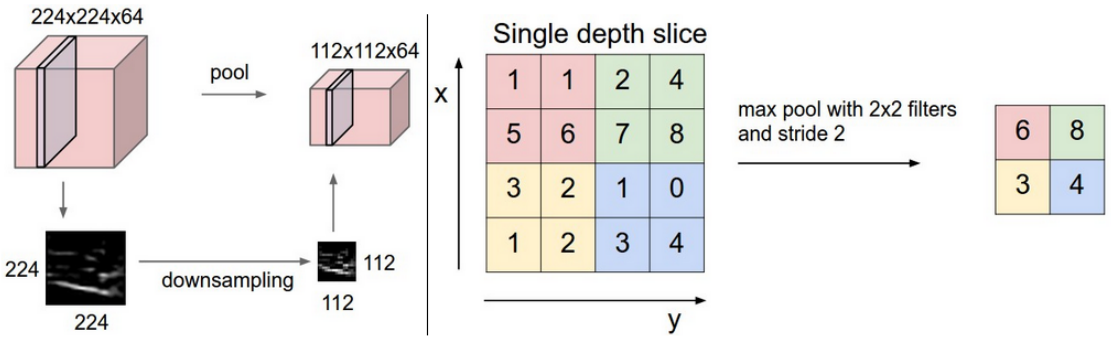

Fig. 5.12: Scheme of the working mechanism of the max pooling layer. [49]

These days, the pooling operation is also often replaced with a larger stride in the convolutional layers. In this way, no pooling layers are needed for the reduction of parameters. Replacing pooling layers with convolutional layers with a larger stride has been motivated by the fact that generative models, like variational autoencoders [64] or generative adversarial networks [65], have shown to strongly benefit by the replacement of the pooling operation [49].

Architecture of a typical convolutional neural network

At this point, all necessary layer types for constructing a convolutional neural network have been introduced: the convolutional layers, the pooling layers and the fully connected layers. Here, each of the convolutional or fully connected layers is followed by an activation layer. Typically, many convolutional layers are stacked in a CNN with some pooling layers in between. After this, fully connected layers are added. The number of fully connected layers can be as small as one, which means that in this case, the single fully connected layer is only used as the last output layer. However, the number of convolutional, pooling and fully connected layers is dependent on the problem and thus up to optimization. A scheme of this layer stacking method for convolutional neural networks is shown in figure 5.13.

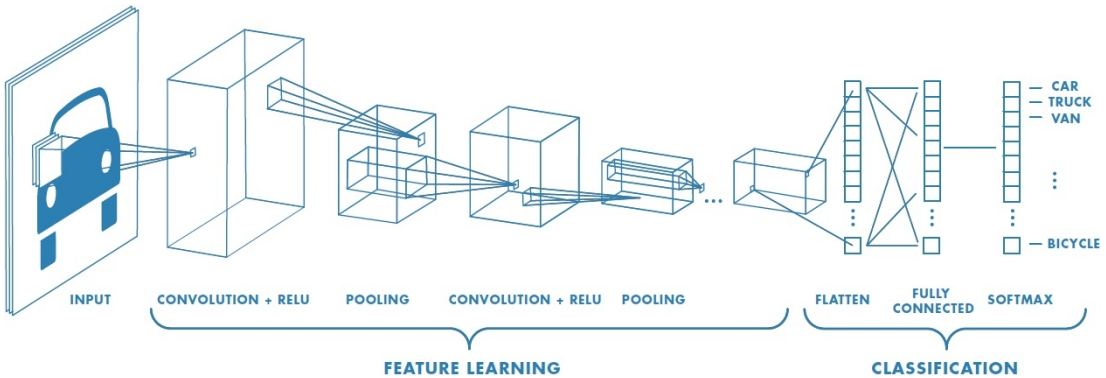

Fig. 5.13: Scheme of a simple convolutional neural network. [66]

One of the first deep networks with this kind of architecture and ReLU activations has been AlexNet [67]. It features comparatively large kernels of size eleven in the first and five in the second convolutional layer as well as a large stride of three. Additionally, dropout layers are used in order to reduce the overfitting on the training dataset. In 2012, AlexNet achieved a top-5 error of 15.3% in the ILSVRC image recognition challenge [61], which was more than ten percentage points better than the second place [67]. Two years later, the so-called VGG architecture improved upon AlexNet in the ILSVRC challenge with a top-5 error of 7.3% [68]. Compared to AlexNet, it replaces the large kernels with smaller ones of size three and it also uses a stride of one. Hence, it could be figured out that multiple convolutional layers with small filters are better suited for the ILSVRC data than a single convolutional layer with large filters. After this, many, even more sophisticated network architectures like residual networks [69][70] and inception networks [71] have been developed. In this work, only VGG-like neural networks are used, such that these more complicated architectures will not be further introduced.

CHAPTER 6

Data preprocessing

Preparation, I have often said, is rightly two-thirds of any venture.

— Amelia Earhart —

THIS chapter introduces how convolutional neural networks can be employed for the event reconstruction in KM3NeT/ORCA. Since CNNs are typically trained in a supervised way, labeled data is necessary. In the case of KM3NeT/ORCA, MC simulations can be used for this purpose.

6.1 ORCA simulations

For this thesis, the goal is to design and apply the CNN reconstruction on simulated data of the full KM3NeT/ORCA detector, consisting of 115 DUs. In the KM3NeT/ORCA collaboration, a full set of MC simulations, generated in 2016 with the production number 180401, is available for a detector geometry with about 23 m horizontal spacing between the DUs and about 9 m vertical spacing between the single DOMs on a single DU.

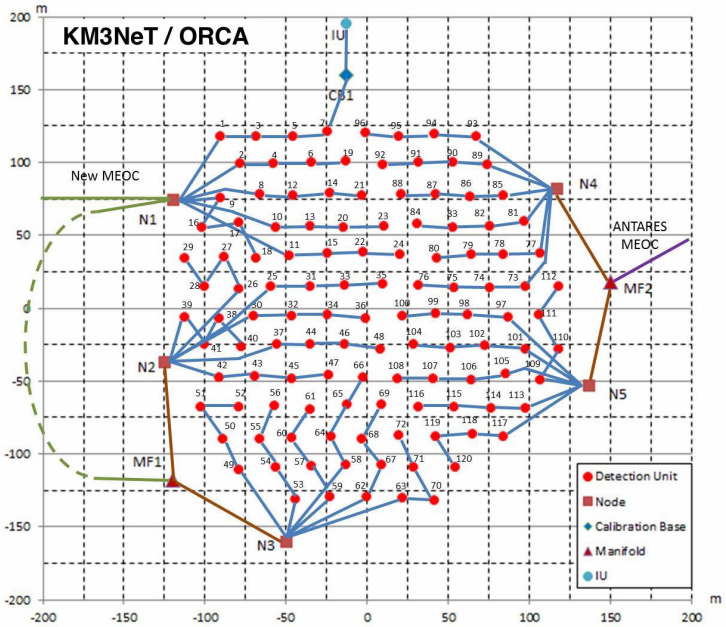

Fig. 6.1: Top-down view of the KM3NeT/ORCA detector layout [12]. There are slight differences of the geometry depicted in the image and the geometry used in the actual MC simulations, which will be shown later. First, the number of detection strings has been changed from 120 to 115 DUs and second, the average horizontal spacing between the DUs has been changed from 20 m to 23 m.

The simulated dataset consists of triggered atmospheric muon events, random noise events and neutrino events. Atmospheric muons have been simulated using the MUPAGE package [72], which generates bundles of atmospheric muons based on parametric formulas. The generated dataset of atmospheric muons represents 14 days of simulated data and it takes a total time of about 17000 CPU-days at the IN2P3 computing center [73] to generate the dataset.

Random noise events have been simulated with 10 kHz of uncorrelated single-noise per PMT, 600 Hz of two-fold coincidences per DOM, 60 Hz of three-fold coincidences per DOM, 7 Hz of four-fold coincidences per DOM, 0.8 Hz of five-fold coincidences per DOM and 0.08 Hz of six-fold coincidences per DOM. These rates are derived from a measurement of exclusive n-fold coincidence rates in KM3NeT/ARCA. Generating this dataset takes about 500 CPU-days.

For neutrinos, the simulated dataset consists of ν_e^{CC}, ν_μ^{CC}, ν_τ^{CC} and ν_e^{NC} events. In this dataset, the direction of the neutrinos is uniformly distributed over 4π in solid angle. Additionally, neutrinos and antineutrinos are generated in a one to one ratio, though the increased cross section of neutrinos leads to a ratio of about two to one of actually interacting neutrinos. The neutrinos have been simulated in two sets: a low-energy dataset with neutrino energies from 1 GeV to 5 GeV and a high-energy dataset with neutrino energies from 3 GeV to 100 GeV. An exception to this are tau neutrinos, which have only been simulated for the high-energy dataset. For the high energy dataset, the energy spectrum during the generation of the events is E^{-3} and for the low energy dataset it is E^{-1}. Thus, the simulated neutrino dataset is not based on an oscillated, atmospheric neutrino flux. Generating both sets of data, i.e. the high-energy and the low-energy MC production, takes about 2200 CPU-days.

For training a deep neural network, having as much training data as possible is beneficial for the performance of the algorithm. For example, in the field of computer vision it has been demonstrated that deep learning algorithms used for representation learning can still benefit from larger training datasets up to 300 million samples [74]. Likewise, during the course of this thesis, it has been assessed that the presented CNNs benefit significantly from a larger training dataset than has been originally available. At the same time, for classification algorithms like a track-shower classifier, the training dataset should be balanced with respect to the number of samples for each class. However, the original MC simulations have not been generated with this in mind, e.g. the fraction of track-like ν_μ^{CC} events to shower-like $\nu_e^{CC} + \nu_e^{NC}$ events is about 2:1. This limits the maximum size of the training dataset significantly.

As a consequence, additional simulations have been carried out in order to get a larger training dataset and to make a better use of the originally available simulation dataset. In particular, two times more random noise events and about one time more neutrino events have been simulated for this work in addition to the already available simulation data. A summary with detailed information about the finally given simulated data is shown in table 6.1.

Event type & Energy range	E_{gen} spectrum	$N_{trigg}[10^6]$
Atmospheric muon	-	65.2
Random noise	-	23.3
ν_e^{NC}: 1-5 GeV / 3-100 GeV	E^{-1} / E^{-3}	1.1 / 3.7
ν_e^{CC}: 1-5 GeV / 3-100 GeV	E^{-1} / E^{-3}	1.5 / 4.4
ν_μ^{CC}: 1-5 GeV / 3-100 GeV	E^{-1} / E^{-3}	1.7 / 8.3
ν_τ^{CC}: 3-100 GeV	E^{-3}	2.1

Table 6.1: Overview on the finally available MC simulations for a 115 DU KM3NeT/ORCA detector composed for the 180401 MC production. The first column shows the simulated event type, i.e. atmospheric muons, random noise or neutrino interactions. For neutrinos, the energy range used in the simulations is shown additionally. The neutrino simulations comprise neutrino and antineutrino interactions of the indicated type. The second column specifies the power-law used to simulate the energy spectrum of the neutrinos. A reweighting scheme is used in this work where appropriate to simulate an atmospheric neutrino flux model. N_{trigg} is the number of events that remain after triggering.

For each hit in a simulated event, the PMT identifier, and thus the relative coordinate of the hit PMT in a DOM is recorded. Additionally, the point in time, when the time-over-threshold measurement of each individual PMT starts and the measured ToT are stored. However, the ToT value is not used as input for the CNNs due to multiple reasons. First, the actual value of the ToT is not directly related to the number of measured photons. Second, for GeV neutrinos, the frequency of single photon induced hits is dominant over hits induced by multiple photons. At last, it needs to be studied first, if the ToT is accurately simulated in order to not introduce a systematic error. Hence, only the hit information and not the ToT is given to the CNN. In order to feed this four-dimensional event data to a CNN, the hits can be binned into rectangular pixels, such that each image encodes three spatial dimensions, XYZ, and the time dimension T.

6.2 Spatial binning

Fulfilling the goal to finely resolve the spatial coordinates of the individual PMTs inside the DOMs, the number of pixels would have to be very large, and most bins would be empty due to the sparsely instrumented detector volume. Therefore, the pixelization is defined in such a way that at most one DOM fits into one bin. For the Z dimension, the DOMs are located in the center of the bins, which is possible due to the regular spacing between the DOMs on a single DU. On the other hand, the single DUs are arranged irregularly in the XY dimensions, such that the DUs in the bins of the XY plane are not centered within the bins. Thus, one of the tasks that the CNN has to tackle internally is to learn the relative positions of the strings within the bins in the XY plane. A scheme of the binning for the XY dimensions and the positions of the KM3NeT/ORCA strings within the related bins is shown in figure 6.2 (left). In addition, an XY projection of an exemplary neutrino event is shown in figure 6.2 (right).

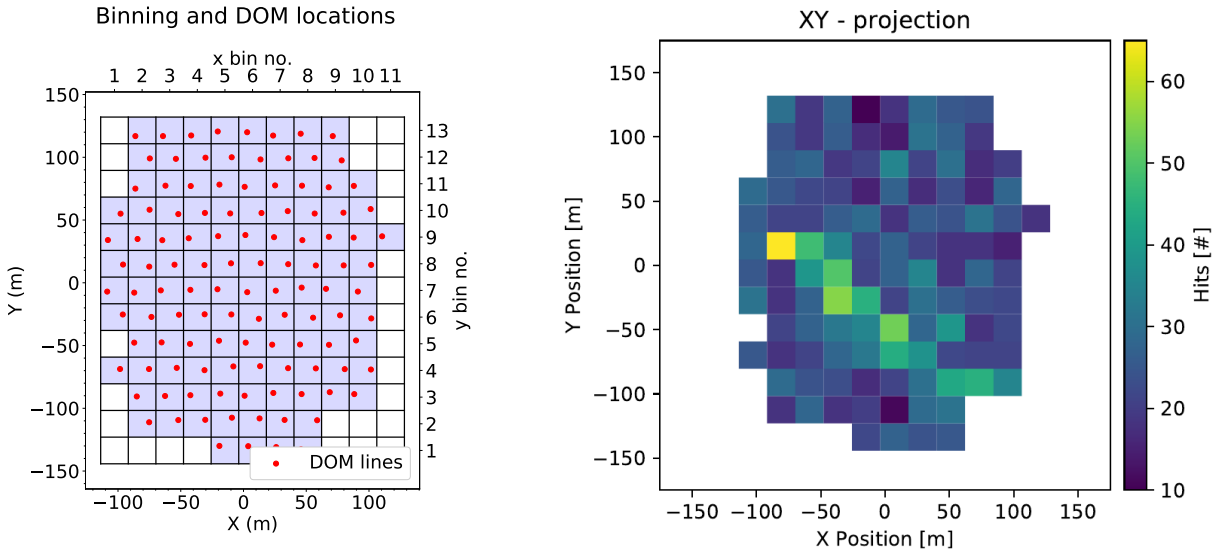

Fig. 6.2: Left: scheme for the binning of the DOMs in the XY direction. The grid that is drawn represents the bin edges of the bins. Right: the XY projection of an exemplary ν_μ^{CC} event with an energy of 78.8 GeV.

In the case of the 115 DU KM3NeT/ORCA detector, the spatial binning results in a $11 \times 13 \times 18$ (XYZ) pixel grid. However, the information which PMT in a DOM has been hit is lost in this way. This is fixed by adding a PMT identifier dimension to the pixel grid, resulting in a XYZP grid. Since one DOM holds 31 PMTs, the final spatial shape of such an image is $11 \times 13 \times 18 \times 31$ (XYZP). Images with these dimensions can also be found in classical computer vision tasks, e.g. as colored videos. The only difference is that in a usual videos, the Z-coordinate is replaced by the time and the PMT identifier is replaced by the red-green-blue color information.

6.3 Temporal Binning

After this, the time information is added to the XYZP image as an additional dimension, yielding five-dimensional XYZTP images. The time resolution of the PMTs in the DOMs of the ORCA detector is of the order of nanoseconds. As explained in section 3.1.1, the time length of an event is about 3 µs, implying 3000 bins for the time dimension to reach a nanosecond resolution. However, with each additional bin, the size of the event image gets larger, leading to additional computations in the first convolutional layer of the CNN. Hence, the number of bins in the final image should be as low as possible, while still containing the relevant timing information.

Most hits that lie outside the time range of the triggered hits are background hits and not signal hits, since the event time margin, cf. section 3.1.1, is based on the detector size. Therefore, background hits are discriminated by selecting the time range in which most signal hits are found. Investigating the distribution of the time of the signal hits relative to the mean of the triggered hit times in individual events, as depicted in figure 6.3 for ν_μ^{CC}events, shows that the distribution is asymmetric and that the relevant time range can be reduced significantly for the binning.

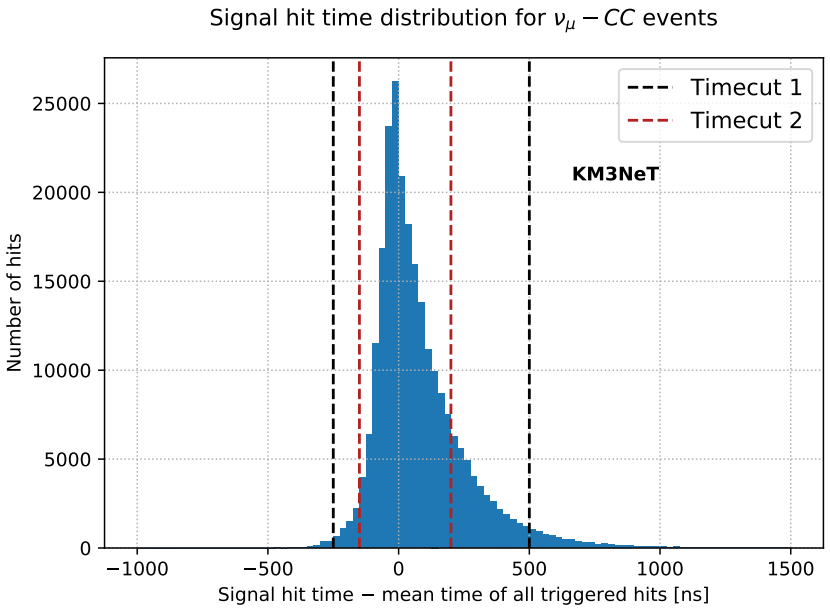

Fig. 6.3: Time distribution of ν_μ^{CC} signal hits centered with the mean time of the triggered hits for single events. For this distribution, about 3000 ν_μ^{CC} events have been used. The dashed lines in black and red visualize possible timecuts.

Since the time range covered by the triggered hits is different for each event, the time range selection is defined relative to the mean time of the triggered hits for each event. In figure 6.3 it can be seen that only a small fraction of the signal hits is cut, if the timecut indicated by the dashed black lines would be used. Hence, a compromise needs to be found between the width of the timecut window, the final time resolution and the number of time bins. In principle, the timecut can also be defined relative to other values, like the median time of the triggered hits or the time of the first triggered hit. Hence, the timecut window is a parameter that can be further optimized with respect to the final performance of a trained neural network. In this work, only a minor amount of such parameter optimization studies has been carried out for the presented CNNs, which implies that their performance for concrete use cases can likely be further improved.

Other than throwing away all hits that are located either before or after the timecut window, one can also include them in a single large time bin, i.e. one bin for hits before the timecut window and one bin for hits after the timecut window. In this way, no information is thrown

away, though the time resolution for these two bins is very small compared to the one of the other time bins. This binning scheme has not been used for the CNNs presented in this work. However, recent investigations with a muon direction reconstruction for a 1 DU ORCA detector show that the performance can be improved compared to throwing away the hits that lie outside of the timecut window. For a full 115 DU ORCA detector, it remains to be seen in the future if comparable gains in reconstruction performance can be made using this technique. Neglecting the PMT identifier dimension, figure 6.4 shows 2D projections of the 4D XYZT image for a single ν_μ^{CC} event with 60 time bins and the timecut 1 that is shown in figure 6.3.

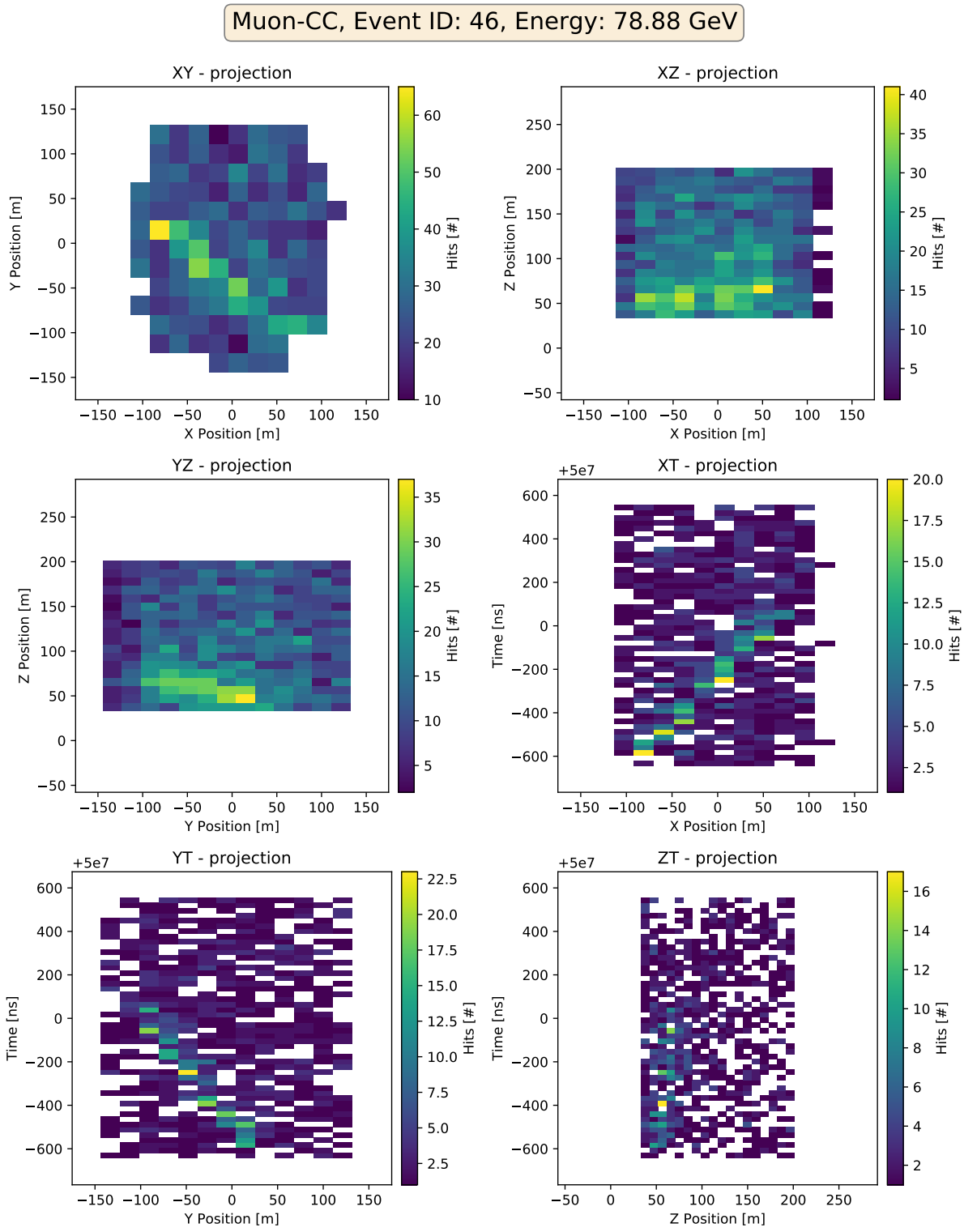

Fig. 6.4: 2D projections of a 4D $[11 \times 13 \times 18 \times 60]$ XYZT image of a ν_μ^{CC} event with timecut 1.

6.4 OrcaNet and multi-image CNNs

After both the spatial as well as the temporal binning, the resulting KM3NeT/ORCA event images are five-dimensional: XYZTP. For the time dimension, the assumption of a correlation between adjacent time bins holds, similar to the spatial dimensions, such that the dimension can be convolved over. For the PMT identifier dimension, it is not clear whether a convolution makes sense, since the single PMTs point into different directions, and the angle between two adjacent PMTs can be significantly different to the angle between two other adjacent PMTs. However, this is similar to the circumstance that the KM3NeT/ORCA DUs are not centered inside the bins of the XY plane. And since the positions of the PMTs in the DOMs are the same for all events, the different angles between adjacent PMTs could be learned by a network. To this end, it would need to be tested if a convolution over the PMT dimension is sensible or not, i.e. whether performance gains can be observed or not. For this work, the PMT channel dimension is used like the red-green-blue channel information in 2D images, where the weights for a neuron are not shared between these channel slices. Thus, a CNN with 4D convolutional layers and 5D XYZTP input would be suited in order to process the input.

The CNNs in this work are implemented using the popular deep learning framework TensorFlow [75] in conjunction with the Keras [76] high-level neural networks library. However, TensorFlow does not support convolutional layers with more than three convolutional dimensions, since five-dimensional inputs are not a usual use case in computer vision. This also applies to several other major deep learning frameworks like PyTorch [77]. Additionally, it would take a huge effort to implement the 4D convolution in TensorFlow, since the implementation of the four-dimensional convolution needs to be executed on a low-level stage at the GPU level within the CUDA [78] framework for Nvidia GPUs. Currently, there is a deep learning framework that supports arbitrary, n-dimensional convolutions called CNTK [79]. However, as this deep learning project for ORCA had been started, CNTK was not supported by Keras. Hence, TensorFlow with the Keras interface has been used for the training of all neural networks in this work.

Consequently, the five dimensions of the XYZTP images need to be reduced to four dimensions, such that three-dimensional convolutional layers can be used. For this, one image dimension is summed up, i.e. the information of the individual PMTs in a DOM is discarded, such that the resulting image is only four-dimensional (XYZT). Since this four-dimensional XYZT input is fed to a three-dimensional convolutional layer, the fourth dimension, in this case the time dimension, will not be convolved over. It is not clear whether this is a drawback for the performance of the network or not. In the end, this has to be tested, when a four-dimensional convolutional layer becomes available in TensorFlow. In addition, it is also not evident, which dimension should be chosen as the last dimension, i.e. the dimension that will not be convolved over. However, performance studies with the track-shower classifier introduced in section 7.2 show that the XYZT projection yields the best performance for the track-shower classifier CNN compared to other permutations like YZTX. Thus, the four-dimensional XYZT projection of the five-dimensional XYZTP images has been used as a baseline for all presented CNNs, though the XYZT permutation may not necessarily be the best one for the other CNNs in this work, i.e. the background classifier and the neutrino event regressor.

In order to not completely lose the PMT identifier information, a second image of the same event with XYZP dimensions is fed to the network that reincludes the information which PMT in a DOM has been hit. Note that this is not equivalent to the information contained in a XYZTP image, since the hit time is discarded in the XYZP image and hence, a time dependent PMT identifier information does not exist. Since the XYZT and the XYZP images only differ in the fourth dimension, i.e. the channel dimension of convolutional layers, the images can be stacked in this dimension. For example, a $[11 \times 13 \times 18 \times \mathrm{T}]$ XYZT image and a $[11 \times 13 \times 18 \times 31]$ XYZP image can be combined to form a $[11 \times 13 \times 18 \times \mathrm{T} + 31]$ image. These images are referred to

as XYZ-T/P images, where the hyphen indicates that the stacked T/P dimension is a channel dimension and thus not convolved over, contrarily to the XYZ dimensions. Significant gains in performance for all CNN applications in this work could be observed by using this stacking method, as compared to just supplying a single XYZT image. Additionally, other projections like a YZTX projection can be supplied as additional input images using so-called multi-image CNNs. This approach has been attempted for the CNN track-shower, cf. section 7.2, and significant gains in performance could be observed. Even though some information has been lost due to the lack of a four-dimensional convolutional layer, it will be shown in this work that networks with such input limitations can still match or even outperform the standard KM3NeT/ORCA reconstruction pipeline.

The training of all presented networks has been executed at the TinyGPU cluster of the RRZE computing center [80]. It consists of 26 computing nodes with 4 GPUs each. The GPUs are either Nvidia GTX1080, Nvidia GTX1080Ti, Nvidia RTX2080 Ti, or Nvidia Tesla V100. For an efficient GPU training, two Nvidia software packages are needed, CUDA [78] and cuDNN [81]. All CNNs in this work have been trained with CUDA 10.0 and cuDNN 7.4.2. In order to train the networks, an open-source software framework called *OrcaNet* [82] has been developed within the scope of this thesis. It is intended to be used as a high-level interface on top of Keras [76]. OrcaNet is specifically suited to the challenges that occur in (astro-) particle physics like large datasets. As of now, OrcaNet is used as a standard tool within KM3NeT and the principal idea is to also employ it for other experiments.

6.5 Main network architecture and training process

After the four-dimensional images of the simulated events have been created, these images are fed to the CNN. All networks that have been designed within the scope of this work share a common CNN architecture. The general structure is based on a simple, VGG-like CNN, such that it can serve as a baseline for more sophisticated CNN architectures in the future. The CNNs is comprised of two main components: the convolutional part with the convolutional layers and a small fully connected network in the end. The convolutional part consists of stacked *convolutional blocks*, and each of them contains a convolutional layer, a batch normalization layer, an activation layer and, optionally, a dropout or a maximum pooling layer. The structure of these convolutional blocks is summarized in table 6.2.

Layer type	Properties
Convolution	Kernel size $(3 \times 3 \times 3)$, He uniform initialization, zero padding
Batch normalization	Parameters as in [58]
Activation	ReLu
Max pooling, stride ρ	optional, no zero padding
Dropout, rate δ	optional

Table 6.2: Structure of a convolutional block.

For the three-dimensional convolutional layer, the initial weights are drawn from a uniform distribution with a variance based on the He initialization, cf. section 5.2.6, and the biases are initialized with zeros. Additionally, the kernel size is three $(3 \times 3 \times 3)$, the stride, i.e. the step size in shifting the convolutional kernel, is one $(1 \times 1 \times 1)$ and zero padding is used. For the batch normalization layers, the standard parameters from [58] are used. After this, a ReLU activation layer is added. The specific order of adding the batch normalization layer before the activation layer is based on the recommendation in [58], though the actual order is still a subject of debate in the computer science community and thus the actual order is up to optimization. These three

layers are found in all convolutional blocks that are used in this work. Additionally, optional maximum pooling layers, to reduce the dimensionality of the network, and dropout layers, to avoid overfitting, are added. In the case of maximum pooling layers, no zero padding is applied. From now on, the symbol δ will be used for the dropout rate and the symbol ρ will be used for the stride in the maximum pooling operation. These convolutional blocks are stacked and after repeating this pattern for multiple times, the four-dimensional output of the layers is *flattened* to one dimension in order to be able to add additional fully connected layers, which include the output layer.

Apart from this CNN structure, residual networks as in [70], cf. section 5.2.9, have been studied for the track-shower classifier CNN, since they show a better performance for standard image recognition tasks like in the ILSVRC image recognition challenge [61]. However, the employment of residual networks led to a significant overfitting and a worse performance compared to the presented CNN architecture. Adding more regularization, e.g. by increasing the dropout rate, resulted in less overfitting but the overall performance was still worse compared to the VGG-like network. Thus, residual neural networks have not been further pursued, though more extensive investigations could lead to an improved performance. Other than residual networks, many other promising CNN architectures [71, 83, 84] exist, which could lead to an improved performance in the future.

All presented CNNs are trained with the Adam gradient descent optimizer, cf. section 5.2.5, using standard parameters, except for the value of the epsilon parameter, which is increased from 1×10^{-8} to 0.1. A larger epsilon value results in smaller weight updates after each training step. For the presented CNNs, it has been observed that occasionally, depending on the random initialization of the parameters, the network never starts to learn. This can be fixed by changing the value of the epsilon parameter to 0.1 as suggested in [85], while significant drawbacks, such as a slower training convergence due to smaller weight updates, are not observed. The weights of the neural network are updated, after one batch of images is passed through the network. The batch size in the training for all presented CNNs is set to 64 and the learning rate is annealed exponentially over the epochs. In [55], a learning rate of 1×10^{-3} is suggested, though this can only be taken as a rough estimate, since the ideal learning rate is a hyperparameter that is dependent on the specific case study. The initial learning rate has been set to a value as high as possible with the condition that the network still starts to learn, i.e. that the loss improves upon the initial loss based on random guessing of the network. For example, in the case of a binary softmax classifier with a balanced training dataset with respect to the number of samples per class, it can be expected that the network initially assigns a probability of 0.5 to each class. For a softmax classifier, cf. section 5.2.2, the loss based on this random guessing is $\ln(0.5) = 0.693$, which can also be observed in the training loss during the early stages of the training. If the learning rate is set to high, the network never starts to learn, i.e. the loss is stuck at 0.693 for the example above, or the loss even diverges. The training is stopped, when the loss does not improve significantly anymore with further training and manually increasing or decreasing the learning rate has no effect as well.

The main goal of this thesis is to establish a full, deep learning-based event reconstruction for KM3NeT/ORCA in order to demonstrate the feasibility and the potential performance gains. Hence, no major hyperparameter optimizations have been carried out for all CNNs in this work, in order to be able to cover the full ORCA reconstruction pipeline. Most optimizations like the number of convolutional layers, the number of pooling layers or the batch size have been done for the track-shower classifier and then adopted for the other CNNs. These parameters of the track-shower classifier may not necessarily be the best for the other CNNs as well. An example is the batch size, which has been fixed to 64 for all CNNs, based on the observation that a smaller (32) or a larger (128) batch size yields a worse performance for the track-shower classifier.

CHAPTER 7

CNN-based event reconstruction for KM3NeT/ORCA

An expert is a person who has found out by his own painful experience all the mistakes that one can make in a very narrow field.

— *Niels Bohr* —

In this chapter, the main results of this thesis, consisting in the CNN-based event reconstruction for the 115 DU KM3NeT/ORCA simulations, will be presented. The chapter is organized as follows. Section 7.1 introduces the CNN background classifier, which distinguishes neutrinos from background events, i.e. atmospheric muons and randomly correlated noise. After this, section 7.2 introduces the track-shower classifier, distinguishing track-like from shower-like neutrino events. At last, section 7.3 presents the CNN event regressor, which reconstructs neutrino properties like the energy and the direction.

7.1 Background classifier

An essential part of the KM3NeT/ORCA reconstruction pipeline is the background classifier, which discriminates atmospheric muons and randomly correlated noise events from neutrino-induced events. For this purpose, the currently employed classification algorithm is based on Random Forests (RFs), cf. section 5.1.1. For the background classification, there are two separate RFs. The one RF is an atmospheric muon classifier, yielding a muon score, while the other one is a random noise classifier, yielding a random noise score. By employing both the atmospheric muon RF classifier as well as the random noise RF classifier, non-neutrino events can be efficiently distinguished from neutrino events based on the respective scores. The inputs of the RFs are high-level observables (features), mainly determined by maximum likelihood-based track and shower reconstruction algorithms. In this section, an alternative classifier based on CNNs is presented and its performance is compared to the standard KM3NeT/ORCA background classifiers.

7.1.1 Image generation

As outlined in section 6.4, stacked XYZP and XYZT images are fed as a single XYZ-T/P input to the network. For both event images, a timecut has to be chosen, as introduced in section 6.3. It has been observed that the time distribution of the signal hits, as shown in figure 6.3 for charged current muon neutrino interactions and in figure 7.1 for atmospheric muons, has a larger variance for atmospheric muon events than for neutrino events. The reason is that, on average, atmospheric muons traverse larger parts of the detector compared to the secondary particles of the neutrino interactions. Hence, the timecut window for all event classes has been set conservatively based on atmospheric muon events. The signal hit time distribution of atmospheric muons, relative to the mean time of all triggered hits, is shown in figure 7.1.

Based on this distribution, the timecut window has been set to an interval of $[-450\,\text{ns}, +500\,\text{ns}]$, keeping more early signal hits compared to late signal hits. The reason for this is that events which produce late hits are typically higher energetic and thus they leave a longer trace in the detector. Hence, they can be reconstructed better than low energetic atmospheric muons that only produce a few hits at the edge of the detector. Consequently, cutting a few late hits has a small effect compared to discarding hits from a low energy atmospheric muon event, which already produces a low number of hits. Ideally, a timecut window that contains nearly all signal hits,

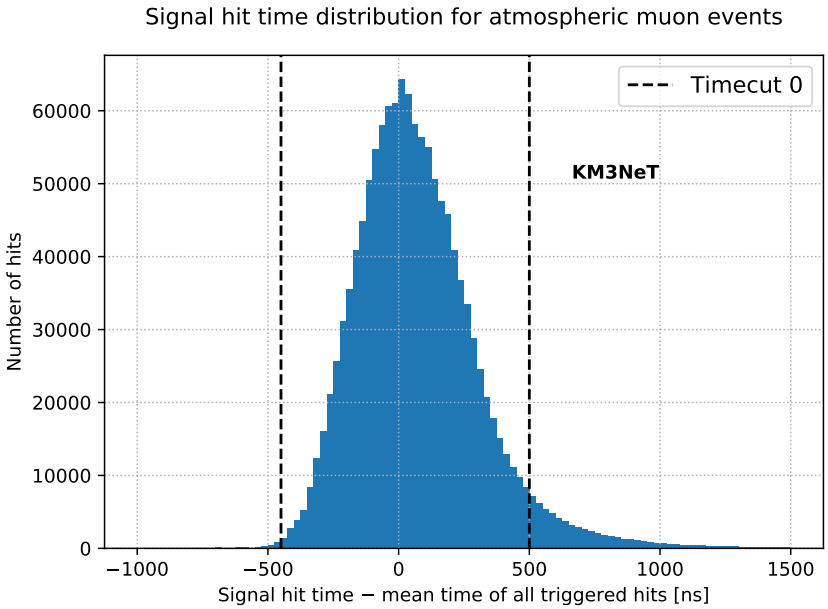

Fig. 7.1: Time distribution of atmospheric muon signal hits relative to the mean time of the triggered hits calculated for each individual event. For this distribution, about 3000 atmospheric muon events have been used. The dashed, black line visualizes the timecut that is used for the generation of the background classifier images. This so-called timecut 0 is set to an interval of $[-450\,\text{ns}, +500\,\text{ns}]$.

i.e. $[-500\,\text{ns}, +1500\,\text{ns}]$, should be chosen. However, if the number of time bins is kept constant, this approach worsens the time resolution of each time bin significantly, as already mentioned in section 6.3.

The number of bins for the time dimension is set to 100, such that the XYZT images have a dimension of $11 \times 13 \times 18 \times 100$. Then, the resulting time period of each time bin is 9.5 ns, which translates to about 2 m of photon propagation distance. This is small compared to the horizontal spacing of the KM3NeT/ORCA DUs (about 23 m) and the vertical spacing between two DOMs on the same DU (about 9 m). However, it needs to be investigated in the future if and by which amount performance gains could still be made with more time bins and thus a higher time resolution. Including the PMT identifier information, the final XYZ-T/P event images have the dimension $11 \times 13 \times 18 \times 131$.

7.1.2 Network architecture

The CNN architecture for the background classifier is based on the convolutional blocks introduced in section 6.5, with two additional fully connected layers, also called *dense* layers, at the end. The output layer of the CNN is composed of two neurons, such that the network only distinguishes between neutrino and non-neutrino events.

Initially, a CNN with three output neurons had been tested, such that neutrinos, atmospheric muons and random noise events could be classified separately. However, it has been observed that the three-class CNN performed slightly worse than the two-class CNN that distinguishes only neutrino events from all others. This is due to the fact that the network does not prioritize neutrino vs. non-neutrino classification in this case. A mistakenly classified atmospheric muon, e.g. classified as a random noise event, has the same effect on the total loss as an atmospheric muon classified as a neutrino. However, since the distinction between an atmospheric muon and a random noise event proves to be easy for the network, the performance advantage of the two-class classifier over the three-class classifier is small. Nevertheless, if one only wants to distinguish neutrinos from non-neutrinos, the two-class classifier with the slight gain in performance is the

best choice and thus this approach is further pursued. An overview of the final network structure is shown in table 7.1.

Building block / layer	Output dimension		
XYZ-T Input	$11 \times 13 \times 18$	$\times$	100
XYZ-P Input	$11 \times 13 \times 18$	$\times$	31
Stacked XYZ-T + XYZ-P Input	$11 \times 13 \times 18$	$\times$	131
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters, $\rho = (2, 2, 2)$)	$5 \times 6 \times 9$	$\times$	64
Convolutional block (128 filters)	$5 \times 6 \times 9$	$\times$	128
Convolutional block (128 filters)	$5 \times 6 \times 9$	$\times$	128
Convolutional block (128 filters)	$5 \times 6 \times 9$	$\times$	128
Convolutional block (128 filters, $\rho = (2, 2, 2)$)	$2 \times 3 \times 4$	$\times$	128
Flatten	3072		
Dense 128 + ReLu	128		
Dense 32 + ReLu	32		
Dense 2 + Softmax	2		

Table 7.1: Network structure of the background classifier's three-dimensional CNN model with XYZ-T/P input. No dropout is used due to the large training dataset of 42.6×10^6 training events. The symbol ρ specifies the stride for the maximum pooling operation used in the respective convolutional block.

No regularization techniques, such as dropout, are used. The reason is that the training dataset is large, cf. section 7.1.3, so that only a minor amount of overfitting occurs and additional dropout layers, even with a small rate like 5%, show no improvement with respect to the performance of the CNN.

7.1.3 Preparation of training, validation and test data

For the training of the background classifier, the simulated data from table 6.1 is used. Here, the dataset is split into a training, a validation and a test dataset. Whenever possible, the same neutrino events have been used in the test datasets of all three presented CNNs, i.e. the background classifier, the track-shower classifier and the event regressor, in order to conserve an as large as possible neutrino data sample for the tau neutrino appearance analysis introduced in chapter 8.

With this in mind, tau neutrinos are not included in the training dataset of the background classifier. The reason is that this circumstance also applies to the training datasets of the track-shower classifier and the event regressor. For the track-shower classifier, tau neutrinos are not included in the training dataset, since ν_τ^{CC} interactions don't have a fixed shower- or track-like signature like for ν_e^{CC}, ν_e^{NC} and ν_μ^{CC} events, cf. section 7.2.3. For the event regressor, the inclusion of tau neutrinos to the training dataset is not straightforward, as will be shown

in section 7.3.3. Thus, tau neutrinos have also not been included in the training dataset of the background classifier, in order to preserve a large tau neutrino sample for the later tau neutrino appearance analysis.

Before the training, the data needs to be balanced with respect to frequency of the classes. For example, the data could be split into 50% neutrino and 50% non-neutrino events (25% atmospheric muons + 25% random noise). This is done in order to avoid a local minimum during the training process, where the network always predicts one class, if one class occurs a lot more frequently than the other one during the training. Consequently, for a classifier with n classes, the data is usually rebalanced into n equal fractions. Considering that the simulated number of neutrinos is lowest, about 22.8×10^6 events, one would have to remove a significant fraction of the 23.3×10^6 generated random noise events, and of the 65.1×10^6 atmospheric muon events for a class-balanced data splitting. On the other hand, based on the RF background classifier performance of the standard reconstruction pipeline, it can be expected that the final accuracy, cf. equation 7.1, of the classifier with this rebalanced dataset is close to 99%. Thus, the following data splitting is used, in order to make a better use of the available simulated data: 1/3 neutrino events, 1/3 random noise events and 1/3 atmospheric muon events. Hence, the final class balance is 1/3 neutrino events and 2/3 non-neutrino events. Using this data splitting and considering the number of simulated events summarized in table 6.1, the size of the training dataset is larger compared to a 50/50 split.

Afterwards, this rebalanced dataset is split into 75% training, 2.5% validation, and 22.5% test events. In addition, the events that have been removed to balance the dataset, consisting in mostly atmospheric muons, are added to the test dataset. In total, the training dataset contains about 42.6×10^6 events. As mentioned in section 6.5, the learning rate is annealed exponentially. Likewise, the initial learning rate has been set to a value of 5×10^{-3}. During the training, the network is monitored based on two metrics, the cross entropy loss and the so-called *accuracy*, where the accuracy is defined as

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} . \tag{7.1}$$

An overview of the training and the validation cross entropy loss as well as the corresponding values for the accuracy of the CNN softmax classifier over the epochs is shown in figure 7.2.

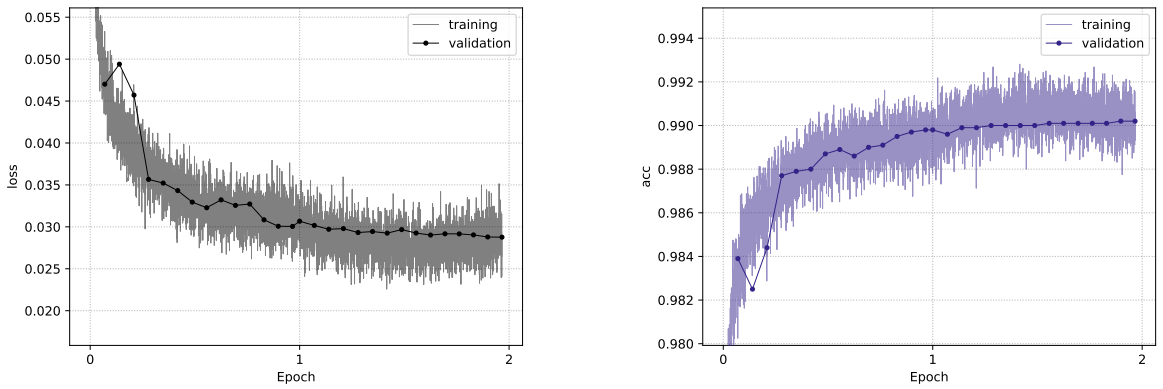

Fig. 7.2: Training and validation cross entropy loss (left) and accuracy (right) of the CNN background classifier during the training. Each data point of the training loss and the training accuracy metric is averaged over 250 batches, i.e. 250×64 event images.

It can be seen that the network starts to learn quickly, since the loss already drops to 0.05 till the first validation phase. Likewise, the accuracy of the CNN is close to 99%. However, this also

needs to be the case, since atmospheric muons and random noise events are far more frequent than neutrino events and thus they need to be distinguished from neutrinos in an efficient way. Using an Nvidia Tesla V100 GPU, it takes about two weeks to train this background classifier CNN.

7.1.4 Performance and comparison to RF

Based on the test dataset, the performance of the CNN can be compared to the two RF background classifiers. Since many triggered events have a low number of signal hits and are difficult to reconstruct, they are typically removed for physics analyses. Therefore, this *pre-selection* is also used, in order to derive a fair comparison between both techniques.

The pre-selection criteria are based on several concepts. Firstly, the fraction of atmospheric muon and random noise events is reduced to a few percent. For atmospheric muons this is achieved by selecting only events for which the reconstructed particle direction origin is below the horizon, i.e. events which are *up-going* in the detector. Furthermore, the events must have been reconstructed with high quality by either the standard KM3NeT/ORCA maximum likelihood reconstruction algorithm for track-like or for shower-like events. Finally, events reconstructed as originating from outside of the instrumented volume of the detector are removed. Aside from this, the maximum likelihood-based reconstruction methods sometimes fail, e.g for events with a very low number of hits, so that these events are also removed for the comparisons in this work. In total, the pre-selected test dataset consists of about 3.3×10^6 neutrino, about 6×10^4 and about 4×10^4 random noise events.

In order to get a first impression of the CNN-based background classifier, the distribution of the predicted neutrino class probability is investigated for all three event types, i.e. neutrinos, atmospheric muons and random noise, cf. figure 7.3. It can be seen that the rate of misclassified

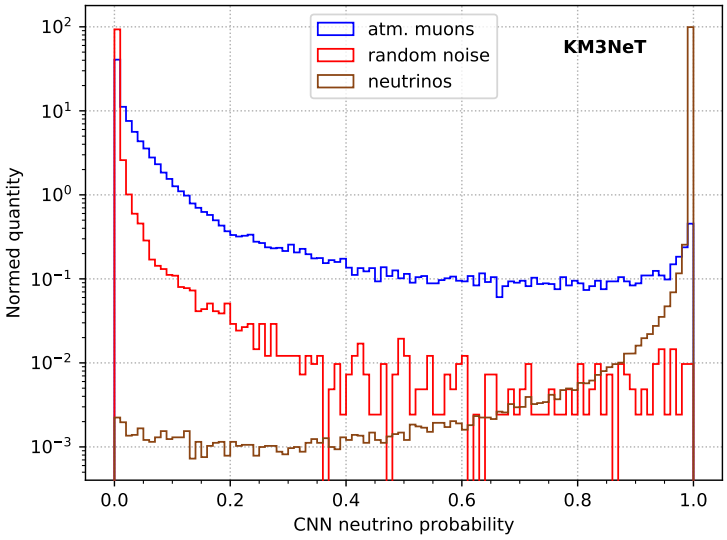

Fig. 7.3: Probability of an event to be neutrino-induced, as determined by the CNN, for pre-selected atmospheric muon (blue), random noise (red), and neutrino (brown) events. All three distributions have been normalized, i.e. the area under each histogram sums to one.

random noise events as non-neutrino events is significantly lower than for atmospheric muons. Using the predicted probability for an event to be classified as neutrino-induced, a threshold value p can be set to remove background events. In order to quantify the performance of the CNN background classifier, the metrics shown in equation 7.2 and equation 7.3 are used to investigate the number of remaining atmospheric muon and random noise events for a given threshold value

p. The atmospheric muon or random noise contamination and the neutrino efficiency are defined as:

$$C_{\mu/\mathrm{RN}}(p) = \frac{N_{\mu/\mathrm{RN}}(p)}{N_{\mathrm{total}}(p)}\,, \tag{7.2}$$

$$\nu_{\mathrm{eff}}(p) = \frac{N_{\nu}(p)}{N_{\nu,\mathrm{total}}}\,. \tag{7.3}$$

Here, $N_{\mu/\mathrm{RN}}$ is the number of atmospheric muon or random noise events, whose probability to be a neutrino-induced event is higher than p, while N_{total} is the total number of events after the same cut on p. Regarding the neutrino efficiency, $N_{\nu}(p)$ is the total number of neutrinos in the dataset, whose neutrino probability is greater than the threshold value and $N_{\nu,\mathrm{total}}$ is the number of neutrinos in the dataset without applying any threshold. Additionally, the events are weighted according to their expected flux, such that the contamination level in the detector can be estimated. For neutrinos, the weighting is based on the Honda atmospheric neutrino flux [34] model and the oscillation parameters given in [86]. For atmospheric muons, the weighting is included in the MUPAGE [72] parametrization. Random noise events are weighted according to the rates introduced in section 6.1. By scanning through the threshold value p the neutrino efficiency dependent on the atmospheric muon or the random noise contamination can be calculated. While the output of the CNN classifier, the probability, is a continuous value, the output of the RFs is discrete. For the RFs, an atmospheric muon or random noise score $s_{\mu/\mathrm{RN}}$ is assigned to each event, where the score is defined as follows:

$$s_{\mu/\mathrm{RN}} = \frac{N_{\mathrm{trees},\mu/\mathrm{RN}}}{N_{\mathrm{trees,total}}}\,. \tag{7.4}$$

Here, $N_{\mathrm{trees},\mu/\mathrm{RN}}$ is the number of trees in the RF that have classified an event as an atmospheric muon or random noise event and $N_{\mathrm{trees,total}}$ is the total number of trees in the RF. Both the atmospheric muon as well as the random noise RF classifier, consist of 101 trees. Thus, the muon or noise score is a discrete number with 102 possible values. A scan through the threshold value p for the atmospheric muon contamination and both the CNN as well as the atmospheric muon RF classifier is shown in figure 7.4. It can be seen that the CNN background classifier has a

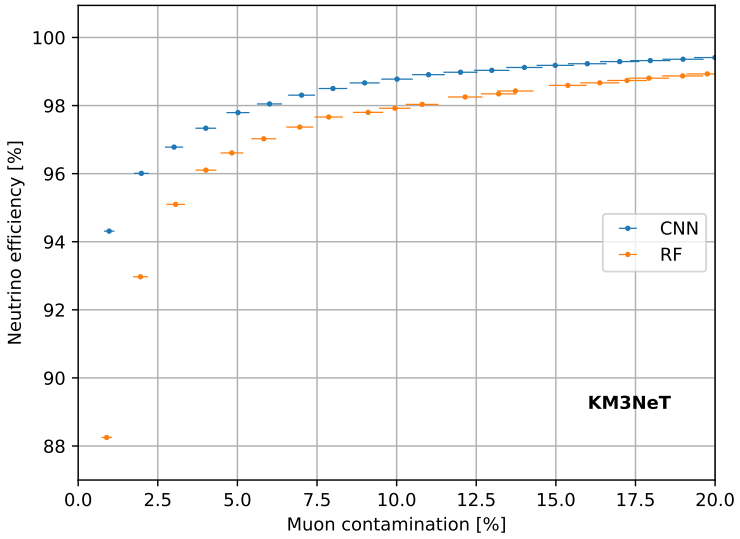

Fig. 7.4: Atmospheric muon contamination C_{μ} and associated neutrino efficiency ν_{eff} based on the pre-selected dataset containing neutrinos with energies ranging from 1 GeV to 100 GeV. The CNN (RF) performance is depicted in blue (orange).

significantly better neutrino efficiency at the same levels of atmospheric muon contamination compared to the atmospheric muon RF classifier. At the same time, the gap in performance between the CNN and the RF depends on the energy of the neutrinos. For lower energetic neutrinos with energies ranging from 1 GeV to 5 GeV, as shown in figure 7.5 (left), the difference in performance between the CNN and the RF is lower compared to the pre-selected dataset containing neutrinos with energies ranging from 1 GeV to 100 GeV. However, the higher energetic

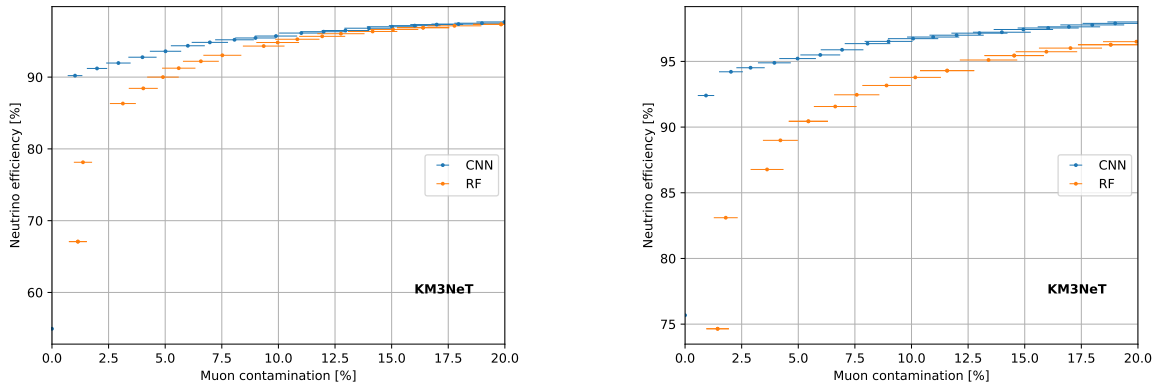

Fig. 7.5: Atmospheric muon contamination C_μ and associated neutrino efficiency ν_{eff} based on the pre-selected dataset containing neutrinos with energies ranging from 1 GeV to 5 GeV (left) or from 10 GeV to 20 GeV (right). The CNN (RF) performance is depicted in blue (orange).

the neutrinos get, the better the deep learning reconstruction performs compared to the standard reconstruction, cf. figure 7.5 (right).

A possible explanation may be that small details in the distribution of the measured hits get more and more important if the neutrino events are less energetic and thus produce less hits. Then, the limitations of the input images, which do not contain the full available information about an event, may become more relevant compared to events with higher energies and many signal hits. However, a similar kind of energy dependent limitation could also exist for the RF classifier. Apart from this, it can be estimated that for both techniques, an atmospheric muon contamination of about 1% can be achieved with a neutrino efficiency in the vicinity of 90%.

The same comparison between the CNN and the RF can be made for random noise events as well, cf. figure 7.6. It can be seen that the rejection of random noise events is easier than for atmospheric muon events, considering the neutrino efficiency of about 99% for a random noise contamination at the 1% level. Additionally, there is no significant difference in performance between the CNN and the random noise RF classifier, though the errors on the random noise contamination are large, due to the limited MC statistics of random noise events that look similar to neutrinos.

7.2 Track-shower classifier

Similar to the background classifier, a Random Forest is used in the current KM3NeT/ORCA reconstruction pipeline to separate track-like events from shower-like events. The RF is commonly referred to as the track-shower classifier of the standard KM3NeT/ORCA reconstruction pipeline, although the classifier is actually trained to distinguish ν_μ^{CC} from ν_e^{CC} and ν_e^{NC}. Thus, based on this terminology, ν_μ^{CC} events are always defined as track-like events, even though a few GeV ν_μ^{CC} event looks similar to a shower-like ν_e event due the instrumentation density of the KM3NeT/ORCA detector. This section introduces a CNN-based track-shower classifier that distinguishes between the same two event classes.

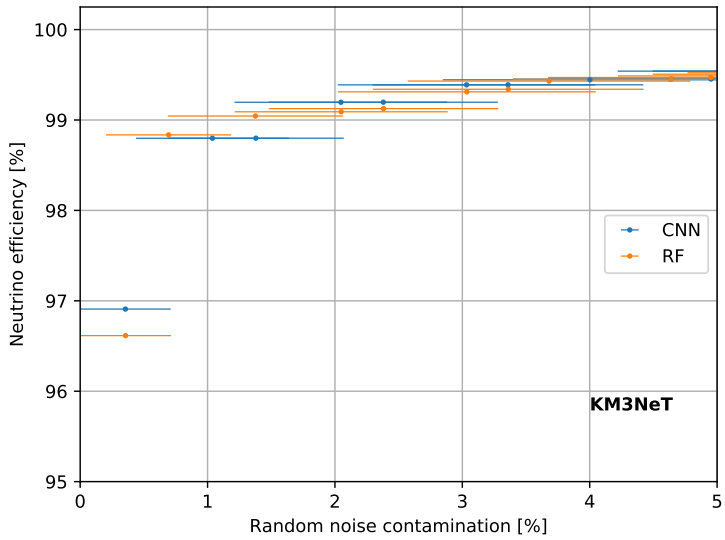

Fig. 7.6: Random noise contamination C_{RN} and associated neutrino efficiency ν_{eff} based on the pre-selected dataset containing neutrinos with energies ranging from 1 GeV to 100 GeV. The CNN (RF) performance is depicted in blue (orange). The error bars are large due to the limited statistics of random noise events in the pre-selected test dataset.

7.2.1 Image generation

The input event images for the CNN-based track-shower classifier are similar to the ones of the background classifier introduced in section 7.1.1, though multiple modifications have been made. First, the timecut for the hit selection is tighter with respect to the background classifier. Since background events have already been rejected by the background classifier, the data presented to the track-shower classifier consists of mostly neutrino interactions. These produce secondary particles that, on average, traverse smaller parts of the detector compared to atmospheric muon events. The event class that shows the signal hit time distribution with the largest variance are ν_μ^{CC} events due to the secondary muon. Thus, the timecut of the track-shower classifier is set based on these events. The time distribution of the signal hits relative to the mean time of all triggered hits for ν_μ^{CC} events is shown in figure 6.3 and the related distributions for all other neutrino interactions, namely ν_e^{CC}, ν_e^{NC} and ν_τ^{CC}, can be found in appendix A.1. Two timecuts are used for the CNN track-shower classifier, timecut 1 with a timecut window of $[-250\,\text{ns}, +500\,\text{ns}]$ and timecut 2 with a tighter timecut window of $[-150\,\text{ns}, +200\,\text{ns}]$. Since both of the two timecut intervals are smaller than the one taken for the background classifier, less time bins are used for binning the time dimension. Compared to the 100 time bins of the background classifier, only 60 time bins are used for the track-shower classifier. Thus, the XYZT images have $11 \times 13 \times 18 \times 60$ pixels, 40% less than the input images for the background classifier CNN. This significantly speeds up the training of the CNN. The time resolution of the images using timecut 1 is 12.5 ns/bin and for timecut 2 the resolution is increased by a factor of two to 5.8 ns/bin.

7.2.2 Network architecture

The network architecture of the track-shower classifier CNN has been modified compared to the background classifier CNN. As already mentioned in section 6.4, it is not clear, how much information is lost by using only a single stacked input consisting of a XYZT and a XYZP projection of the full five-dimensional XYZTP data as an input to the CNN. Likewise, it has been pointed out that the last dimension in the four-dimensional input images, e.g. the time dimension in XYZT images, will not be convolved over in a three-dimensional convolutional layer.

Whether this is detrimental or not for the performance of the CNN is not clear. For further investigations, an additional, permuted XYZT image, the YZTX image, is fed as a separate input to the CNN. Then, the impact of the permuted image on the performance of the CNN can be assessed. An YZTX image contains the same hit information as an XYZT image, but here the X dimension takes the place of the time dimension as the dimension, over which no convolution will take place. From all possible permutations of the XYZT image, the YZTX permutation has been chosen for this investigation, since studies with single-input, track-shower CNNs without stacked XYZP information show that the YZTX projection achieves the best performance, only surpassed by a CNN trained with XYZT input. Since the dimensions of the XYZT and YZTX images differ not only in the last dimension, the channel dimension, but also in the other dimensions, the YZTX image cannot be just put on top of the already stacked XYZT and XYZP images, which together form the single XYZ-T/P input of the CNN. Thus, the YZTX image is fed as a second, separate input to the CNN, which has its own convolutional blocks. In order to connect the information of the convolutional blocks of the first and the second input, the output of the last convolutional layers, i.e. four-dimensional arrays for three-dimensional convolutional layers, of both inputs is flattened to one dimension. Then, these two flattened outputs are combined to a single, one-dimensional array in a so-called concatenate operation. This concatenated output is fed to a fully connected network, such that the whole network forms a so-called multi-input CNN. Figure 7.7 shows a scheme of such a multi-input CNN for the specific case of two inputs.

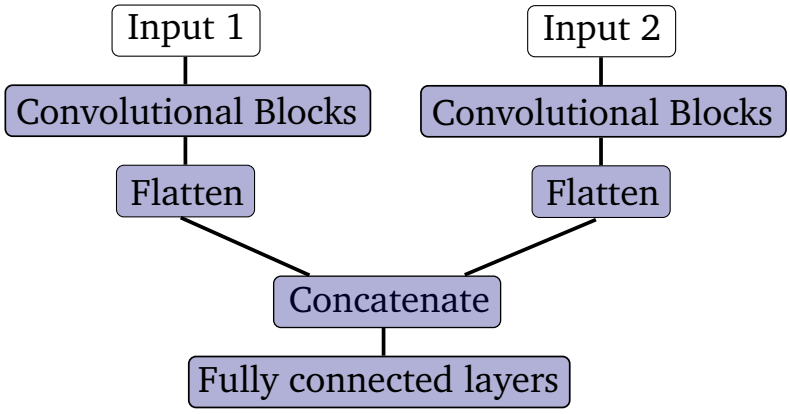

Fig. 7.7: Principle architecture of multi-input CNNs used in this work. For example, input 1 could be an image composed of stacked XYZT and XYZP images, i.e. an XYZ-T/P image, and input 2 could be a YZTX event image.

Another open question is if the network would benefit from a higher resolution in the time dimension. This can be tested by increasing the number of time bins used in the image generation, e.g. increasing the number of bins by a factor of two. However, the more bins the input of the CNN contains, the longer it takes for the network to train. Another possibility is to supply the same image, e.g. XYZT, twice to the CNN, where both images have been made with a different timecut, namely timecut 1 and timecut 2 introduced in section 7.2.1. Hence, the network is a multi-input CNN consisting of two convolutional branches, cf. figure 7.7, where both branches get images with the same four-dimensional XYZT projection, but with different timecut windows and thus different time resolutions.

Multi-input CNNs can have an arbitrary number of inputs, though the computational cost and the effect of the additional inputs on the performance needs to be considered. As a proof of principle, a CNN with four inputs has been designed. The first input is an XYZ-T/P image based on timecut 1. The second input is a YZTX image using timecut 1 as well. Input three and four are the same images, just with a different timecut, namely timecut 2.

In principle, such a multi-input CNN can be trained from scratch. However, the inputs can also be trained separately in single-input CNNs and only be put together later on. The procedure for this technique is as follows:

1. Train both the XYZT and the YZTX networks with timecut 1 and timecut 2 independently.
2. Construct a new multi-input model by using the four pre-trained models from the first step:
   2.1. Remove the fully connected layers of all four pre-trained models.
   2.2. The output of each of the four CNNs is now a four-dimensional array, i.e. three spatial dimensions and one channel dimension. Flatten each four-dimensional array to a one-dimensional array and concatenate them to a single, one-dimensional array.
   2.3. Add randomly initialized, fully connected layers where the first layer takes the single, one-dimensional array as an input.
   2.4. Retrain the fully connected layers of this multi-input CNN. For this purpose, freeze the convolutional blocks of the multi-input CNN, such that all parameters in the layers of the convolutional blocks are fixed. Otherwise, the training of the new fully connected layers can destroy the pre-trained features in the convolutional layers in the beginning of the training.

A visualization for constructing this multi-input CNN, containing the convolutional blocks of pre-trained, single-input CNNs with two inputs and a single timecut is shown in figure 7.8.

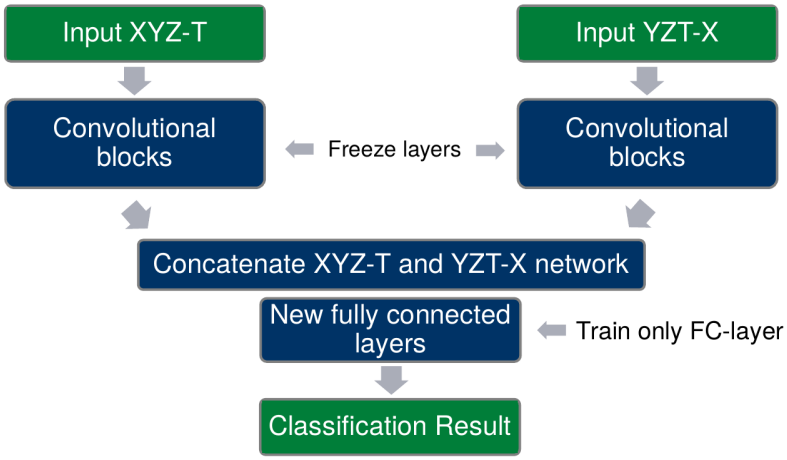

Fig. 7.8: Illustration of the construction of a multi-input CNN with the two inputs XYZT and YZTX. A layer is called *frozen*, when all parameters in the layer are fixed during the training.

It turns out that training the inputs separately in single-input CNNs and only putting them together later on yields a significantly better performance compared to training them from scratch in a multi-input CNN. Additionally, it seems sensible to unfreeze the convolutional layers in the multi-input model, after the new fully connected layers have been sufficiently trained. The reasoning behind this is that the convolutional layers, trained based on single-input CNNs, can then adjust to the multi-input CNN. However, unfreezing the convolutional layers results in a sudden drop of performance of about 2% in accuracy, after only a few batches of training. After this degradation, the model starts to improve slowly again, but it never improves upon the previous performance with frozen convolutional layers. Multiple approaches have been designed in order to fix this issue, like using half-trained convolutional blocks instead of fully-trained ones, but still, the best result has been achieved with frozen convolutional layers. Hence, this technique will be used for all multi-input CNNs in this work.

The network architecture of the multi-input, track-shower CNN is shown in table 7.2. Compared

Building block / layer	Output dimension
Inputs: XYZ-T/P, YZT-X using timecut 1/2	
Convolutional blocks XYZ-T/P, timecut 1, cf. table A.1	$2 \times 3 \times 4 \times 128$
Convolutional blocks XYZ-T/P, timecut 2, cf. table A.1	$2 \times 3 \times 4 \times 128$
Convolutional blocks YZT-X, timecut 1, cf. table A.2	$3 \times 4 \times 7 \times 128$
Convolutional blocks YZT-X, timecut 2, cf. table A.2	$3 \times 4 \times 7 \times 128$
Flatten	3072, 3072, 10752, 10752
Concatenate	27648
Dense 128 + ReLu	128
Dropout ($\delta = 0.1$)	128
Dense 32 + ReLu	32
Dense 2 + Softmax	2

Table 7.2: Network structure of the track-shower, multi-input CNN with four inputs: XYZ-T/P using timecut 1/2 and YZT-X using timecut 1/2. The convolutional blocks are taken from the respective four, pre-trained and single-input CNNs. During the training, the convolutional blocks remain frozen, i.e. the parameters in the layers are kept fixed.

to the background classifier, introduced in section 7.1, the training dataset of the track-shower classifier is significantly smaller than that of the background classifier. Already during the training of the single-input CNNs, which are later put together for the multi-input, track-shower CNN, it can be observed that overfitting occurs without any regularization. Thus, additional dropout layers with a rate of $\delta = 0.1$ have been added to both the convolutional blocks as well as to the fully connected layers. The rate of $\delta = 0.1$ has been chosen based on a small-scale hyperparameter optimization, where rates of $\delta = 0.05$ and $\delta = 0.2$ generally yielded a worse performance.

7.2.3 Preparation of training, validation and test data

In order to train the single-input, track-shower classifier CNNs, as well as the multi-input, track-shower classifier CNNs, simulated neutrino events are used. The total dataset is rebalanced such that 50% of the events are track-like (ν_μ^{CC}) and 50% are shower-like. The shower class consists of 50% ν_e^{CC} and 50% ν_e^{NC} events. Additionally, the dataset has been balanced in a way that the ratio of track-like to shower-like events is always one independently of the neutrino energy. ν_τ^{CC} events are only included in the test dataset. Principally, ν_τ^{CC} events can be included in the training as track- or shower-like events dependent on the decay mode of the τ lepton. However, since the simulated number of ν_τ^{CC} events merely amounts to about 10% of the total simulated neutrino sample, cf. table 6.1, it has been decided to only include them in the test dataset for simplicity. The rebalanced dataset is split into three datasets with 70% training, 6% validation, and 24% test events. In total, the training dataset contains about 13×10^6 events. Training a single input, XYZ-T/P (YZTX) track-shower classifier CNN takes about one and a half weeks (three weeks) using an Nvidia Tesla V100 GPU.

7.2.4 Performance and comparison to Random Forest classifier

In this section, the performance of the track-shower classifier CNN is presented and compared to the performance of the Random Forest classifier from the standard KM3NeT/ORCA reconstruction pipeline. Prior to the comparison of the multi-input CNN with the RF, the performance of the single-input CNNs is reported and compared to the performance of the multi-input CNN.

Performance of single-input and multi-input CNNs

Figure 7.9 shows the loss of the single-input XYZ-T/P CNN with timecut 1. Even though the

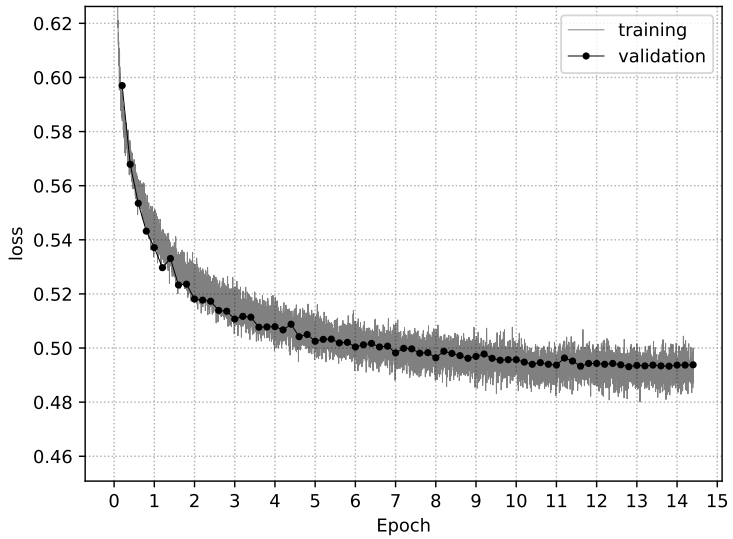

Fig. 7.9: Training and validation cross entropy loss of the XYZ-T/P CNN track-shower classifier with timecut 1 during the training process. Each data point of the training loss curve is averaged over 250 batches, i.e. 250×64 event images.

validation cross entropy loss is in between the statistical fluctuations of the training loss curve at the end of the training, some minor overfitting occurs. The reason is that during the application of the trained network on the validation dataset, no neurons are being dropped by the dropout layers, contrary to the training phase. Therefore, the validation loss should be lower than the average training loss, if no overfitting is observed. This can also be seen by investigating the training and validation loss curves in the earlier stages of the training process, e.g. between epoch 0 and 3. Here, the validation loss is typically found at the bottom of the training loss curve. As introduced in section 6.1, additional simulations have been carried out for this work, since the presented CNNs benefit significantly from a larger training dataset. Based on the XYZ-T/P track-shower CNN using timecut 1, it could be seen that with an increased training dataset, the final validation loss gets significantly better, while the amount of overfitting is reduced at the same time. Thus, the remaining amount of overfitting is a hint that the track-shower CNN may benefit from an even larger training dataset.

Table 7.3 shows the final losses and the accuracy metrics of the single-input CNNs for both the training and the validation dataset.

CNN type	Loss [Train, Val]	Accuracy [Train, Val]
XYZ-T/P, timecut 1	[0.4915, 0.4933]	[74.49%, 74.32%]
XYZ-T/P, timecut 2	[0.4904, 0.4931]	[74.55%, 74.27%]
YZT-X, timecut 1	[0.5151, 0.5130]	[72.38%, 72.46%]
YZT-X, timecut 2	[0.5151, 0.5142]	[72.25%, 72.19%]

Table 7.3: Loss and accuracy metrics of the four single-input CNNs used for the multi-input, track-shower CNN.

As mentioned in section 6.4, the XYZ-T/P networks generally perform better than the YZT-X networks. However, if both the XYZ-T/P and the YZT-X CNNs with timecut 1 are put together in a double-input CNN, cf. table 7.4, there is a small performance gain over the CNN with XYZ-T/P input only. Thus, both networks optimize on slightly different events based on the two

CNN type	Loss [Train, Val]	Accuracy [Train, Val]
2-Inputs: XYZ-T/P + YZT-X, timecut 1	[0.4715, 0.4887]	[75.94%, 74.54%]
4-Inputs: XYZ-T/P + YZT-X, timecut 1 & 2	[0.4610, 0.4795]	[76.69%, 75.24%]

Table 7.4: Loss and accuracy metrics of the double-input and the four-input track-shower CNN.

different projections of the five-dimensional XYZTP event images. The same applies to the two different timecuts that are used for the networks, timecut 1 and timecut 2. Adding two additional timecut 2 inputs, XYZ-T/P and YZT-X, to the double-input XYZ-T/P & YZT-X, timecut 1 CNN yields a gain in performance, cf. table 7.4. Compared to the XYZ-T/P, single-input CNN with timecut 1, an absolute performance gain of 0.0138 in cross entropy loss and 0.92% in accuracy is observed. In principle, all other permutations of the five-dimensional XYZTP event information could be added, albeit diminishing returns may occur. Ultimately though, four-dimensional convolutional layers should be used when they become available in TensorFlow, in order to pass the full event information to a CNN.

To recap, it has been demonstrated that the usage of multiple inputs for CNNs with three-dimensional convolutional layers yields a significant improvement in performance for the track-shower classifier CNN. In the future, these multi-input CNNs can also be used for the already introduced background classifier CNN and for the CNN event regressor, which will be introduced in section 7.3. Thus, the presented performance of the CNN background classifier and the CNN event regressor should be taken as conservative estimates of the potential performance of CNNs for KM3NeT/ORCA.

Comparison to Random Forest classifier

In this section, the performance of the four-input, track-shower classifier CNN will be compared to the performance of the standard KM3NeT/ORCA track-shower RF classifier. Prior to this, the output of the track-shower CNN is investigated. Figure 7.10 shows the binned, predicted probability distribution $P^{\nu}_{i,\text{track}}$ of being a track-like event for all neutrino events with energies in the range of 1 GeV to 40 GeV. This energy range has been chosen, since the classification performance saturates at about 40 GeV, as can be seen in figure 7.11 which is further discussed below. Other than that, the neutrino events have been selected according to the pre-selection criteria introduced in section 7.1.4. It can be seen that ν_μ^{CC} events are predominantly identified as track-like events with a probability close to one. The correct identification of muon tracks scales with their length and hence with their energy. On the other hand, the probability distribution of ν_e events peaks in the region around a probability of 0.18. For ν_τ^{CC}, most events are identified to be shower-like, similar to ν_e events, but some tau neutrinos look very track-like as well, with a CNN track probability close to one. This is as expected, due to the different decay channels of the τ lepton and the associated branching ratios, cf. section 4.1. Apart from this, antineutrino interactions of $\overline{\nu}_\tau^{\text{CC}}$ and $\overline{\nu}_\mu^{\text{CC}}$ interactions show a higher track probability than their neutrino counterparts. The cause is the reaction inelasticity for deep inelastic, charged current antineutrino-nucleon interactions, which is lower on average compared to that of neutrino-nucleon interactions, cf. section 2.4.2. Thus, the secondary lepton carries more energy on average, which makes the whole interaction appear more track-like than for non-antineutrino interactions.

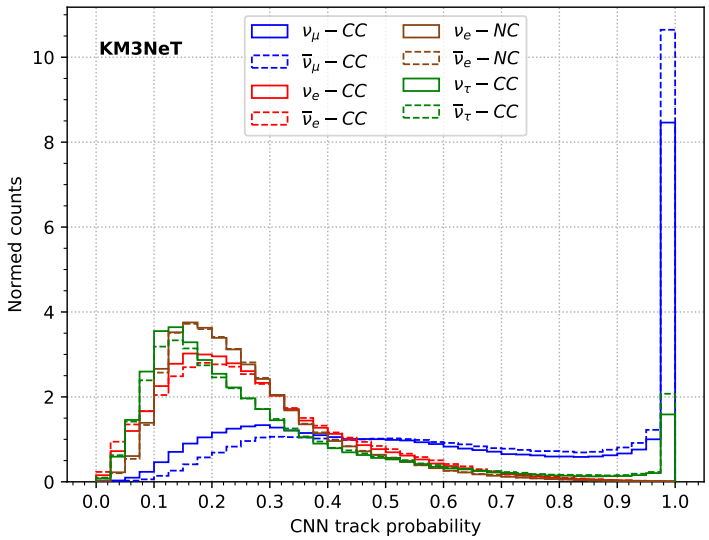

Fig. 7.10: Probability to be classified as a track-like event as determined by the CNN for pre-selected neutrino events of different flavors and interaction channels in the energy range of 1 GeV to 40 GeV.

The top panel of figure 7.11 shows the fraction of events classified as track-like as a function of neutrino energies in the range 1 GeV to 40 GeV. An event is accepted as track-like, if its predicted probability to be track-like by the CNN is 0.5 or above. It can be seen that the classification performance of the classifier depends on the energy of an event. The reason is that the spatial and temporal distribution of hits induced by low-energy ν_μ^{CC} events can be similar to that of shower-like events, since the outgoing muon decays after propagating only a short distance in the detector. Comparing ν_e^{CC} and $\bar{\nu}_e^{CC}$ with energies below 10 GeV, it can be seen that $\bar{\nu}_e^{CC}$ events are more likely classified as track-like by the CNN classifier. This is again due to the fact that the outgoing, charged lepton carries on average more energy in antineutrino charged current interactions with respect to neutrino interactions. On the other hand, for energies below 10 GeV, neutral current interactions have a lower misclassification rate compared to ν_e^{CC}, since there is no charged secondary lepton that could mimic the signature of a low-energy muon as induced in ν_μ^{CC} events. For energies above 10 GeV the opposite is true, since likely the ratio of the light yield to the size of the light source disfavors a ν_μ^{CC} event.

A measure for the classification performance is the distance between the fractions of track-like events for ν_μ^{CC} (blue line) and ν_e (red and brown lines) in the top panel. In the bottom panel of the figure, the CNN and the RF classifier are compared by the relative improvement in this measure. According to this measure, the CNN performs better than the RF for neutrino energies above 3 GeV. The RF performs better than the CNN for energies below 3 GeV. As stated at the end of section 7.1.4, this could again be due to limitations in the input images that become more relevant for low-energy events with only a few signal hits.

Since this performance comparison depends on the threshold value that is set for an event to be classified as track-like, a new comparison metric, that is independent of the threshold value, is introduced. To this end, the probability distributions, already shown in figure 7.10, can be used. In particular, the correlation of the probability distributions $P_{\text{track}}^{\nu_\mu}$ and $P_{\text{track}}^{\nu_e}$ can be investigated. $P_{\text{track}}^{\nu_\mu}$ is defined as the probability that a classifier recognizes a ν_μ^{CC} event as track-like. For $P_{\text{track}}^{\nu_e}$ the same applies, though either charged current or neutral current ν_e events can be taken. Based on the binned track probability distributions $P_{\text{track}}^{\nu_\mu}$ and $P_{\text{track}}^{\nu_e}$, the ability of a classifier to separate track-like and shower-like events can be estimated by investigating the correlation between both distributions for each probability bin. If both distributions are totally anti-correlated, a

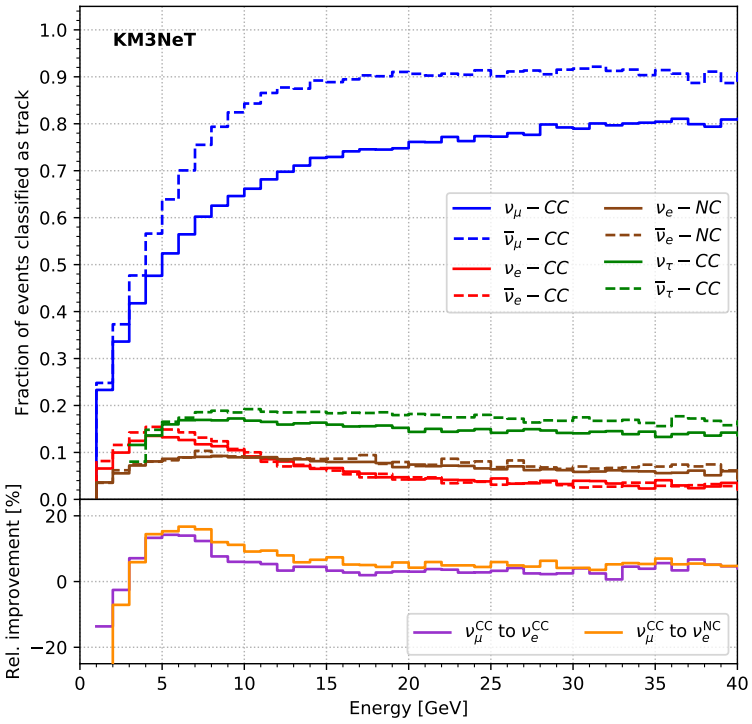

Fig. 7.11: Fraction of events classified as track-like, i.e. with a track probability > 0.5, for different interaction channels (top panel) versus the true MC neutrino energy. The relative improvement of the CNN with respect to the RF classifier in discriminating between neutrino ν_μ^{CC} events and shower-like neutrino ν_e^{CC}/ν_e^{NC} channels is shown in the bottom panel. The discrimination power is defined as the distance between the fractions of events classified as track-like and shower-like in the top panel.

classifier has the best possible performance and vice versa. In order to assess the similarity, the distance between $P_{\text{track}}^{\nu_\mu}$ and $P_{\text{track}}^{\nu_e}$ needs to be measured for each bin. Principally, there are many distance functions that can be used. For example, the Kullback-Leibler divergence, introduced in section 5.2.2, could be utilized. For this work, a measure is used that has already been employed for studies on intrinsic resolution limits in ν_μ^{CC} and ν_e^{CC} interactions of the KM3NeT/ORCA detector [87].

This so-called *separability* measure in a certain energy range ΔE is calculated as follows:

$$S(\Delta E) = 1 - c(\Delta E) = 1 - \frac{\sum_i P_{i,\text{track}}^{\nu_\mu}(\Delta E) \cdot P_{i,\text{track}}^{\nu_e}(\Delta E)}{\sqrt{\sum_i \left(P_{i,\text{track}}^{\nu_\mu}(\Delta E)\right)^2 \cdot \sum_i \left(P_{i,\text{track}}^{\nu_e}(\Delta E)\right)^2}} \tag{7.5}$$

Here, $c(\Delta E)$ is the correlation factor of the two probability distributions $P_{i,\text{track}}^{\nu_\mu}$ and $P_{i,\text{track}}^{\nu_e}$, which is calculated as a sum over all bins i. Figure 7.12 shows the separability $S(\Delta E)$ of ν_μ^{CC} and ν_e^{CC} as well as ν_μ^{CC} and ν_e^{NC} as a function of bins in neutrino energy for the CNN and the RF classifier. For both the CNN as well as the RF classifier, it can be seen that the separability of ν_μ^{CC} and ν_e^{NC} is better than that of ν_μ^{CC} and ν_e^{CC} below energies of 7 GeV. Above energies of 7 GeV, the behavior is opposite, for the reasons that have already been given in the discussion of figure 7.11. Apart from this, the CNN-based classifier performs better in this metric by an absolute value of up to about 30% for energies below 10 GeV. In the end, the impact of the improved track-shower classification depends on the physics analysis that is carried out, e.g. the study of tau neutrino appearance. Due to the apparent performance advantage of the CNN track-shower

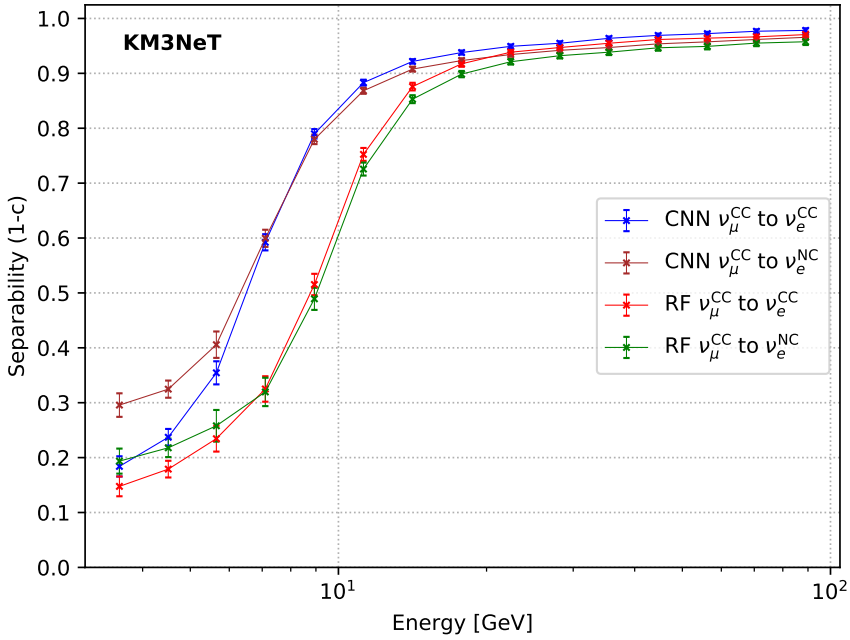

Fig. 7.12: Separability $S(\Delta E) = 1 - c(\Delta E)$ based on ν_μ^{CC} and ν_e^{CC} (ν_μ^{CC} and ν_e^{NC}) events as a function of true MC neutrino energy for the CNN represented by blue (brown) markers and the RF represented by red (green) markers. Only neutrinos have been used for the comparison. The error bars indicate statistical errors.

classifier over the RF classifier, significant efforts have been devoted in the KM3NeT/ORCA collaboration to further improve the performance of the RF classifier. And indeed, improvements could be made by the design of new, hit-based features that are fed to the RF. Unfortunately, the recently improved RF cannot be compared directly to the presented track-shower classifier CNN, since the new RF has only been applied to a different set of simulations. Nevertheless, the presented track-shower CNN is also not fully optimized and further improvements can be expected. For example, if four-dimensional convolutional layers become available in TensorFlow, the full event information could be supplied to the CNN. Other than that, more sophisticated model architectures like Inception [71] networks could lead additional gains in performance. And finally, a full hyperparameter optimization, e.g. with respect to the number of layers, number of filters, kernel size, etc. has to be carried out in the future, potentially leading to an improved performance.

7.3 Regression of neutrino properties

This section introduces the CNN-based *event regressor*. The employed CNN reconstructs properties of the interacting neutrinos, i.e. the direction, the energy and the position, also called *vertex*, of the interaction. Additionally, an estimate for the uncertainty on each of these reconstructed observables is inferred. In the standard reconstruction pipeline for KM3NeT/ORCA [12], these reconstruction tasks are handled by two separate, maximum likelihood-based algorithms optimized for track-like and shower-like events, respectively. For the KM3NeT/ORCA neutrino mass hierarchy analysis, cf. [88], the track (shower) reconstruction algorithm is used for events that are classified as track-like (shower-like) events by the RF track-shower classifier. In the following, this algorithm for the reconstruction of track-like (shower-like) events will be referred to as TOR (SOR).

The major difference of the CNN event regressor with respect to the classifiers in the previous sections is that its output consists of continuous variables instead of a single categorical variable.

7.3.1 Image generation

For the CNN event regressor, the event images are created similar to the image generation for the background classifier. Thus, both XYZT and XYZP images are produced and then stacked in the last dimension, resulting in XYZ-T/P images. Since the background classifier has already been applied to the data, at this final stage of the reconstruction pipeline predominantly neutrinos remain as input to the event regressor. Hence, the timecut for the image generation of the CNN event regressor is based on ν_μ^{CC} events like for the CNN track-shower classifier, cf. section 7.2.1. Since timecut 1 shows a better performance compared to timecut 2 for the CNN track-shower classifier, cf. section 7.2.4, timecut 1 has been used for the image generation of the CNN event regressor images.

7.3.2 Network architecture and loss functions

The network architecture of the CNN event regressor is identical to the one of the CNN background classifier, apart from a few modifications. For example, no dropout layers are used. For this work, it has been found that dropout layers in the CNN event regressor lead to increased fluctuations in the training loss, even with a small dropout rate of $\delta = 0.05$. Furthermore, the validation loss of the event regressor CNNs with dropout layers has proven to be significantly worse than for CNNs without dropout. Apart from this, the fully connected layers, connected to the output of the last convolutional layer, are modified as well. The properties that should be reconstructed by the network are the energy, the direction and the vertex position of the initial neutrino interaction. For the direction and the vertex reconstruction, the CNN output consists of a one-dimensional array with three elements, containing the X, Y and Z components of the reconstructed direction or vertex position vector. Consequently, the output of the network consists of seven reconstructed floating point numbers, denoted as $\vec{y}_{\text{reco}}$, one component for the energy, three for the direction and three for the vertex position. The final fully connected output layer therefore consists of seven neurons. The CNN architecture up to this point is shown in table 7.5.

Since the aim of this network is to reconstruct continuous variables, a categorical cross entropy loss, as employed for the background and the track-shower classifier, cannot be used. Instead, other cross entropy-based loss functions can be used, e.g. the mean squared error (MSE) or the mean absolute error (MAE), respectively defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_{\text{reco}} - y_{\text{true}}\right)^2 , \tag{7.6}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_{\text{reco}} - y_{\text{true}}\right| . \tag{7.7}$$

The difference between the MSE and the MAE loss is that the MSE loss punishes outlier events that are badly reconstructed with a significantly larger loss value compared to the MAE loss. Since the light signatures of many triggered events are not fully contained in the detector, the reconstruction quality is intrinsically limited, such that these events are on average less well reconstructed. Thus, the MAE loss is used for the energy, the direction and the vertex regression, so that the network focuses less on outlier events.

Using a neural network, it is also possible to estimate the uncertainties on the components of $\vec{y}_{\text{reco}}$. One possibility is to let the network predict the absolute residual of the MAE loss. Thus, an additional neuron that yields the reconstructed uncertainty, σ_{reco}, for each of the reconstructed components of $\vec{y}_{\text{reco}}$ is needed. Consequently, the loss function that measures the quality of the reconstructed uncertainty value is as follows:

$$L = \frac{1}{n}\sum_{i=1}^{n}\left(\sigma_{\text{reco}} - \left|y_{\text{true}} - y_{\text{reco}}\right|\right)^2 . \tag{7.8}$$

Building block / layer	Output dimension		
XYZ-T Input	$11 \times 13 \times 18$	$\times$	60
XYZ-P Input	$11 \times 13 \times 18$	$\times$	31
Final stacked XYZ-T + XYZ-P Input	$11 \times 13 \times 18$	$\times$	91
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters, $\rho = (2, 2, 2)$)	$5 \times 6 \times 9$	$\times$	64
Convolutional block (128 filters)	$5 \times 6 \times 9$	$\times$	128
Convolutional block (128 filters)	$5 \times 6 \times 9$	$\times$	128
Convolutional block (128 filters)	$5 \times 6 \times 9$	$\times$	128
Convolutional block (128 filters, $\rho = (2, 2, 2)$)	$2 \times 3 \times 4$	$\times$	128
Flatten	3072		
Dense 128 + ReLu	128		
Dense 32 + ReLu	32		
Dense 7 + Linear	7		

Table 7.5: Network structure of the CNN event regressor with XYZ-T/P input, but without the additional fully connected network for the uncertainty estimation. In the convolutional blocks and the last fully connected layers no dropout is used.

Using this loss function L, the network learns to estimate the absolute residual on average:

$$\sigma_{\text{reco}} \approx \langle y_{\text{abs}} \rangle \ . \tag{7.9}$$

Assuming that the residuals are normally distributed, the reconstructed uncertainty value σ_{reco} can be converted to a standard deviation by multiplying with $\sqrt{\frac{\pi}{2}}$ [89].

In general, it can be expected that features that are learned by the convolutional part of the network, which are useful for the reconstruction, are also crucial for the inference of the uncertainty on the reconstruction. Therefore, both inference tasks can be handled by the same CNN. However, the CNN must be prevented from focusing too much on the uncertainty estimation in the learning process, since its main goal is the prediction of $\vec{y}_{\text{reco}}$ and not the prediction of the uncertainties. For this purpose, a second fully connected network with three layers is added after the convolutional output, but the gradient update during the backpropagation stage is stopped after the first fully connected layer of this sub-network, cf. figure 7.13. The specific structure of the fully connected network for the uncertainty reconstruction is shown in table 7.6.

7.3.3 Preparation of training, validation and test data

In order to train the CNN event regressor, only simulated neutrino events are used. Similar to the track-shower classifier, the dataset consists of 50% track-like events (ν_μ^{CC}) and 50% shower-like events (25% ν_e^{CC}, 25% ν_e^{NC}). Additionally, the total dataset is again split into a training, validation and test dataset. As for the track-shower classifier, the training datasplit makes up 70%, the validation dataset 6% and the test dataset 24% of the total rebalanced dataset. The

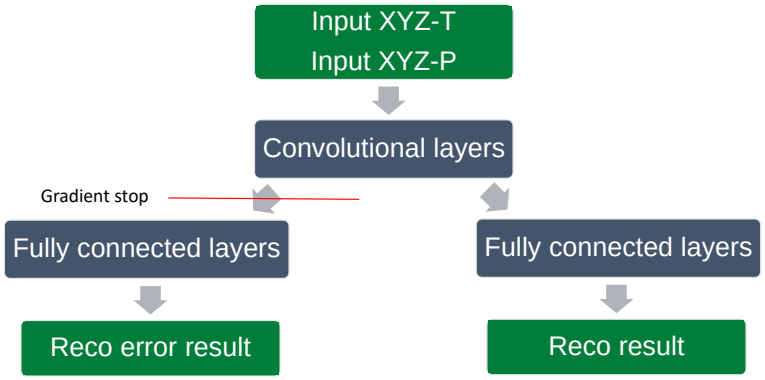

Fig. 7.13: Scheme of the CNN architecture used for the event regressor. The properties of the convolutional layers and of the fully connected network for the reconstruction of $\vec{y}_{\text{reco}}$ are depicted in table 7.5 and in table 7.6.

Building block / layer	Output dimension
Dense 128 + ReLu	128
Dense 64 + ReLu	64
Dense 32 + ReLu	32
Dense 7 + Linear	7

Table 7.6: Layout of the fully connected uncertainty reconstruction network, which is part of the event regressor CNN with XYZ-T/P input. The input to this sub-network is the flattened output of the last convolutional layer of the main CNN.

track and shower classes have not been balanced in energy, in order to get a larger training dataset compared to the track-shower classifier, cf. section 7.2.3. For ν_e^{NC} events, the true energy, which should be reconstructed by the CNN, is set to the visible energy during the training, since the network cannot infer how much energy is carried away by the outgoing neutrino in the neutral current interaction. In addition, ν_τ^{CC} events are only included in the test dataset, similar to the background and the track-shower classifier. For ν_τ^{CC} events, the amount of non-visible energy depends on the decay mode of the τ lepton. Thus, ν_τ^{CC} events are only included in the test dataset for simplicity.

As shown in table 6.1, the available neutrino simulations consist in a low-energy production with neutrino energies from 1 GeV to 5 GeV and a high-energy production with neutrino energies from 3 GeV to 100 GeV. Thus, a significant overlap in the energy range from 3 GeV to 5 GeV exists, where the number of triggered events is already high compared to higher energies due to the power-law in neutrino energy that is used for the generation of the neutrino events. At the same time, the CNN uses every kind of information that is given to it, as long it is beneficial to the loss. Thus, the network implicitly learns the energy distribution of the neutrinos in the training dataset. For example, the network learns that neutrino energies below 1 GeV and above 100 GeV do not exist in the training dataset and thus, the CNN will never yield reconstructed energies outside of the 1 GeV to 100 GeV energy range, cf. section 7.3.5. This particular case is only a minor issue, since the simulations can be produced with a larger neutrino energy range if necessary. For KM3NeT/ORCA, this issue matters even less, for two reasons. First, neutrinos with energies below 1 GeV are very rarely triggered, which is also the reason why the lower energy bound for the neutrinos has been set to 1 GeV and not to a lower value. And second, high energy

neutrinos close to 100 GeV do not contribute significantly to the sensitivity of KM3NeT/ORCA on the neutrino mass hierarchy and the same applies to the appearance of tau neutrinos as well, cf. figure 4.4. However, not only the limits of the neutrino energy spectrum but also the energy spectrum itself in the training data will be learned by the CNN. In order to avoid a biased energy reconstruction, the number of neutrinos in the training dataset per energy bin should be the same. For the available neutrino simulations summarized in table 6.1, the of such a training dataset is not possible due to the power-law that was used to simulate the energy spectrum of the neutrinos. At the same time, it has been observed that using all available neutrino simulations, i.e. including all neutrinos in the overlap region of 3 GeV to 5 GeV, leads to distinct features in the distribution of the reconstructed energies by the CNN. Further information on this circumstance can be found in section 7.3.5. By removing neutrinos from the low-energy production in the 3 GeV to 5 GeV energy range and thus by removing the overlap between the low- and the high-energy simulations, it could be observed that this effect is significantly reduced. Thus, for the training and validation dataset, 3 GeV to 5 GeV neutrinos from the low-energy simulations have been removed. Neutrinos that have been deselected in this way are added to the test dataset. All figures that are shown in the performance evaluations later on are based on a CNN trained with this deselection if not indicated otherwise. In total, the training dataset contains about 12×10^6 events. Using an Nvidia Tesla V100 GPU, it takes about one and a half weeks to train the CNN event regressor.

7.3.4 Loss functions and loss weights

Prior to the training of the CNN event regressor, suitable *loss weights* need to be defined, in order to scale the losses for each of the CNN outputs to the same magnitude. For example, the neutrino energy in the dataset ranges from 1 GeV to 100 GeV, while the components of the direction vector range from -1 to 1. Thus, the impact of the energy loss would be overwhelmingly large compared to the loss that is contributed by the direction reconstruction. Consequently, the individual losses of the components of $\vec{y}_{\text{reco}}$ and their reconstructed uncertainties need to be scaled with a loss weight, such that all single losses are of the same order of magnitude at the start of the training. However, it has been observed that the CNN sometimes does not start to learn during the first few epochs of the training, if the individual losses of the energy, direction and vertex reconstruction are of the same scale. The reason is that the direction and vertex reconstruction contributes six out of seven variables that should be learned, so that their contribution to the total loss is dominant. Hence, any improvement with respect to the energy reconstruction, i.e. in the discrimination of background and signal hits, is marginalized by random statistical fluctuations in the loss for the direction or vertex components.

It is expected that one of the first concepts that the neural network will learn during the training is the difference between signal and background hits. At the same time, the recognition of signal hits is a prerequisite for the direction and vertex reconstruction. Thus, the loss weights are set in such a way that the scale of the energy loss at the start of the training is about two times larger than the scale of the three combined directional losses. For the same reason, the loss weights of the vertex reconstruction are set in a way that they have the same scale as the directional losses. Indeed, it could be observed that the energy loss always converges significantly earlier during the training than the loss for the direction or vertex. Since the scale of the individual losses changes dramatically after the network has started to learn a property such as the energy or the direction, the loss weights need to be tuned during the training. As a general strategy, it has proven useful to do this after the energy loss has sufficiently converged. How often and by how much the individual loss weights are tuned is a parameter optimization process. For this work, only a minor amount of optimization has been executed. In particular, the loss weights of the direction and vertex components have been increased by about a factor of three after the convergence of the energy loss. At the end of the training, these loss weights have been doubled again to investigate if further improvements can be made. Even though the direction and vertex losses are significantly larger than the energy loss after the tuning, the scale of the

energy loss is still large enough such that the energy reconstruction does not significantly worsen during the further training. Apart from this, the loss weight for the reconstructed z-coordinate of the direction vector has been increased by about 25% compared to the loss weights of the x- or y-coordinates of the direction vector. The reason is that the zenith reconstruction and thus the reconstruction of the z-coordinate of the direction vector is important for the sensitivity of KM3NeT/ORCA on the neutrino mass hierarchy and the appearance of tau neutrinos. At the same time, the azimuth reconstruction and thus the reconstruction of the x and y-coordinates of the direction vector is not relevant for both science goals mentioned above.

For the loss weights of the uncertainty outputs none of this matters, since their gradients are stopped after the first fully connected layer of the uncertainty reconstruction sub-network. Thus, the loss weights for these variables just need to be set in such a way that the scale of the individual losses is about the same.

7.3.5 Energy reconstruction performance

In this section, the performance of the CNN event regressor for the neutrino energy reconstruction is investigated. This investigation is done separately for the different neutrino flavors and interaction channels. The events have been selected according to the criteria detailed in section 7.1.4. In particular, only events reconstructed as up-going by the standard KM3NeT/ORCA reconstruction algorithms are selected for the following comparisons. In addition to the selection introduced in section 7.1.4, ν_μ^{CC} (ν_e^{CC}, ν_e^{NC}, ν_τ^{CC}) must have been reconstructed with high confidence by the standard KM3NeT/ORCA maximum likelihood reconstruction algorithm for track-like (shower-like) events.

Figure 7.14 (left) shows the reconstructed energy by the CNN versus the true MC energy for ν_μ^{CC} events. This distribution can be compared to figure 7.14 (right) which shows the same information for TOR. Based on the two figures it can already be seen that the TOR distribution

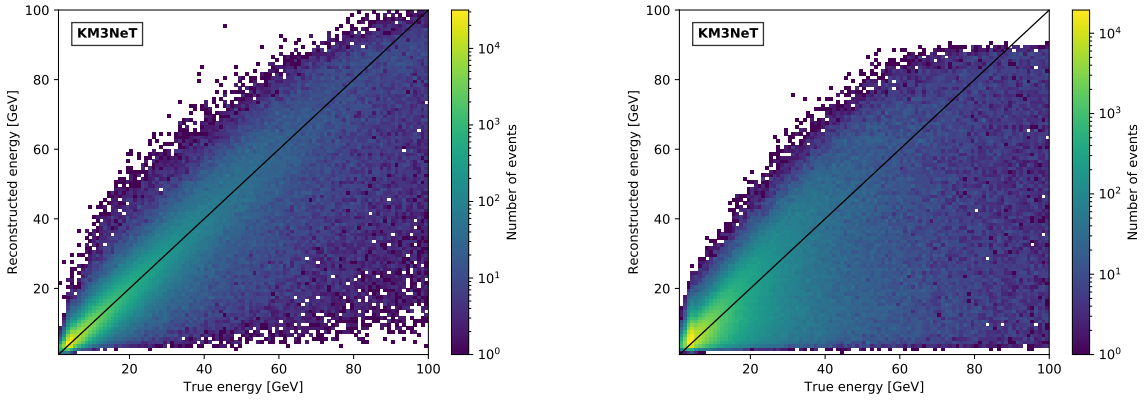

Fig. 7.14: Reconstructed energy vs. true MC neutrino energy for pre-selected ν_μ^{CC} events. Left: CNN event regressor. Right: standard KM3NeT/ORCA maximum likelihood reconstruction algorithm for track-like events (TOR).

shows a larger variance of reconstructed values for a given true MC energy. Figure 7.15 shows the reconstructed energy of the CNN versus the true energy for pre-selected ν_e^{CC} events. As can be seen in the figure, the energy reconstruction tends to underestimate the true MC energy of the neutrino, in particular for high energies above 80 GeV. This is even more evident for SOR, so that a correction function is applied to the SOR energy reconstruction. This correction function has been derived from MC simulations of ν_e^{CC} events and depends on several reconstructed quantities, such as the energy, the zenith angle, and the interaction inelasticity [22].

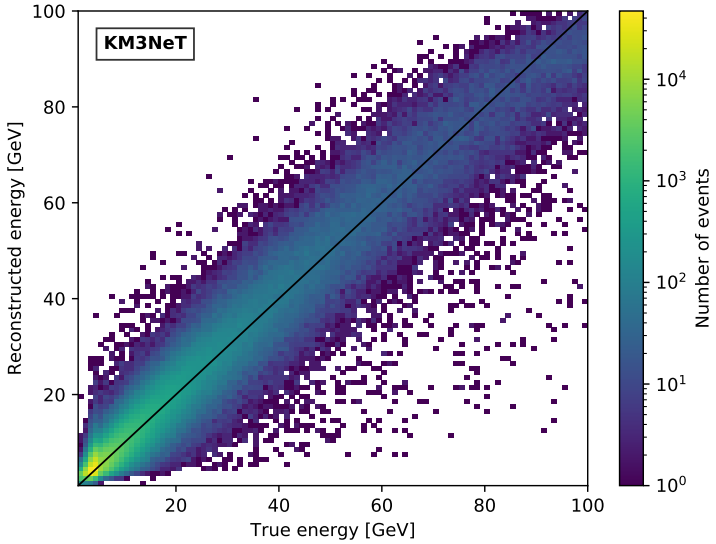

Fig. 7.15: Reconstructed energy of pre-selected ν_e^{CC} events by the CNN event regressor vs. true MC neutrino energy.

In order to allow for a comparison between the SOR algorithm and the CNN event regressor, a similar but simpler correction function, only using the reconstructed energy, has been derived and applied to the energy reconstruction of the CNN event regressor for ν_e^{CC} events. This is only done for a better comparison to the SOR algorithm, since the CNN could easily learn such a correction by itself if it is beneficial for the loss. Since the CNN energy reconstruction will be compared to the SOR energy reconstruction for all neutrino interactions with the exception of ν_μ^{CC} events, the correction is applied for all of these interaction types (ν_e^{CC},ν_e^{NC}, ν_τ^{CC}). The resulting reconstructed and corrected neutrino energy versus the true neutrino energy of ν_e^{CC} events for both the CNN as well as for the SOR algorithm is shown in figure 7.16.

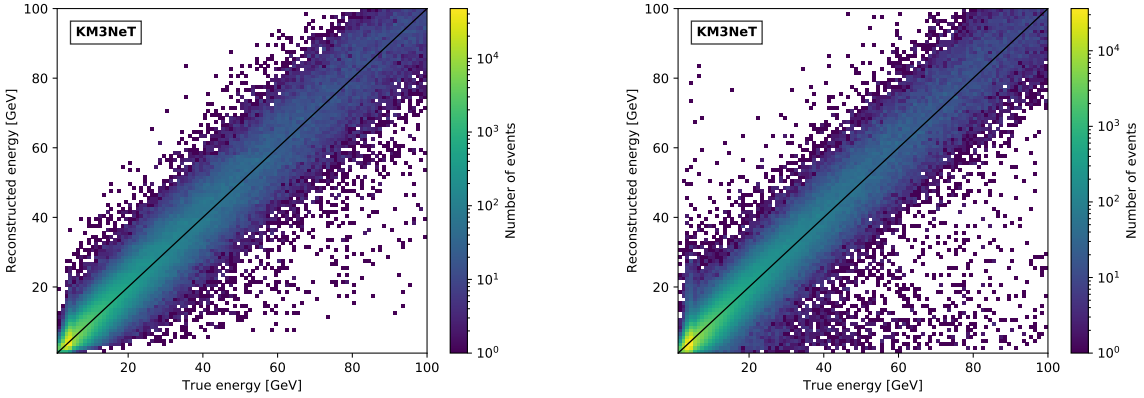

Fig. 7.16: Corrected, reconstructed energy vs. true MC neutrino energy for pre-selected ν_e^{CC} events. Left: CNN event regressor. Right: standard ORCA maximum likelihood reconstruction algorithm for shower-like events (SOR).

As can be seen from comparing figure 7.16 and figure 7.14, the energy reconstruction for shower-like ν_e^{CC} events is better than that for track-like events. In addition, the performance of the CNN event regressor and the SOR algorithm is on a similar level. The results for ν_e^{NC} and ν_τ^{CC} events can be found in the appendix in figure A.7 and figure A.12. As expected, the CNN as well as the

SOR algorithm underestimate the energy of ν_e^{NC} interactions due to the outgoing neutrino in the interaction, which carries away a part of the energy. For ν_τ^{CC} the same behavior can be seen, though the effect on the energy reconstruction is not as evident as for ν_e^{NC} interactions.

In order to derive a quantitative comparison of the CNN event regressor and either the TOR or the SOR algorithm, two performance metrics are used, the *median relative error* (MRE) and the *relative standard deviation* (RSD). Both metrics are calculated for each true energy bin. The MRE is defined as:

$$\text{Median relative error} = \text{Median}\left(\left|\frac{E_{\text{reco}} - E_{\text{true}}}{E_{\text{true}}}\right|\right) . \tag{7.10}$$

For the RSD, the standard deviation σ of the reconstructed energy distribution for each MC true energy bin is calculated and then divided by the energy of the bin:

$$\text{Relative standard deviation} = \frac{\sigma}{E_{\text{true}}} . \tag{7.11}$$

As already mentioned in section 7.3.3, the CNN learns the energy spectrum of the neutrinos in the training dataset, if it is beneficial for the loss. The result of this can be best seen for the MRE of ν_e^{CC} events without a correction of the reconstructed energy. Figure 7.17 (left) shows this for a CNN, which has been trained with a training dataset, which includes the already mentioned 3 GeV to 5 GeV overlap from the low- and the high-energy neutrino simulations.

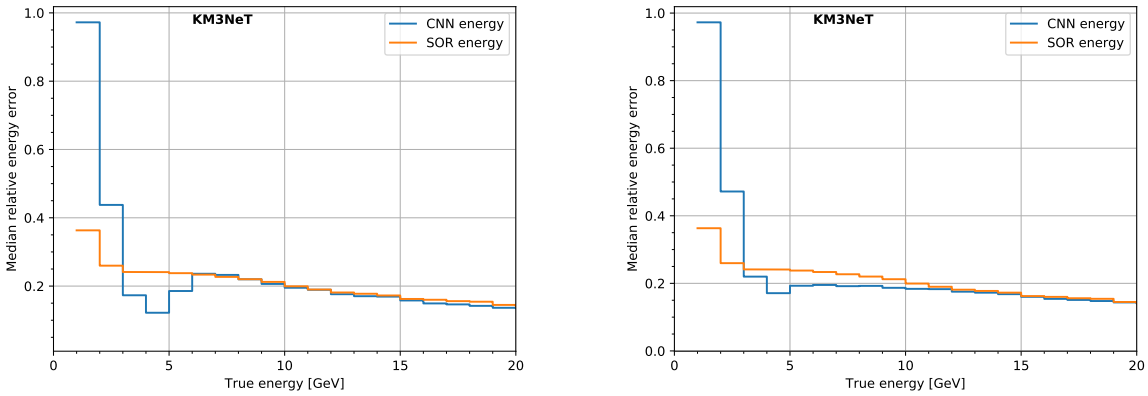

Fig. 7.17: Median relative error of non-corrected, reconstructed CNN energies per true energy bin for pre-selected ν_e^{CC} events and a CNN trained with (left, blue color) and without (right, blue color) the 3 GeV to 5 GeV overlap from the low- and the high-energy neutrino simulations. The MRE of the associated SOR energy reconstruction is shown in orange color, such that it can be used as a baseline for the CNN energy reconstruction. Left: a dip in MRE from about 3 GeV to 6 GeV can be seen. Right: the dip in MRE is significantly reduced due to the deselection of 3 GeV to 5 GeV neutrino events of the low-energy neutrino simulation dataset.

For shower-like events, the energy reconstruction performance is intrinsically limited by light yield fluctuations of the hadronic particle cascade [87]. For the SOR algorithm, it has been shown that the energy resolution is close to this intrinsic limit [87]. Assuming that the CNN event regressor comes close to this intrinsic resolution limit as well, it is to be expected that the MRE and RSD performances for both the CNN event regressor as well as for the likelihood-based approach agree. Thus, the MRE of the SOR algorithm in figure 7.17 (left, orange color) can be taken as a baseline for the performance of the CNN event regressor. For the SOR algorithm, the MRE in energy improves with higher neutrino energies, which is to be expected. For the CNN though, there is a

dip in MRE from 3 GeV to 6 GeV, similar to the energy range of the 3 GeV to 5 GeV overlap from the low-energy and the high-energy neutrino simulations. This dip can be significantly reduced by removing 3 GeV to 5 GeV neutrinos from the low-energy neutrino simulations in the training dataset, cf. figure 7.17 (right, blue).

Other than this, it can be seen that the MRE for the CNN is significantly worse for energies below 3 GeV. The reason for this is that the number of events in the training dataset with energies below 3 GeV is low, since such low energetic events are only rarely triggered. For example, the number of events in the 1 GeV to 2 GeV (2 GeV to 3 GeV) energy range is more than 10 (4) times lower than the number of events in the 3 GeV to 4 GeV energy range. Thus, the CNN tends to reconstruct these low energetic events as higher energetic events, since they are far more frequent in the training dataset. Ultimately, an CNN for the reconstruction of the neutrino energy should be trained with a uniformly distributed neutrino energy spectrum in order to avoid this bias, if the available simulated data makes this possible, cf. section 7.3.3.

At this point, the MRE and the RSD can be compared for the different neutrino interactions. For ν_μ^{CC} events, the MRE is shown in figure 7.18 (left), while the result for the RSD is shown in figure 7.18 (right). Note that the RSD is only a measure for the widths of the reconstructed energy distributions per true energy bin, which, in particular for the TOR algorithm, are highly asymmetric.

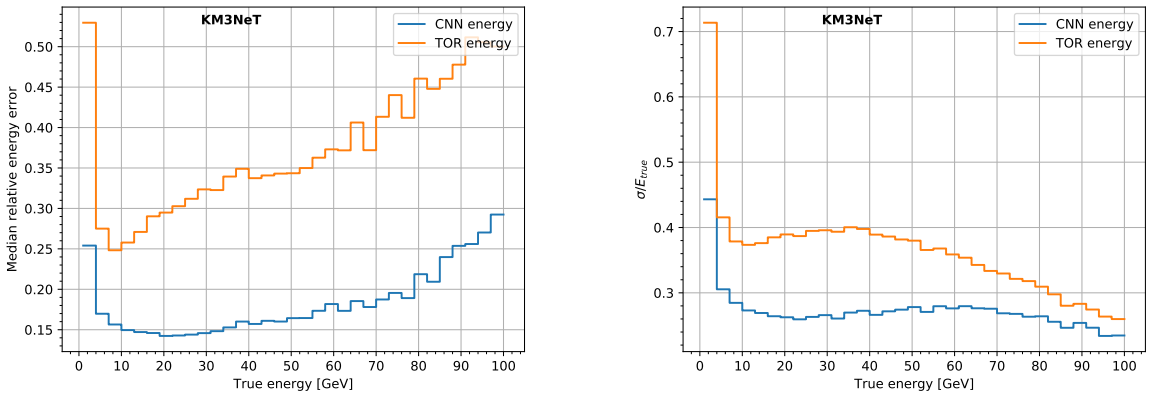

Fig. 7.18: Left: median relative error of the reconstructed energy for pre-selected ν_μ^{CC} events for the CNN (blue) and TOR (orange). Right: relative standard deviation of the reconstructed energy for pre-selected ν_μ^{CC} events for the CNN (blue) and TOR (orange).

The absolute improvement in MRE between the CNN and TOR increases with energy from about 10% at 10 GeV to about 20% at 75 GeV. The RSD almost flattens at about 28% in a broad range of energies for the CNN event regressor. Based on both the MRE as well as the RSD, it can be seen that the CNN energy reconstruction significantly outperforms the TOR algorithm. For the TOR reconstruction algorithm a track hypothesis is assumed and more details can be found in [12]. Recent investigations with a new MC dataset for a more densely instrumented detector, i.e. 20 m of horizontal spacing between the DUs compared to the 23 m of the simulations in this work, show that the SOR algorithm, applied to ν_μ^{CC} events, shows a significantly better energy reconstruction performance than the TOR algorithm. Thus, the TOR algorithm is currently further improved.

Figure 7.19 shows the same metrics for ν_e^{CC} events. For the MRE, the performance of the CNN event regressor and the SOR algorithm agrees very well. The slight improvement for the CNN in the MRE compared to the SOR algorithm below 10 GeV and above 90 GeV may be based on the fact that the CNN implicitly learns the limits of the energy spectrum of the simulated

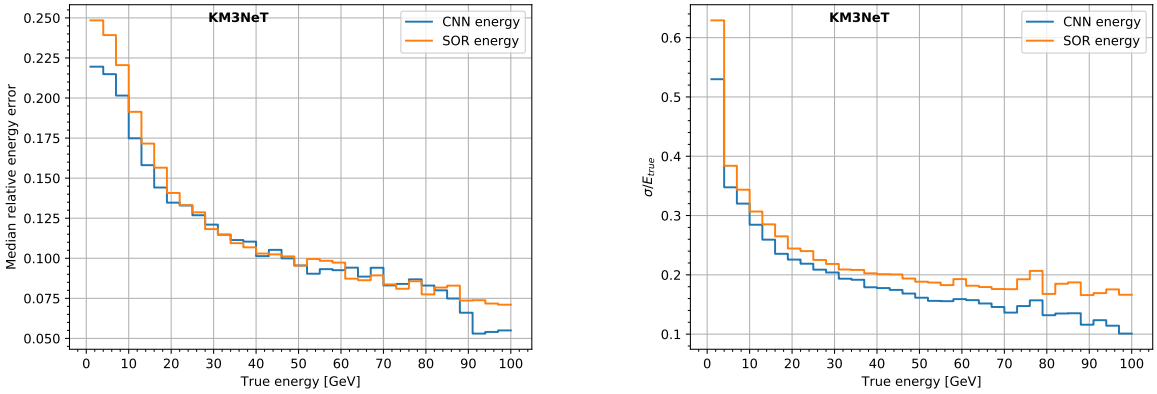

Fig. 7.19: Left: median relative error of the reconstructed energy for pre-selected ν_e^{CC} events for the CNN (blue) and SOR (orange). Right: relative standard deviation of the reconstructed energy for pre-selected ν_e^{CC} events for the CNN (blue) and SOR (orange).

neutrinos, i.e. 1 GeV and 100 GeV. For the RSD, it is notable that the CNN event regressor improves by a few percent with increasing true neutrino energy with respect to SOR. However, the difference is not large, such that the CNN and the SOR energy reconstruction for ν_e^{CC} events can be considered to be on the same level of performance. In the case of ν_e^{NC} events, the energy reconstruction performance is limited by the non-detectable energy that is carried away by the outgoing neutrino. Figure A.8 shows the MRE and the RSD of ν_e^{NC} events for the CNN and the SOR algorithm. For the MRE, there is a slight improvement of the CNN compared to SOR up to energies of about 50 GeV. A possible reason for this may be that the CNN is able to distinguish internally between ν_e^{NC} and ν_e^{CC} events. Then, the network may learn the bjorken-y distribution of the ν_e^{NC} events in the training dataset, which can give an edge over the SOR algorithm. For the RSD, there seems to be a slight improvement as well, but this time over the whole energy range. It is not possible to tell if this is due to a significant improvement on the SOR algorithm or if this is based on the fact that the CNN utilizes the information from the distributions in the training dataset, i.e. the bjorken-y distribution, while the SOR algorithm does not. For a more precise performance comparison, the CNN would need to be trained with a dataset with uniform distributions of neutrino energy and reaction inelasticity y. The same applies to the MRE and RSD of ν_τ^{CC} events, cf. figure A.13.

7.3.6 Direction reconstruction performance

The direction reconstruction performance of the CNN event regressor is investigated in the same way as for the energy reconstruction. For the investigation of the direction reconstruction performance, the X,Y and Z components of the reconstructed direction are converted to spherical azimuth and zenith coordinates. As already mentioned, the azimuth reconstruction is not important for the science goals of KM3NeT/ORCA, e.g. for the neutrino mass hierarchy or the tau neutrino appearance analysis. Thus, this section mostly focuses on the discussion of the zenith angle reconstruction performance.

Figure 7.20 shows the reconstructed zenith angle versus the true MC zenith angle for ν_μ^{CC} events for both the CNN event regressor (left) and the TOR algorithm (right). As can be seen in the right hand part of the figure, only events with a TOR-reconstructed zenith larger than zero (up-going events) have been selected for the comparison, cf. section 7.3.5. The CNN-based reconstruction results in a significantly tighter distribution compared to TOR. A similar behavior is found for the azimuth angle reconstruction, cf. figure 7.21. The associated distributions for the azimuth and the zenith angle for ν_e^{CC}, ν_e^{NC} and ν_τ^{CC} events can be found in figure A.5 and A.6, figure A.9 and A.10 and figure A.14 and A.15. In general, it can be seen that the CNN produces less reconstructed events with a very high reconstruction error compared to TOR and SOR.

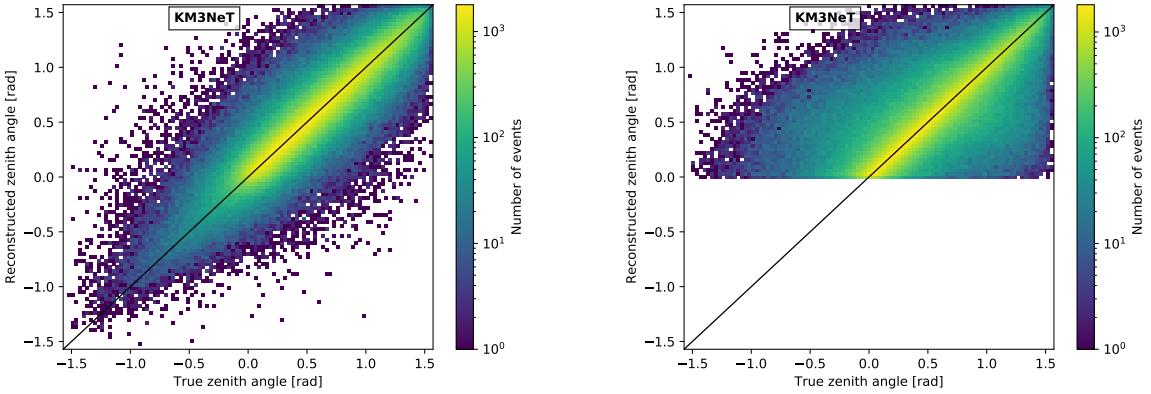

Fig. 7.20: Reconstructed zenith angle vs. true zenith angle of the CNN event regressor (left) and the TOR reconstruction (right) for ν_μ^{CC} events.

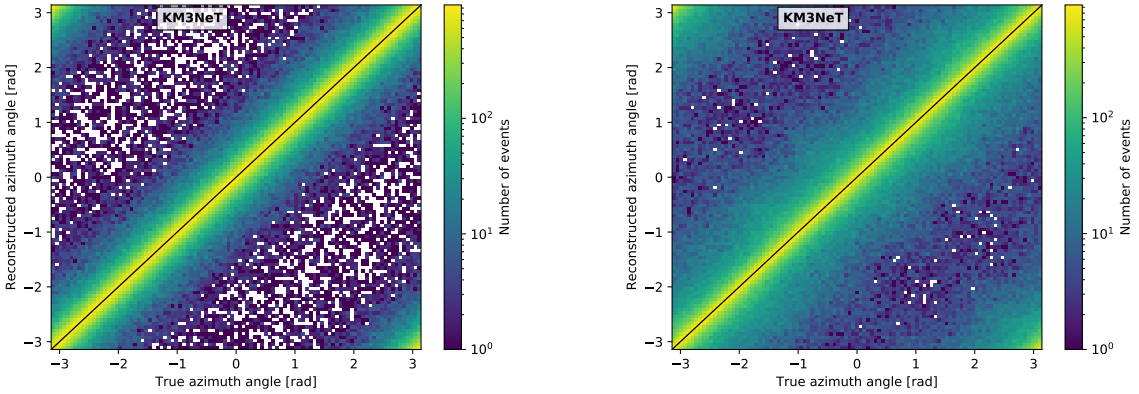

Fig. 7.21: Reconstructed azimuth angle vs. true azimuth angle of the CNN event regressor (left) and the TOR reconstruction (right) for ν_μ^{CC} events.

For a quantitative estimate of the direction reconstruction performance, another metric is introduced. The median absolute error (ME) per true energy bin is defined as the median of the distribution of the absolute residuals between the reconstructed values y_{reco} and the true MC values y_{true}:

$$\text{Median absolute error} = \text{Median}\left(|y_{\text{reco}} - y_{\text{true}}|\right) . \tag{7.12}$$

The ME of ν_μ^{CC} events for both the space angle and the zenith angle of the CNN event regressor and TOR is shown in figure 7.22 (left). The space angle is defined as the angle between the reconstructed and the true direction vector. The CNN event regressor reconstruction clearly outperforms TOR for energies below 10 GeV, while the standard reconstruction is slightly better in this metric for higher energies. A likely reason for the worse performance of the CNN event regressor for energies above 40 GeV is the strict timecut for the hits used in the image generation process. This timecut is particularly relevant for bright, high-energy ν_μ^{CC} events due to the increased fraction of late hits associated to the long muon trajectory. Deselecting these late hits likely leads to a worse direction resolution due the shortened lever arm. The ME for ν_e^{CC} events is shown in figure 7.22 (right). For energies below 3 GeV, the ME is very similar for both reconstructions, while the difference in performance increases in favor of SOR with increasing energy. The corresponding comparison for ν_e^{NC} and ν_τ^{CC} events is shown in figure 7.23. For ν_e^{NC} events, the CNN is slightly better than the SOR algorithm below energies of about 6 GeV, while the difference in performance increases again with higher energies in favor of the SOR algorithm.

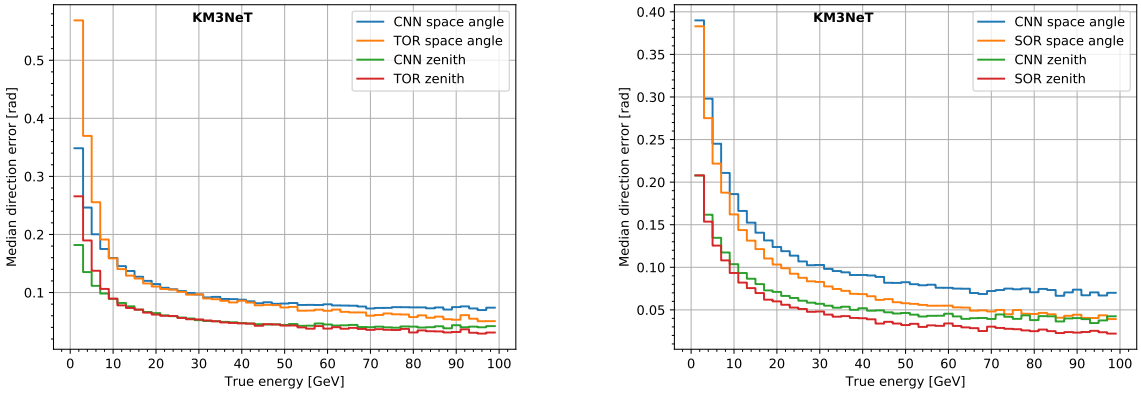

Fig. 7.22: Median absolute error in units of radians on the space angle and the zenith angle reconstruction for the CNN event regressor and the standard KM3NeT/ORCA reconstruction algorithms TOR (left) and SOR (right) versus the true MC energy for ν_μ^{CC} (left), and ν_e^{CC} (right) events.

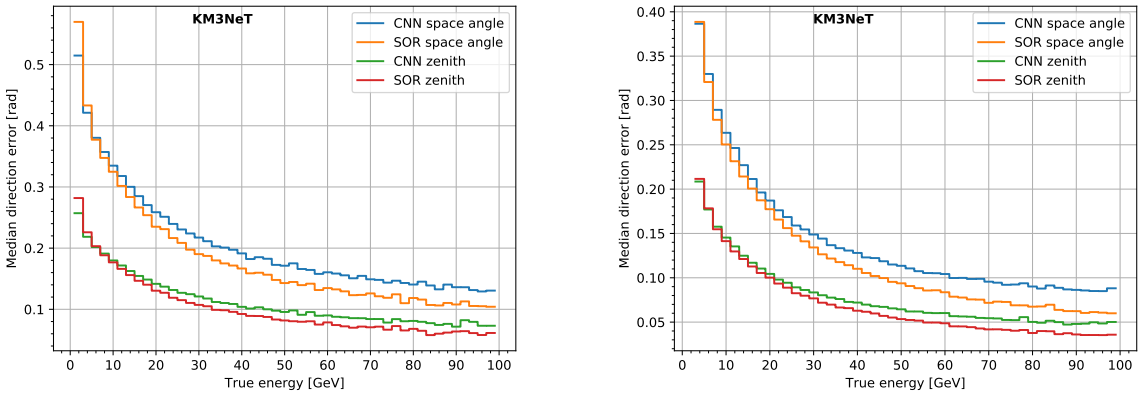

Fig. 7.23: Median absolute error in units of radians on the space angle and the zenith angle reconstruction for the CNN event regressor and the standard KM3NeT/ORCA reconstruction algorithms TOR (left) and SOR (right) versus the true MC energy for ν_e^{NC} (left) and ν_τ^{CC} (right) events.

The ME of ν_τ^{CC} events shows the same characteristics between the CNN and the SOR algorithm as for ν_e^{CC} events.

7.3.7 Vertex reconstruction performance

For the comparison of the vertex reconstruction performance the distance between the reconstructed and the true MC vertex point is investigated based on the longitudinal and the perpendicular component of the residual vector with respect to the direction of the incoming neutrino. The CNN event regressor is trained in such a way that it reconstructs the neutrino interaction vertex, while SOR and TOR reconstruct the brightest point of the Cherenkov light emission, which is shifted by an energy-dependent displacement of the order of meters. Thus, the vertex resolution of the CNN and the TOR or SOR reconstructions cannot be directly compared.

Figure 7.24 (left) shows the perpendicular versus the longitudinal distance of the reconstructed vertex with respect to the true ν_μ^{CC} interaction point for the CNN event regressor, while the right hand side of the figure shows the same distribution for TOR.

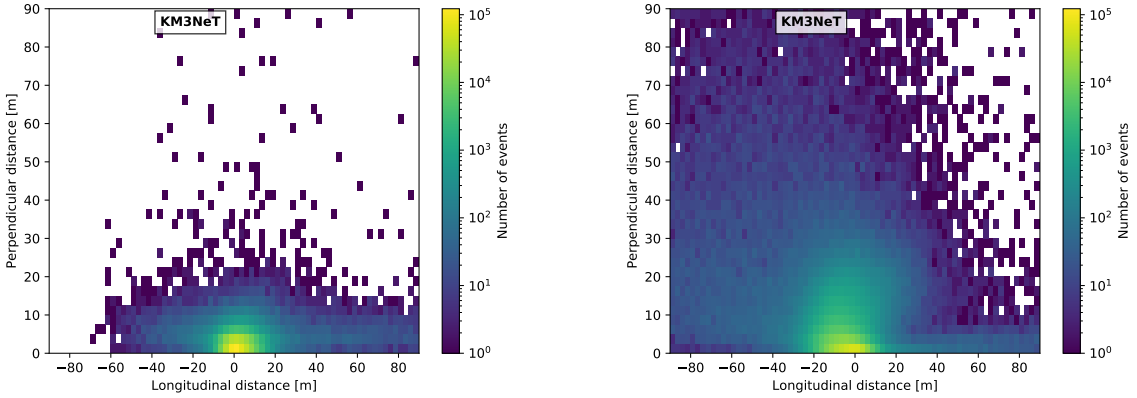

Fig. 7.24: Reconstructed neutrino interaction point with respect to the true MC vertex for ν_μ^{CC} events. Shown is the perpendicular component versus the longitudinal component of the distance vector. Left: CNN event regressor. Right: TOR.

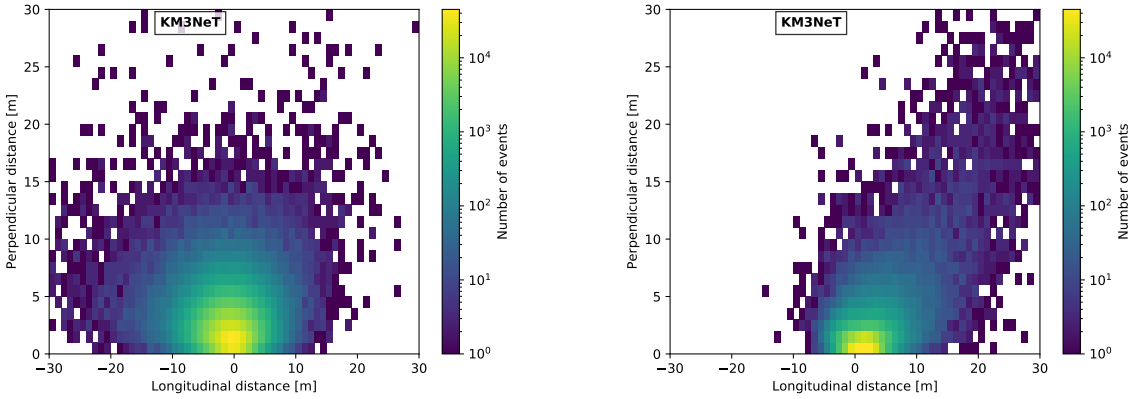

Fig. 7.25: Reconstructed neutrino interaction point with respect to the true MC vertex for ν_e^{CC} events. Shown is the perpendicular component versus the longitudinal component of the distance vector. Left: CNN event regressor. Right: SOR.

It can be seen that the vertex reconstruction error of TOR is significantly worse than that of the CNN, especially for the perpendicular component. Figure 7.25 depicts the same for ν_e^{CC} events for the CNN event regressor (left) and SOR (right). The above mentioned displaced reconstruction of the brightest emission point can be seen for SOR in the right hand part of the figure.

Additionally, the vertex resolution of the CNN compared to SOR for ν_e^{CC} events is worse. One reason for this is that each time bin in the input images of the CNN event regressor covers 12.5 ns, cf. section 7.3.1. In water, the speed of light is about $0.225\,\mathrm{m\,ns^{-1}}$ and thus this time resolution translates to about 2.8 m. Thus, this limits the vertex reconstruction of the CNN significantly. The vertex resolution for ν_e^{NC} events compared to that of ν_e^{CC} events looks very similar, cf. figure A.11. For ν_τ^{CC} events, cf. figure A.16, the SOR algorithm performs better than the CNN as well. Apart from this, the SOR vertex reconstruction error is mostly based on a positive longitudinal component, while the longitudinal error of the CNN is evenly distributed in both the positive as well as the negative direction.

For physics analyses, the vertex reconstruction is mostly used for containment cuts in order to derive a selection of well reconstructed events. For all neutrino interactions, the vertex resolution of the CNN is of the order of a few meters in the investigated energy range of up to 100 GeV, which is sufficient for this purpose.

7.3.8 Error estimation

The CNN is also used to reconstruct the uncertainty for all reconstructed quantities, as explained in section 7.3.2. For the following performance investigations, no event pre-selections have been applied, i.e. all triggered events of the simulated test dataset are used. In order to evaluate the goodness of the uncertainty reconstruction for any reconstructed event property, e.g. the energy or the direction, the events are binned according to their reconstructed uncertainty value, σ_{reco}. For the events in each bin, the standard deviation, σ_{true}, of the residuals of the reconstructed and true MC values $|y_{\text{reco}} - y_{\text{true}}|$ is calculated. Figure 7.26 (left) shows σ_{true} versus σ_{reco} for the energy reconstruction of ν_e^{CC} events.

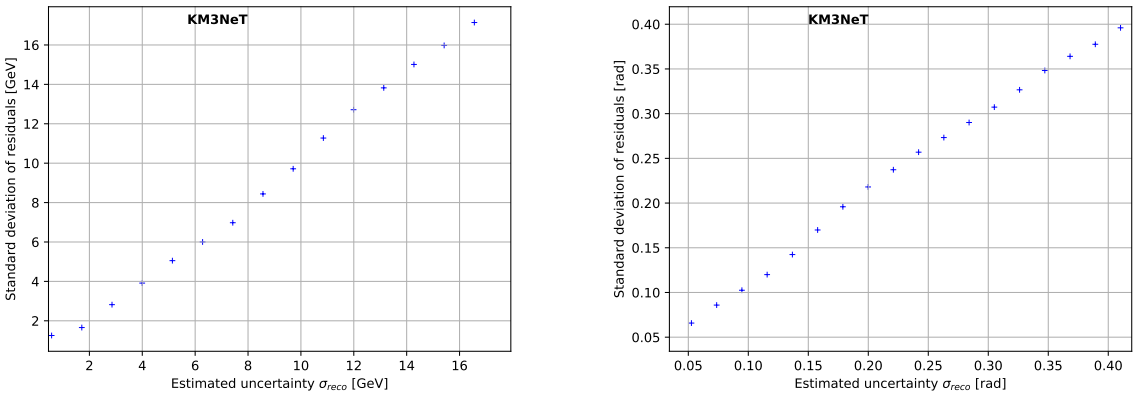

Fig. 7.26: Standard deviation of the difference of true MC and reconstructed values versus the uncertainty reconstructed by the CNN event regressor. Left: for the energy reconstruction of ν_e^{CC} events. Right: for the z-component of the reconstructed direction vector of ν_e^{CC} events.

Even though a significant fraction of events is included that has previously been discarded by the criteria discussed in section 7.3.5, e.g. non-contained events, the CNN event regressor estimation of the reconstruction uncertainty is clearly correlated to the true uncertainty. As can be seen, the energy uncertainty is slightly underestimated for events that are difficult to reconstruct and therefore show large reconstructed uncertainties. Figure 7.26 (right) shows σ_{true} versus σ_{reco} for the z-component of the reconstructed direction vector in ν_e^{CC} events. In the range of $0.15 < \sigma_{\text{reco}} < 0.30$ a slight underestimation in the uncertainty reconstruction can be seen. Figure A.4 shows the same for ν_μ^{CC} events. For the energy reconstruction, the uncertainty is generally underestimated. This is due to track-like events which are not fully contained in the detector, i.e. where the secondary μ lepton leaves the instrumented volume and thus, the light yield is not fully contained. Apart from this, the reconstructed uncertainty on the z-component of the reconstructed direction vector of ν_μ^{CC} events is sometimes slightly underestimated, e.g. where $\sigma_{\text{reco}} < 0.05$, and sometimes slightly overestimated, e.g. close to $\sigma_{\text{reco}} = 0.4$. Since there is a clear correlation between the reconstructed and the true uncertainty, it is possible to discriminate badly reconstructed events based on the reconstructed uncertainty value σ_{reco}. In order to demonstrate this, the zenith angle reconstruction for shower-like events is investigated in the following.

Figure 7.27 shows the standard deviation of the residual distribution of the reconstructed cosine of the zenith angle for ν_e^{CC} events versus the fraction of events discarded by the CNN uncertainty estimator. For all three depicted energy intervals, the zenith angle resolution improves significantly when events that have the largest reconstruction errors as predicted by the CNN are discarded.

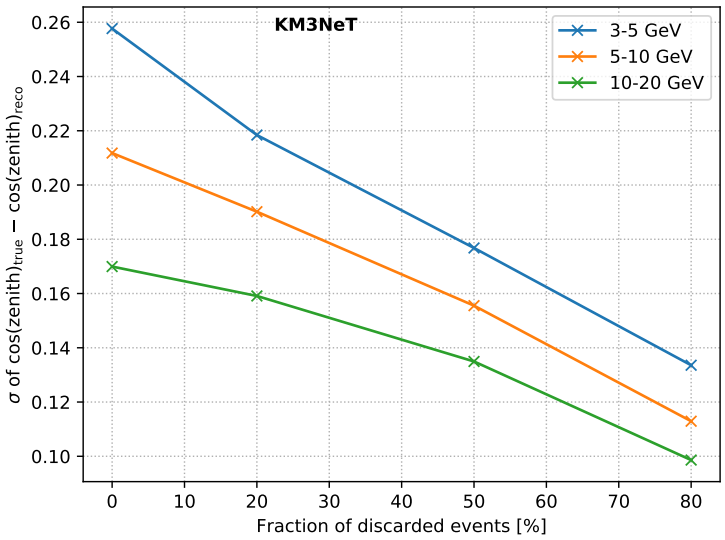

Fig. 7.27: Standard deviation of the residual distribution of the reconstructed cosine of the zenith angle for ν_e^{CC} versus the fraction of events discarded by the CNN uncertainty estimator. The investigation is split into three different neutrino energy intervals as indicated by the three colored lines.

Another method to investigate the performance of the CNN uncertainty reconstruction is the discarding significance S_i per reconstructed zenith bin i, which is defined as:

$$S_i = \frac{N_{\text{remaining},i} - (1 - F_\text{d}) \cdot N_{\text{total},i}}{\sqrt{(1 - F_\text{d}) \cdot N_{\text{total},i}}}. \tag{7.13}$$

$N_{\text{remaining},i}$ is the number of remaining events in bin i after discarding the fraction F_d of worst reconstructed events as estimated by the CNN event regressor. $N_{\text{total},i}$ is the total number of events in the $i-$th bin before the selection is applied. The defined significance S_i has positive values if less events have been discarded in a bin than would be expected by a random process deselecting a fraction of F_d events, and has negative values if correspondingly more events have been deselected by the CNN. Figure 7.28 depicts the discarding significance in bins of reconstructed zenith angle and true MC zenith angle for ν_e^{CC} events. The left (right) figure shows the significance for a fraction $F_\text{d} = 20\%$ ($F_\text{d} = 50\%$) of discarded events. The contour lines indicate the regions of the phase space for which the absolute value of the discarding significance is greater than 2σ. For both discarding fractions shown in figure 7.28 the $+2\sigma$-contour narrowly encompasses well reconstructed events, while the -2σ-contour is directly adjacent and encircles a broad off-diagonal region where badly reconstructed events have been deselected. For events in the range $-2\sigma < S_i < 2\sigma$ no significant statement can be made due to insufficient statistics.

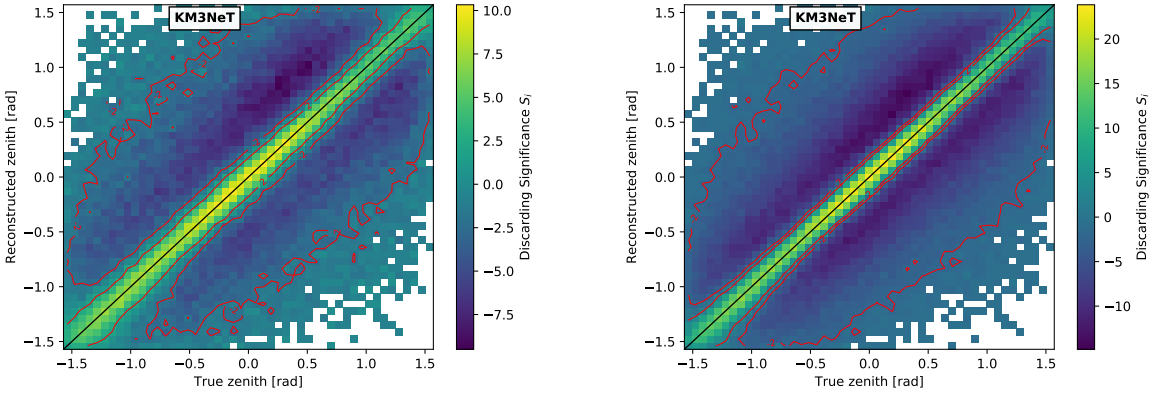

Fig. 7.28: Discarding significance S_i as a function of reconstructed zenith angle and true MC zenith angle, both given in units of radians, for shower-like ν_e^{CC} events. The contour lines indicate the regions of the phase space for which the absolute value of the discarding significance is greater than 2σ. Left: significance for an event discarding fraction of $F_d = 20\%$. Right: significance for an event discarding fraction of $F_d = 50\%$.

CHAPTER 8

Tau neutrino appearance study

Not only is the Universe stranger than we think, it is stranger than we can think.
— *Werner Heisenberg* —

In this chapter, the CNN reconstruction algorithms that have been presented in section 7 are used to study the sensitivity on the appearance of tau neutrinos for the simulated, 115 DU KM3NeT/ORCA detector. The main goal is to estimate the possible gain in sensitivity due to the CNN reconstruction compared to the standard KM3NeT/ORCA reconstruction.

For the analysis in this chapter, the collaboration software MONA [90] has been used and further modified. MONA is a RooFit-based [91] analysis framework originally created for sensitivity estimates of KM3NeT/ORCA on the neutrino mass hierarchy.

8.1 Neutrino rate calculation

As introduced in section 4.2 and shown in figure 4.4, the oscillation probabilities for up-going electron and muon neutrinos into tau neutrinos depend on the energy and the zenith angle of the neutrinos. Thus, the final measured data of KM3NeT/ORCA for a tau neutrino appearance analysis consists of observed neutrino event rates as a function of reconstructed neutrino energy and zenith angle. By comparing this data to the expected rates based on a given model, the tau neutrino normalization can be determined. The tau neutrino normalization is an energy and zenith angle independent factor that scales the total number of tau neutrino events.

For this purpose, the model needs to be able to predict the expected rate of reconstructed neutrinos in the KM3NeT/ORCA detector for a given atmospheric neutrino flux and oscillation model as a function of the neutrino energy and the zenith angle. A scheme of the necessary stages for this computation is shown in figure 8.1.

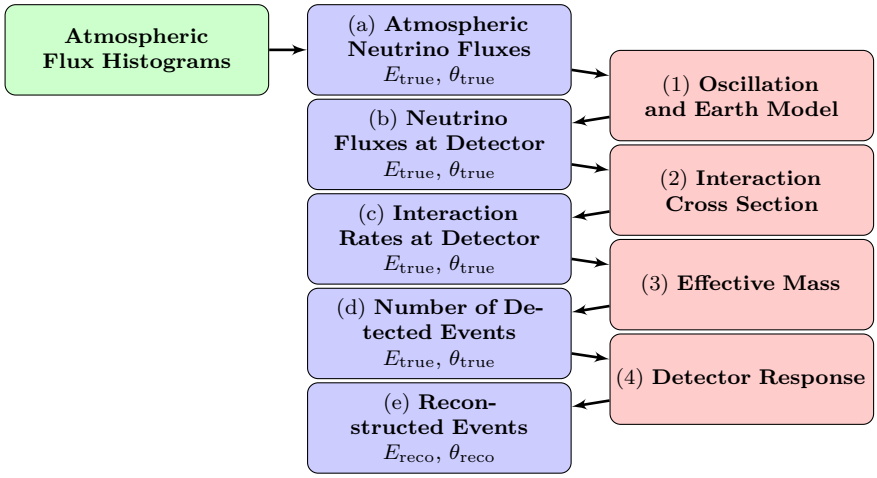

Fig. 8.1: A flowchart showing the different steps in the computation chain used for the tau neutrino appearance analysis. The blue blocks show the intermediate results as a function of the two variables written below the title. The red blocks describe the steps to go from one intermediate result to the next. Taken from [12] and modified.

The computation of the reconstructed rate consists of two principal parts, a detector-independent part and a detector-dependent part, similar to the rate calculation in the KM3NeT Letter of Intent for the sensitivity on the neutrino mass hierarchy of KM3NeT/ORCA, cf. section 3.6.1 in [12].

8.1.1 Detector-independent part

With the detector-independent part of the computation chain, the neutrino interaction rate at the KM3NeT/ORCA detector, dependent on the true neutrino energy and the true zenith angle, is derived, cf. step (c) in figure 8.1. This is done separately for each combination of neutrino flavor (e, μ, τ), neutrino particle or antiparticle type (ν, $\bar{\nu}$) and interaction type (CC, NC). A single unique combination of these properties will be further referred to as the neutrino *type*.

Modeling of the atmospheric neutrino flux

At first, the atmospheric neutrino flux as a function of E_{true} and θ_{true} is deduced, cf. step (a) in figure 8.1. For this, the HKKM2014 simulations [34] have been used. Based on these simulations, flux tables are publicly available that specify the average flux in a certain azimuth and zenith range. These flux tables are used and interpolated in MONA to derive a model for the atmospheric neutrino flux.

Oscillation and Earth model

Afterwards, in step (1) of figure 8.1, the flux is propagated to the detector by a neutrino oscillation model. For this work, the oscillation parameters are based on NuFIT 3.2 [86, 92], cf. table 8.1. Prior studies, using the standard KM3NeT/ORCA reconstruction, show that the sensitivity on the appearance of tau neutrinos is nearly independent of the neutrino mass hierarchy [93]. Thus, the currently favored mass hierarchy by NuFIT 3.2, the normal hierarchy, is assumed further on for simplicity. As introduced in section 2.2, the oscillation probabilities in matter differ from those in vacuum. Thus, the modification of the oscillation probabilities that occurs during the propagation of the neutrinos through the Earth needs to be taken into account. For this, a model for the density profile of the Earth is necessary. In this work, the PREM model [36] is used. Using this neutrino oscillation model, the neutrino flux at the detector is derived, cf. step (b) in figure 8.1.

Modeling of the neutrino interaction cross section

In order to calculate the neutrino interaction rates at the detector, step (c) in figure 8.1, the cross section of the neutrinos needs to be taken into account, shown as step (2). For this purpose, the cross sections from the GENIE [94] neutrino Monte Carlo generator, version v4r1, are used. For a tau normalization different to one, the cross section of tau neutrinos is scaled with the corresponding tau normalization factor. Then, the output of the detector-independent part of the computation chain consists of the neutrino interaction rates at the detector, dependent on the true energy and the true zenith angle.

8.1.2 Detector-dependent part

In this section, the detector-dependent part of the computation chain is introduced. This part is necessary, in order to get from the neutrino interaction rates at the detector as a function of the true energy and the true zenith angle to the rate of reconstructed neutrino events as a function of the reconstructed energy and the reconstructed zenith angle.

Effective mass

At first, it needs to be estimated how many interacting neutrinos can be reconstructed, since not all of them produce photon distributions that lead to a fired trigger. This characteristic is

described by the neutrino energy and zenith angle dependent *effective mass* parameter, cf. step (3) in figure 8.1. The effective mass is defined as follows:

$$M_{\text{eff}} \coloneqq V_{\text{gen}} \times \rho_{\text{water}} \times N_{\text{sel}}/N_{\text{gen}} \,,\; V_{\text{gen}} \to \infty \,. \tag{8.1}$$

V_{gen} is a large generation volume where the events are generated, N_{gen} is the total number of generated events and ρ_{water} is the density of seawater. N_{sel} is the number of generated events that pass some selection cuts, e.g. events that are triggered. Prior to this study, the sensitivity on the appearance of tau neutrinos had already been estimated with a toolkit different to MONA, using the standard KM3NeT/ORCA reconstruction algorithms. This has been done based on the pre-selected dataset, with the addition that the events must have been reconstructed with high confidence, cf. section 7.3.5. Thus, the same selection is employed for this study, in order to derive a fair comparison between both reconstruction methods. After the effective mass has been calculated, the number of detected events as a function of true neutrino energy and zenith angle can be derived, cf. step (d) in figure 8.1.

Detector response

In the last step, the response of the detector is modeled in order to estimate the rate of reconstructed events as a function of the reconstructed neutrino energy and zenith angle, cf. step (e) in figure 8.1. The detector response is modeled based on the set of simulations that has already been shown in table 6.1. In particular, only events from the joint test dataset of the three CNN reconstructions are used, i.e. these events have not been used for the training of the CNNs and additionally, they have also not been used for the estimation of the CNN reconstruction performance during the training.

In MONA, the detector response is modeled by the *DetResponse* class, which includes the response by the background classifier, the track-shower classifier and the reconstruction of the energy and the zenith angle. Like the prior steps in the computation chain, the response is dependent on the neutrino energy and the zenith angle. In the end, the detector response function needs to be able to map events from the true space, with E_{true} and θ_{true}, to the reconstructed space, with E_{reco} and θ_{reco}. For this purpose, two-dimensional histograms in energy and the cosine of the zenith angle are filled for the true space as well as for the reconstructed space based on the simulation data and a given neutrino type. Here, the neutrino energy is binned linearly in $\log_{10}(E)$ from 1 GeV to 100 GeV in 40 bins and the zenith angle is binned linearly in $\cos(\theta)$ from -1 to 1 in 40 bins. The binning is the same that has already been used for studies of the sensitivity on the neutrino mass hierarchy with KM3NeT/ORCA using MONA. Now, the fraction of events for bins in true space that contribute to a single bin in reconstructed space can be calculated, which is necessary for the mapping from the true space to the reconstructed space.

Till now, background events, i.e. atmospheric muons and randomly correlated noise, have been neglected. Based on the CNN background classifier, or the RF background classifiers for the standard reconstruction pipeline, the background can be reduced to a sub-percent level with a neutrino efficiency in the vicinity of 90%. For the CNN, this is done based on the reconstructed probabilities of events to be neutrinos and for the RFs, this is done by the assigned atmospheric muon and random noise score, cf. section 7.1. Due to the high rejection efficiency of background events, backgrounds can be neglected in the analysis, though the neutrino efficiency needs to be modeled in the detector response. For this, an event selection is applied to the histogram in reconstructed space, i.e. for the CNN only events that surpass a threshold value on the reconstructed neutrino probability, are added to the histogram. For the CNN background classifier, the threshold value is set to 0.9998, which results in a neutrino efficiency of 93.8%. At the same time, the atmospheric muon contamination is reduced to 0.7% and for random noise events all events in the simulated dataset can be removed. I.e. the contamination is 0%, though

the actual value cannot be inferred due to the limited statistics in the random noise dataset. For the RF background classifiers of the standard reconstruction pipeline, a selection is used that achieves approximately the same contamination levels, i.e. 0.7% for atmospheric muons and 0% for random noise events. More specifically, the atmospheric muon score and the random noise score of neutrino events needs to be below 0.02. The resulting neutrino efficiency is 85.8%, an absolute amount of 8% less compared to the neutrino efficiency of the CNN background classifier, whose performance has already been discussed in section 7.1.4.

At last, the information provided by the CNN or RF track-shower classifier is integrated. This can be done by splitting the detector response, which maps (E_{true}, θ_{true}) space to (E_{reco}, θ_{reco}) space, into two separate detector responses, i.e. one for track-like and one for shower-like events. The procedure is similar to the integration of the background classifier information. For the detector response of track-like events only events that have a reconstructed track probability of larger than 0.5 are added to the (E_{reco}, θ_{reco}) histogram in reconstructed space and vice versa. Recent studies with MONA on the sensitivity with KM3NeT/ORCA to the neutrino mass hierarchy show that the sensitivity improves if the response is split into three classes, a track class ($P^{\nu}_{\text{track}} > 0.7$), a middle class ($0.7 \geq P^{\nu}_{\text{track}} > 0.3$) and a shower class ($0.3 \geq P^{\nu}_{\text{track}}$). Thus, these three track-shower classes are also used in this study. Principally, the number of track-shower classes and the probability ranges that are assigned to each class, e.g. ($0.7 \geq P^{\nu}_{\text{track}} > 0.3$) for the middle class, are hyperparameters of the analysis that can be optimized. However, the main purpose of this study is to compare the performance of the CNN-based reconstruction to the standard KM3NeT/ORCA reconstruction and as such, no hyperparameter optimization is executed. Since the standard KM3NeT/ORCA reconstruction features separate reconstruction algorithms for track- and shower-like events, the TOR and SOR algorithm, it needs to be decided, which algorithm should be used for the three individual detector responses. In the previously mentioned neutrino mass hierarchy study, the TOR energy and zenith reconstruction is used for the track detector response, while the SOR reconstruction is used for the middle and the shower detector response. This choice has also been adapted for this study.

Based on the three modeled detector responses, the rate of detected events in the true space, step (d), can be mapped to the rate of reconstructed events in the reconstructed space, step (e). Up to this point, the computation chain shown in figure 8.1 has been executed separately for the different neutrino types. For the final output of the computation chain, the rates of the individual neutrino types are combined. Thus, the reconstructed neutrino rates, which are a function of the reconstructed neutrino energy and zenith angle, consist of three sets of data, one for the track class, one for the middle class and one for the shower class.

8.2 Tau appearance sensitivity calculation

The goal of this study is to estimate by which significance a tau normalization different to 1 can be rejected for a given period of data taking. For a single $\tau_{\text{norm}} \neq 1$, this is a binary hypothesis testing problem, where one hypothesis is $\tau_{\text{norm}} \neq 1$ and the other one is $\tau_{\text{norm}} = 1$. This is similar to neutrino mass hierarchy studies, where one hypothesis is the normal hierarchy and the other one is the inverted hierarchy.

For both the tau neutrino appearance as well as for the neutrino mass hierarchy study, the sensitivity can be estimated by using likelihood ratio distributions from pseudo-experiments. More details about this technique can be found in section 3.6.1 of [12] and in chapter 6 of [88]. The drawback with this technique is that the generation of the pseudo-experiments for given oscillation and systematic parameters, based on the computation chain introduced in section 8.1, is computationally expensive. For the neutrino mass hierarchy analysis, several thousands of pseudo-experiments are needed in order to achieve high enough statistics [88].

For the tau neutrino analysis, the situation is even worse, since we would like to estimate the sensitivity for many $\tau_{\text{norm}} \neq 1$ values, and not just for a single one, i.e. many binary hypotheses with different $\tau_{\text{norm}} \neq 1$ must be tested. An alternative technique, which resolves this problem, is to use a so-called Asimov-dataset approach instead of the pseudo-experiments. Here, the sensitivity is calculated based on the expected dataset for a fixed set of given oscillation and systematic parameters. In other words, this "average", Asimov dataset can be fit exactly with the set of parameters that has been used to generate the Asimov dataset. In the following, a step-by-step explanation is presented that introduces, with which significance a single tau normalization $\tau_{\text{norm}} \neq 1$ can be excluded for a given period of data taking based on the Asimov dataset approach.

1. For each track-shower class, wrap the computation chain described in section 8.1 into a MONA FitPDF instance, which is based on RooFit probability density functions.

2. Using the PDFs, generate three sets of (E_{reco}, $\cos(\theta_{\text{reco}})$) histograms for the given period of data taking, one for each track-shower class. The oscillation parameters in the PDFs are set to the best-fit NuFIT 3.2 oscillation parameters and the tau normalization is initialized with $\tau_{\text{norm}} = 1$. These histograms represent the Asimov dataset, which constitutes the most representative experiment. The neutrino energy is binned linearly in $\log_{10}(E)$ from 1 GeV to 100 GeV in 40 bins and the zenith angle is binned linearly in $\cos(\theta)$ from -1 to 1 in 40 bins. Wrap the histograms in RooFit RooDataHists.

3. Fit the Asimov dataset ($\tau_{\text{norm}} = 1$), consisting of the three (E_{reco}, $\cos(\theta_{\text{reco}})$) histograms, with the PDFs in a binned maximum likelihood fit. The fitting is done in the up-going region only, i.e. with $\cos(\theta)$ from -1 to 0. For each iteration in the fitting procedure, the three track-shower classes are fitted simultaneously. For this purpose, the single PDFs are combined into a RooFit RooSimultaneous PDF. In the simultaneous fit, the likelihood during each iteration is calculated separately for each track-shower class. Then, the likelihood for the three cases is added together and the sum of the negative log-likelihoods is minimized. Note down the minimum of the negative log-likelihood, $\min(-\ln \mathcal{L})_{\tau_{\text{norm}}=1}$.

4. Fit the Asimov dataset from step 2. with the PDFs as in step 3., but set the tau normalization in the PDFs to a constant value, where $\tau_{\text{norm}} \neq 1$. Note down the minimum of the negative log-likelihood, $\min(-\ln \mathcal{L})_{\tau_{\text{norm}}\neq 1}$.

5. The significance on the exclusion of a tau normalization with $\tau_{\text{norm}} \neq 1$ for the given period of data taking can be calculated as $\sigma = \sqrt{2 \cdot \left(\min(-\ln \mathcal{L})_{\tau_{\text{norm}}\neq 1} - \min(-\ln \mathcal{L})_{\tau_{\text{norm}}=1}\right)}$.

For the simplified case of one-dimensional, Poisson distributed data, the likelihood in the binned maximum likelihood fit is calculated as follows:

$$\chi^2_{\text{P}} = -2 \ln \mathcal{L}_{\text{stat}} = \sum_{i=1}^{I} \left(N_{\text{model},i} - N_{\text{obs},i} + N_{\text{obs},i} \ln\left(\frac{N_{\text{obs},i}}{N_{\text{model},i}} \right) \right) , \tag{8.2}$$

where I is the total number of bins, $N_{\text{model},i}$ is the number of events in the i-th bin based on the model from step 1., $N_{\text{obs},i}$ is the number of events in the i-th bin of the Asimov dataset, $\mathcal{L}_{\text{stat}}$ is the likelihood and χ^2_{P} is the chi-square for Poisson distributed data. In addition, systematic parameters with Gaussian-like prior knowledge, denoted as η_p, are included in the fit [88]:

$$\mathcal{L} = \mathcal{L}_{\text{stat}} \times \mathcal{L}_{\text{syst}} , \tag{8.3}$$

$$\mathcal{L}_{\text{syst}} = \prod_{p=1}^{P} \frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\left(\frac{\eta_p - \mu_p}{2\sigma_p^2} \right) . \tag{8.4}$$

Here, P is the total number of systematic parameters with prior knowledge and μ_p and σ_p^2 is the mean and the squared standard deviation of the Gaussian prior distribution for the systematic parameter p. The fitting procedure itself is performed with the MUNUIT2 package, contained in the ROOT [95] toolkit. The parameters of the fit are summarized in table 8.1.

parameter	initial value	prior	treatment
$\sin^2\theta_{12}$	0.307	-	fixed
$\sin^2\theta_{13}$	0.02206	-	fitted
$\sin^2\theta_{23}$	0.538	-	fitted
δ_{CP} [$\circ$]	1.3	-	fitted
$\frac{\Delta m^2_{21}}{1\times10^{-5}\,\mathrm{eV}^2}$	7.40	-	fixed
$\frac{\Delta m^2_{31}}{1\times10^{-3}\,\mathrm{eV}^2}$	2.494	-	fitted
NC scaling	1	$\mu = 1\ ,\ \sigma = 0.1$	fitted
E-tilt	0	-	fitted
$\cos(\theta)$-tilt	0	-	fitted
$\nu_\mu/\bar{\nu}_\mu$ skew	0	$\mu = 0\ ,\ \sigma = 0.1$	fitted
$\nu_e/\bar{\nu}_e$ skew	0	$\mu = 0\ ,\ \sigma = 0.1$	fitted
ν_μ/ν_e skew	0	$\mu = 0\ ,\ \sigma = 0.1$	fitted

Table 8.1: Oscillation parameters, from NuFIT 3.2 [86, 92], and systematic parameters used in the fit for the tau neutrino appearance analysis. The initial values in the second column are used as starting parameters, when the minimizer starts the fit.

For the starting values of the fit, the normal hierarchy is assumed, due to the reasons described in section 8.1.1. It can be observed that the fitter never changes the octant, i.e. the best-fit solution of θ_{23} for all $\tau_{\mathrm{norm}} \neq 1$ fits is always contained in the second octant. The solar parameters θ_{12} and Δm^2_{21} have been fixed, since they are well constrained by external experiments. For the systematic parameters with Gaussian-like prior knowledge, the values of the mean and the standard deviation of the prior distribution are based on the already mentioned KM3NeT/ORCA neutrino mass hierarchy studies. The NC scaling parameter is an energy and zenith angle independent factor that scales the number of neutral current events, similar to the tau normalization. The E-tilt and the $\cos(\theta)$-tilt parameters are related to the atmospheric neutrino flux. For the energy, the flux is multiplied by a factor of $E^{E\text{-tilt}}$ and for the zenith the flux is multiplied by a factor of $\cos(\theta)$-tilt $\times \cos(\theta)$. The $\nu_\mu/\bar{\nu}_\mu$ skew parameter skews the ratio of the muon neutrino flux compared to the anti-muon neutrino flux, while the total number of muon neutrinos is conserved. The same applies to the $\nu_e/\bar{\nu}_e$ and the ν_μ/ν_e skew parameters.

8.3 Results and comparison with standard ORCA reconstruction

In this section, the sensitivity on the tau neutrino appearance of the CNN-based reconstruction is investigated and compared to the sensitivity of the standard KM3NeT/ORCA reconstruction. To this end, the sensitivity calculation described in section 8.2 is executed for tau normalizations in the range of $[0, 2]$ with a step size of 0.0125 and for detector lifetimes of one to twelve months with a step size of one month. Figure 8.2 shows the KM3NeT/ORCA rejection significance of a tau normalization different to one after one year of data taking for the deep learning-based reconstruction chain and the standard KM3NeT/ORCA reconstruction chain.

It can be seen that for both reconstruction chains, a tau normalization smaller than 0.8 and larger than 1.2 can be excluded after one year of data taking with a significance of about $2.5\,\sigma$.

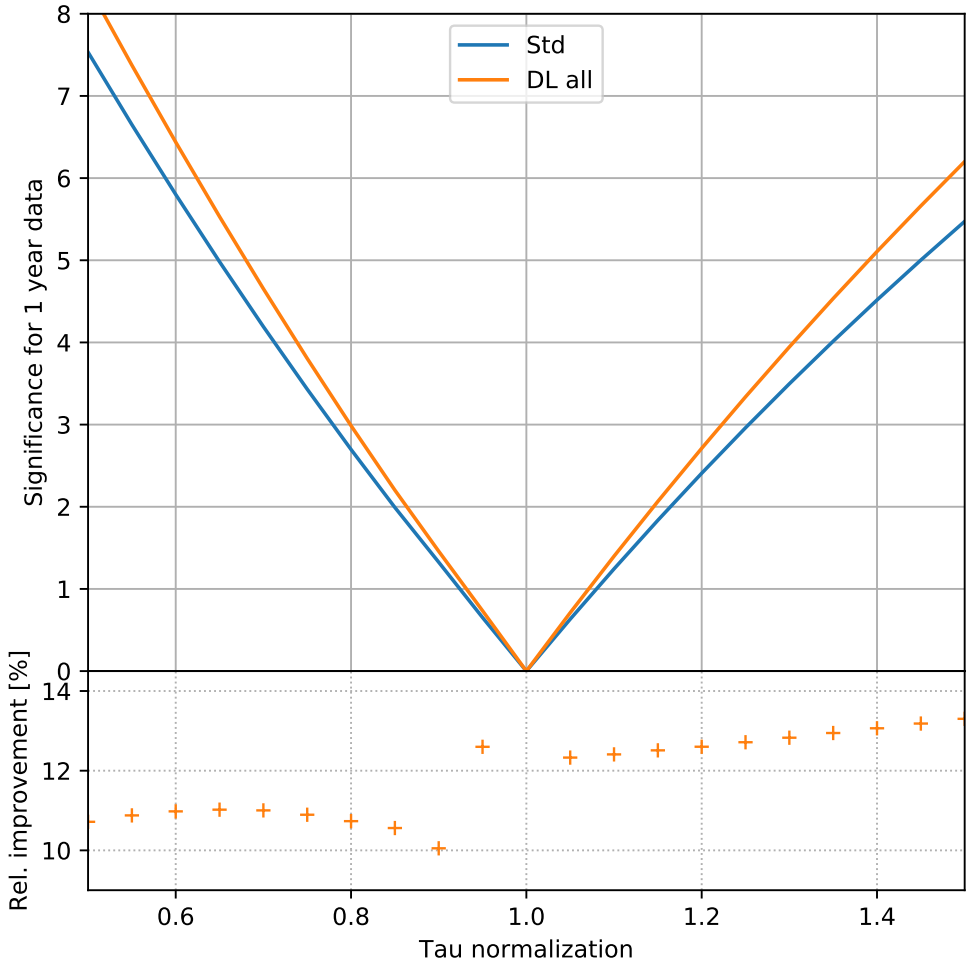

Fig. 8.2: Top panel: sensitivity on the rejection of tau normalization values different to one after one year of data taking for the deep learning-based, CNN reconstruction chain ("DL all", orange) and for the standard KM3NeT/ORCA reconstruction chain ("Std", blue). Bottom panel: relative improvement in sensitivity of the CNN-based reconstruction compared to the standard KM3NeT/ORCA reconstruction.

The relative gain in sensitivity of the CNN-based reconstruction compared to the standard KM3NeT/ORCA reconstruction ranges from about 10% to about 13%, depending on the rejected tau normalization value. This leads to the question, to which reconstructed value of the reconstruction chain the significance on the tau neutrino appearance is most sensitive to. This has previously been studied in [93] for the standard KM3NeT/ORCA reconstruction, where it could be shown that a perfect event regression, i.e. a perfect energy and zenith angle reconstruction, leads to a significantly higher gain in sensitivity than for a perfect track-shower classification. However, this study has only been done for two track-shower classes. Thus, it is not clear if this circumstance also applies if additional track-shower classes are introduced. For this work, the study from figure 8.2 has been repeated in such a way, that only a part of the deep learning reconstruction is employed for the calculation of the detector response, i.e. an admixture of the standard reconstruction and the CNN-based reconstruction is used. Hereby it can be determined, which part of the CNN-based reconstruction chain leads to the highest improvement in sensitivity. The result of this study is shown in figure 8.3. It can be seen that the direction reconstruction, followed by the background classification and the track-shower classification, shows the smallest effect on the sensitivity. For the background classification, it has been estimated that the neutrino

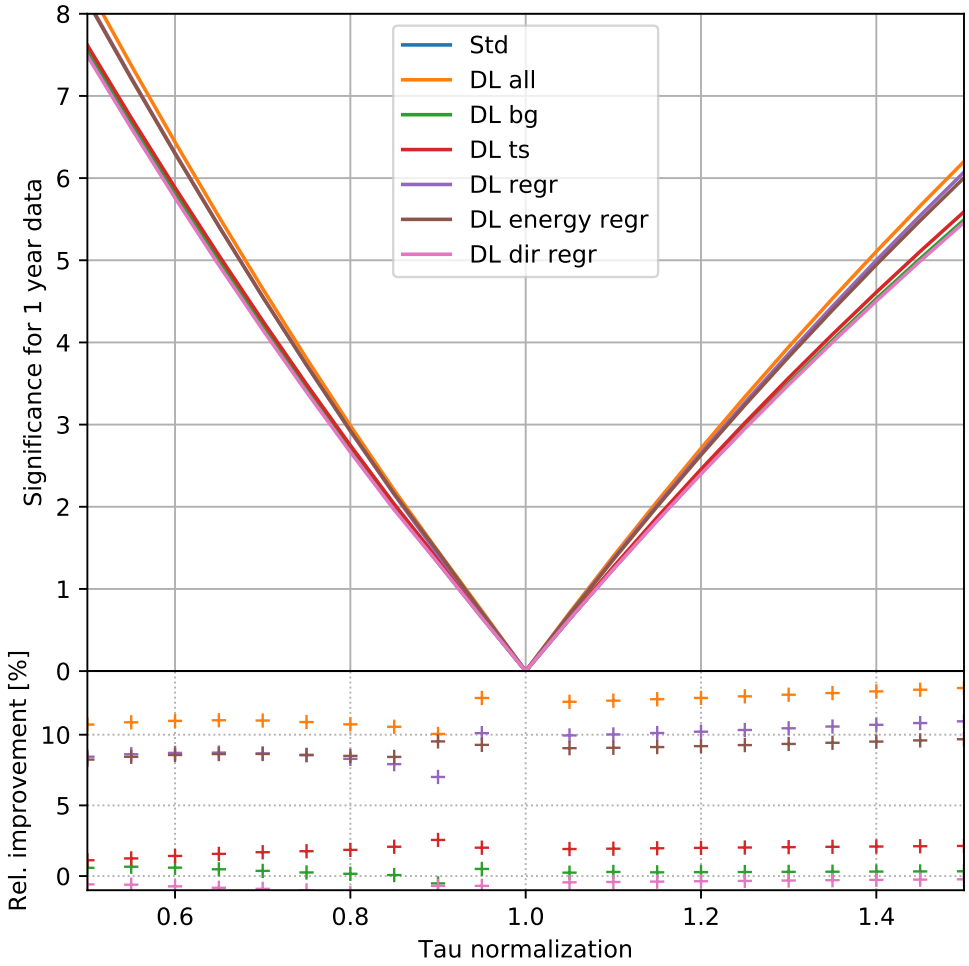

Fig. 8.3: Top panel: sensitivity on the rejection of tau normalization values different to one after one year of data taking for the standard KM3NeT/ORCA reconstruction chain ("Std", blue) and for different admixtures of the standard and the CNN-based reconstruction (other colors). For example, the term "DL bg" specifies, that the deep learning-based background classifier has been employed for the reconstruction, while the other reconstruction tasks are handled by the standard KM3NeT/ORCA reconstruction. Bottom panel: relative improvement in sensitivity to the standard KM3NeT/ORCA reconstruction ("Std", blue) for each of the other curves in the top panel.

efficiency can be improved by an absolute amount of 8%, cf. section 8.1.2. However, the neutrinos that contribute to the gain in neutrino efficiency have only energies of a few GeV. And if the energy of the tau neutrino is smaller than the energy threshold for the production of the tau lepton, about 3.5 GeV, a charged current tau neutrino interaction is not possible. Thus, an improved reconstruction for these kind of events has no effect on the tau appearance sensitivity. The largest gain in sensitivity is made by the energy reconstruction, which must be attributed to the energy reconstruction of track-like, ν_μ^{CC} events, since there is no significant improvement on the energy reconstruction for shower-like events. The sensitivity gain only due to the energy reconstruction compared to the track-shower classification is significantly larger, agreeing with the results in [93]. Till now, the sensitivity on the appearance of tau neutrinos has only been investigated for one year of data taking. Figure 8.4 shows the rejection significance of a tau normalization different to one dependent on the data taking period of the detector for the CNN-based reconstruction and for the standard reconstruction chain.

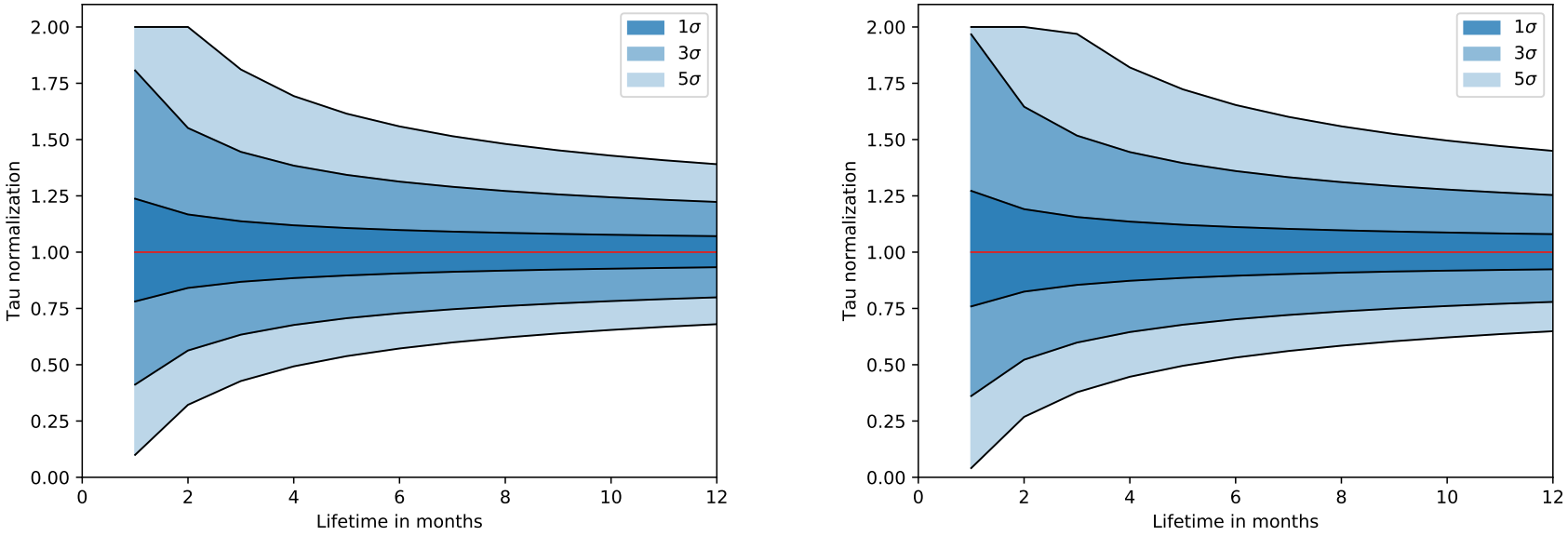

Fig. 8.4: Exclusion bands on the tau normalization dependent on the detector lifetime for the CNN-based reconstruction chain (left) and the standard ORCA reconstruction (right). It can be seen that the CNN-based reconstruction shows an improvement in sensitivity.

It can be seen that the exclusion bands for the CNN-based reconstruction are tighter than those of the standard reconstruction over all the different detector lifetimes.

Studies based on the standard KM3NeT/ORCA reconstruction [93] show that the sensitivity can be improved with a looser event selection. Thus, the sensitivity has also been estimated for the full dataset, with no event selection. Compared to the previously used, pre-selected dataset, about 50% more neutrinos are contained in the full dataset. However, since their light signature is not fully contained in the detector, these events generally cannot be reconstructed well, potentially leading to additional systematic effects. For example, the neutrino interaction can take place outside of the detector, such that the secondaries of a high-energy neutrino may only deposit few hits in the detector. Figure 8.5 shows the rejection significance of a tau normalization different to one after one year of data taking for the deep learning-based reconstruction chain based on the full, non-selected dataset compared to the sensitivity already shown in figure 8.2.

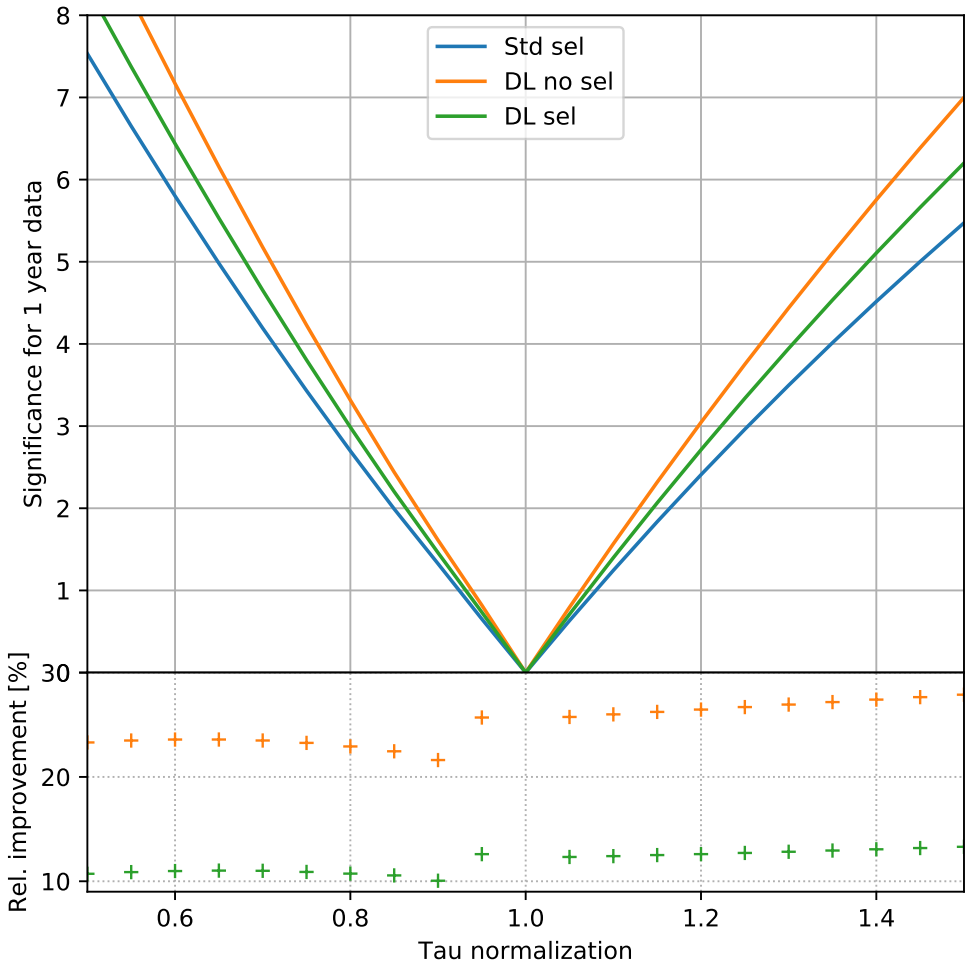

Fig. 8.5: Top panel: sensitivity on the rejection of tau normalization values different to one after one year of data taking for the CNN-based reconstruction chain with no event selection ("DL no sel", orange) compared to the CNN-based / standard reconstruction with an event selection ("DL sel", green / "Std sel", blue). The dataset with no event selection contains about 50% more neutrino events. Bottom panel: relative improvement in sensitivity of the CNN-based reconstruction with or without an event selection compared to the standard KM3NeT/ORCA reconstruction with an event selection.

It can be seen that there is an additional absolute improvement in sensitivity of more than 10% compared to the CNN-based reconstruction with used event selection. However, one reason for the typically used event selection, which, among other criteria, is based on fully contained events, is that these events are well understood. As already mentioned, for non-contained events, one needs to make sure that there are no systematic effects that arise when the analysis is applied to real data. Figure 8.6 shows the rejection significance of a tau normalization different to one as a function of the data taking period of the detector for the CNN-based reconstruction without any event selection. Comparing this figure to figure 8.4, shows a significant improvement when no event selection is used for several different data taking periods.

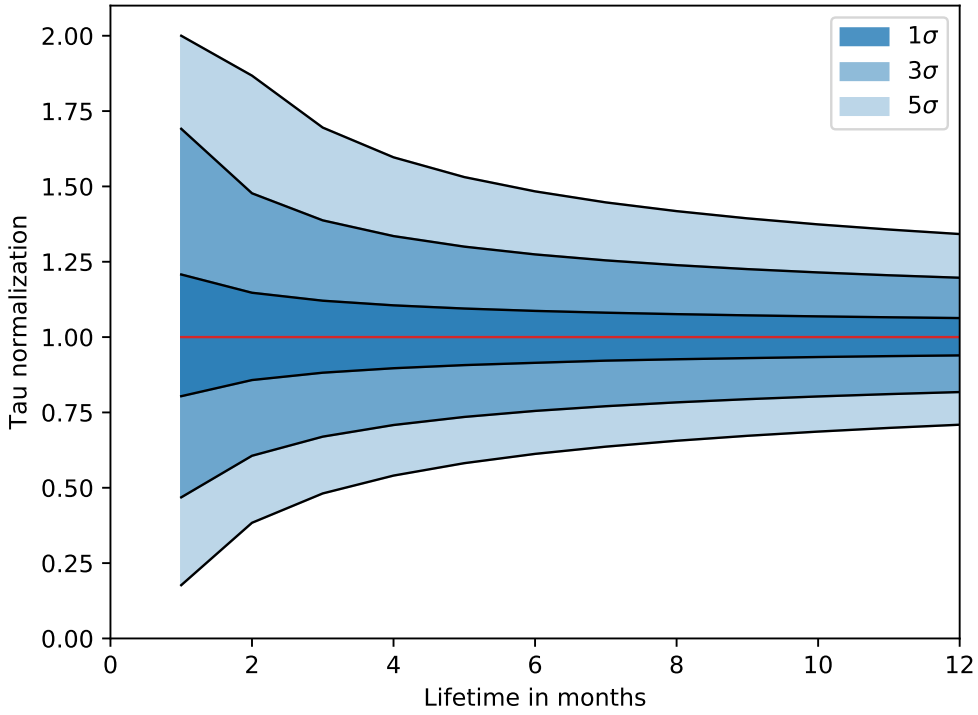

Fig. 8.6: Exclusion bands on the tau normalization after a certain amount of detector lifetime for the CNN-based reconstruction chain without any event selection.

CHAPTER 9

Summary and Outlook

The important thing is not to stop questioning.

— Albert Einstein —

Within this decade, significant progress has been made in the machine learning community due to the advent of deep learning techniques. In the last few years, these techniques have been more and more adapted by (astro-) particle physics experiments [96, 97, 98, 99, 100]. These studies show that the emergence of deep learning algorithms in (astro-) particle physics shows a huge potential.

In this work, the first application of deep convolutional neural networks to the reconstruction of simulated neutrino events of the neutrino telescope KM3NeT/ORCA has been reported. Two different CNNs have been employed to separate neutrino events from background and track-like from shower-like neutrino events. In addition, another CNN has been designed to reconstruct the direction, the energy, and the interaction point of the neutrinos. Furthermore, the uncertainty on each of these reconstructed quantities is estimated by the CNN as well. As a result, all tasks of the event reconstruction and classification pipeline for KM3NeT/ORCA have been tackled, such that a full, deep learning-based reconstruction pipeline is now available as an alternative to the standard KM3NeT/ORCA reconstruction chain. Based on simulations for the 115 DU, KM3NeT/ORCA detector it has been demonstrated that the application of the CNNs exhibits promising, and in many cases already better performances than the standard KM3NeT/ORCA reconstruction methods. At the same time, only a minor amount of optimization has been executed for the CNNs, such that further improvements can be expected.

In the future, it needs to be investigated how the presented CNNs can be reliably applied to real data. For example, the effect of missing optical modules in the detector or the movement of the detection units needs to be taken into account. In addition, since the CNN-based methods may potentially be sensitive to other systematics of the simulation compared to the standard KM3NeT/ORCA algorithms, the agreement between the reconstructed CNN outputs on simulations and data needs to be investigated.

As presented in the last chapter of this work, the CNN-based reconstruction has been applied to the sensitivity on the appearance of tau neutrinos for the KM3NeT/ORCA detector. It shows that a significant gain in sensitivity can be achieved compared to an analysis based on the standard KM3NeT/ORCA reconstruction algorithms. In general though, the actual improvement of the sensitivity depends on the kind of physics analysis that is carried out. The reason is that different kinds of analyses are sensitive to different parts of the reconstructed data. For example, for one analysis, the track-shower classification may have a huge impact on the sensitivity, while for another analysis it may not. Preliminary results on the application of the CNN-based track-shower reconstruction for the sensitivity of KM3NeT/ORCA on the neutrino mass hierarchy show that the relative gain in sensitivity is already larger than for the tau neutrino appearance study, which uses the whole, deep learning-based reconstruction. Thus, it will be interesting to see in the future, how large the gain in sensitivity is if the whole CNN-based reconstruction is used.

Bibliography

[1] W. Pauli. *Offener Brief an die Gruppe der Radioaktiven bei der Gauvereinstagung zu Tübingen.* 1930. URL: http://inspirehep.net/record/45177/files/meitner_0393.pdf?version=1%20%20December%201930.

[2] M. Aker et al. "An improved upper limit on the neutrino mass from a direct kinematic method by KATRIN." In: *Phys. Rev. Lett. 123, 221802 (2019)* (Sept. 13, 2019). DOI: 10.1103/PhysRevLett.123.221802. arXiv: 1909.06048v1.

[3] K. A. Olive et al. "Review of Particle Physics." In: *Chin. Phys.* C38 (2014), p. 090001. DOI: 10.1088/1674-1137/38/9/090001.

[4] B. Pontecorvo. "Neutrino Experiments and the Problem of Conservation of Leptonic Charge." In: *Soviet Journal of Experimental and Theoretical Physics* 26 (May 1968), p. 984.

[5] Q. R. Ahmad et al. "Measurement of the Rate of ve+d -> p+p+e Interactions Produced by B8 Solar Neutrinos at the Sudbury Neutrino Observatory." In: *Physical Review Letters* 87.7 (July 2001). DOI: 10.1103/physrevlett.87.071301. URL: http://dx.doi.org/10.1103/PhysRevLett.87.071301.

[6] S. M. Bilenky and C. Giunti. "Neutrinoless double-beta decay. A brief review." In: *Mod. Phys. Lett. A 27* (Mar. 2012). DOI: 10.1142/S0217732312300157.

[7] M. Kobayashi and T. Maskawa. "CP-Violation in the Renormalizable Theory of Weak Interaction." In: *Progress of Theoretical Physics* 49.2 (Feb. 1973), pp. 652–657. DOI: 10.1143/PTP.49.652.

[8] S. M. Bilenky and C. Giunti. "Neutrinoless Double-Beta Decay: a Probe of Physics Beyond the Standard Model." In: *International Journal of Modern Physics A 30, 1530001 (2015)* (Nov. 2014). DOI: 10.1142/S0217751X1530001X.

[9] M. Wurm. *Neutrino mass hierarchy.* URL: http://www.staff.uni-mainz.de/wurmm/juno.html (visited on 06/30/2016).

[10] S. P. Mikheyev and A. Y. Smirnov. "Resonance enhancement of oscillations in matter and solar neutrino spectroscopy." In: *Yadernaya Fizika* 42 (1985), pp. 1441–1448.

[11] L. Wolfenstein. "Neutrino oscillations in matter." In: *Phys. Rev. D* 17 (9 May 1978), pp. 2369–2374. DOI: 10.1103/PhysRevD.17.2369. URL: https://link.aps.org/doi/10.1103/PhysRevD.17.2369.

[12] S. Adrián-Martínez et al. "Letter of Intent for KM3NeT2.0." In: (Jan. 2016). arXiv: 1601.07459.

[13] I. 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: *JHEP 01 (2019) 106* (Nov. 13, 2018). DOI: 10.1007/JHEP01(2019)106. arXiv: 1811.05487v1.

[14] *NuFIT 4.1 (2019).* Dec. 12, 2019. URL: www.nu-fit.org.

[15] S. Bilenky. *Introduction to the Physics of Massive and Mixed Neutrinos.* Lecture notes in Physics 817. Springer, 2010.

[16] M. Tanabashi et al. "Review of Particle Physics, chapter 13: CP Violation in the Quark Sector." In: *Phys. Rev. D* 98 (3 Aug. 2018), p. 030001. DOI: 10.1103/PhysRevD.98.030001. URL: https://link.aps.org/doi/10.1103/PhysRevD.98.030001.

[17] C. Giganti, S. Lavignac, and M. Zito. "Neutrino oscillations: the rise of the PMNS paradigm." In: (Oct. 2, 2017). DOI: 10.1016/j.ppnp.2017.10.001. arXiv: 1710.00715v2.

[18] M. D. Schwartz. *Quantum Field Theory and the Standard Model*. Cambridge University Press, 2014. ISBN: 1107034736. URL: http://www.cambridge.org/us/academic/subjects/physics/theoretical-physics-and-mathematical-physics/quantum-field-theory-and-standard-model.

[19] A. Pich. "The Standard Model of Electroweak Interactions." In: (Feb. 1, 2005). arXiv: hep-ph/0502010v1.

[20] K. McFarland. "Neutrino Interactions." In: (Apr. 24, 2008). arXiv: 0804.3899v1.

[21] J. A. Formaggio and G. P. Zeller. "From eV to EeV: Neutrino Cross Sections Across Energy Scales." In: *Rev. Mod. Phys.* 84 (2012), pp. 1307–1341. DOI: 10.1103/RevModPhys.84.1307. arXiv: 1305.7513.

[22] J. Hofestädt. "Measuring the neutrino mass hierarchy with the future KM3NeT/ORCA detector." PhD thesis. Friedrich-Alexander University Erlangen-Nürnberg, 2017.

[23] D. Casper. "The nuance neutrino physics simulation, and the future." In: *Nuclear Physics B - Proceedings Supplements* 112.1 (2002), pp. 161–170. ISSN: 0920-5632. DOI: https://doi.org/10.1016/S0920-5632(02)01756-5. URL: http://www.sciencedirect.com/science/article/pii/S0920563202017565.

[24] Institute for Cosmic Ray Research, University of Tokyo. *About Kamioka Observatory*. URL: http://www-sk.icrr.u-tokyo.ac.jp/aboutus/index-e.html (visited on 06/30/2016).

[25] S. Umehara et al. "Search for Neutrino-less Double Beta Decay with CANDLES." In: *Phys. Procedia* 61 (2015), pp. 283–288. DOI: 10.1016/j.phpro.2014.12.046.

[26] H. Kakubata. "Study of Backgrounds in CANDLES to Search for Double Beta Decays of 48Ca." PhD thesis. Osaka University, Graduate School of Science, 2015.

[27] Z. Li et al. "A Measurement of the Tau Neutrino Cross Section in Atmospheric Neutrino Oscillations with Super-Kamiokande." In: (2017). arXiv: 1711.09436.

[28] M. G. Aartsen et al. "PINGU: a vision for neutrino and particle physics at the South Pole." In: *Journal of Physics G: Nuclear and Particle Physics* 44.5 (2017), p. 054006. URL: http://stacks.iop.org/0954-3899/44/i=5/a=054006.

[29] N. Agafonova et al. "Discovery of τ Neutrino Appearance in the CNGS Neutrino Beam with the OPERA Experiment." In: *Phys. Rev. Lett.* 115.12 (2015), p. 121802. DOI: 10.1103/PhysRevLett.115.121802. arXiv: 1507.01417.

[30] N. Agafonova et al. "Observation of tau neutrino appearance in the CNGS beam with the OPERA experiment." In: *PTEP* 2014.10 (2014), p. 101C01. DOI: 10.1093/ptep/ptu132. arXiv: 1407.3513.

[31] M. G. Aartsen et al. "Evidence for High-Energy Extraterrestrial Neutrinos at the IceCube Detector." In: *Science 342, 1242856 (2013)* (Nov. 20, 2013). DOI: 10.1126/science.1242856. arXiv: http://arxiv.org/abs/1311.5238v2.

[32] U. F. Katz and C. Spiering. "High-Energy Neutrino Astrophysics: Status and Perspectives." In: (Nov. 2, 2011). DOI: 10.1016/j.ppnp.2011.12.001. arXiv: 1111.0507v1.

[33] C. Patrignani and P. D. Group. "Review of Particle Physics." In: *Chinese Physics C* 40.10 (2016), p. 100001. URL: http://stacks.iop.org/1674-1137/40/i=10/a=100001.

[34] M. Honda et al. "Atmospheric neutrino flux calculation using the NRLMSISE-00 atmospheric model." In: *Phys. Rev. D* 92 (2 July 2015), p. 023004. DOI: 10.1103/PhysRevD.92.023004. URL: https://link.aps.org/doi/10.1103/PhysRevD.92.023004.

[35] J. P. Yañez and A. Kouchner. "Measurement of atmospheric neutrino oscillations with very large volume neutrino telescopes." In: *Advances in High Energy Physics Volume 2015 (2015)* (Sept. 28, 2015). DOI: 10.1155/2015/271968. arXiv: 1509.08404v2 [hep-ex].

[36] A. M. Dziewonski and D. L. Anderson. "Preliminary reference Earth model." In: *Physics of the Earth and Planetary Interiors* 25.4 (1981), pp. 297–356. ISSN: 0031-9201. DOI: https://doi.org/10.1016/0031-9201(81)90046-7. URL: http://www.sciencedirect.com/science/article/pii/0031920181900467.

[37] S. Hallmann. "Tau neutrinos through oscillation in the atmospheric neutrino flux with KM3NeT/ORCA and Search for a neutrino signal from the "Fermi Bubbles" with ANTARES." PhD thesis. Friedrich-Alexander University Erlangen-Nürnberg, 2020.

[38] T. Gaisser and A. Karle. *Neutrino Astronomy: current status, future prospects*. World Scientific, 2017. DOI: 10.1142/9964. URL: https://www.worldscientific.com/doi/abs/10.1142/9964.

[39] J. Tiffenberg. *UHE Neutrino searches with the Pierre AugerObservatory*. 2011. URL: http://users.ictp.it/~smr2246/monday/tiffenberg-NUSKY.pdf.

[40] Y.-S. Tsai. "Pair production and bremsstrahlung of charged leptons." In: *Rev. Mod. Phys.* 46 (4 Oct. 1974), pp. 815–851. DOI: 10.1103/RevModPhys.46.815. URL: https://link.aps.org/doi/10.1103/RevModPhys.46.815.

[41] S. Hallmann. "Sensitivity of ORCA to the neutrino mass hierarchy (Bachelor thesis)." Bachelor's Thesis. Friedrich-Alexander University Erlangen-Nürnberg, 2013.

[42] D. Crevier. *AI: The Tumultuous History of the Search for Artificial Intelligence*. New York, NY, USA: Basic Books, Inc., 1993. ISBN: 0-465-02997-3.

[43] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[44] J. R. Kager. "Investigation on Applying Deep Learning Methods for Event Reconstruction in IceCube." Bachelor's Thesis. Technische Universität München, 2017.

[45] A. Singh. Apr. 3, 2018. URL: https://www.slideshare.net/hortonworks/data-science-workshop.

[46] S. Geißelsöder. "Model-independent search for neutrino sources with the ANTARES neutrino telescope." PhD thesis. Friedrich-Alexander University Erlangen-Nürnberg, 2016.

[47] L. Breiman. "Random Forests." In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[48] URL: https://en.paradigmadigital.com/techbiz/machine-learning-dummies/.

[49] Stanford University. *CS231n: Convolutional Neural Networks for Visual Recognition. Neural Networks 1.* 2016. URL: http://cs231n.github.io/neural-networks-1/.

[50] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[51] *Scheme of a single neuron*. URL: https://i.stack.imgur.com/gzrsx.png.

[52] P. Roelants. *How to implement a neural network Intermezzo*. URL: http://peterroelants.github.io/posts/neural_network_implementation_intermezzo02/.

[53] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[54] Y. Nesterov. "A Method for Solving a Convex Programming Problem with Convergence Rate O(1/K2)." In: *Soviet Mathematics Doklady* 27 (1983), pp. 372–367.

[55] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980.

[56] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*. 2010.

[57] K. He et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.” In: *CoRR* abs/1502.01852 (2015). arXiv: 1502.01852.

[58] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167.

[59] N. Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2670313.

[60] Stanford University. *CS231n: Convolutional Neural Networks for Visual Recognition. Neural Networks 2.* 2016. URL: http://cs231n.github.io/neural-networks-2/ (visited on 04/09/2018).

[61] O. Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge.” English (US). In: *International Journal of Computer Vision* 115.3 (Dec. 2015), pp. 211–252. ISSN: 0920-5691. DOI: 10.1007/s11263-015-0816-y.

[62] Stanford University. *CS231n: Convolutional Neural Networks for Visual Recognition. Convolutional Neural Networks (CNNs): An Illustrated Explanation.* 2016. URL: https://xrds.acm.org/blog/wp-content/uploads/2016/06/Figure1.png (visited on 04/10/2018).

[63] Stanford University. *CS231n: Convolutional Neural Networks for Visual Recognition. Convolutional Networks.* 2019. URL: http://cs231n.github.io/convolutional-networks/.

[64] D. P. Kingma and M. Welling. “An Introduction to Variational Autoencoders.” In: *Foundations and Trends in Machine Learning: Vol. 12 (2019): No. 4, pp 307-392* (June 6, 2019). DOI: 10.1561/2200000056. arXiv: 1906.02691v3.

[65] I. J. Goodfellow et al. “Generative Adversarial Networks.” In: (June 10, 2014). arXiv: 1406.2661v1.

[66] MathWorks. *Convolutional Neural Network.* URL: https://www.mathworks.com/content/mathworks/www/en/discovery/convolutional-neural-network/jcr:content/mainParsys/image_copy.adapt.full.high.jpg/1517522275430.jpg (visited on 04/11/2018).

[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” In: *Advances in Neural Information Processing Systems 25*. 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[68] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In: *CoRR* abs/1409.1556 (2014).

[69] K. He et al. “Identity Mappings in Deep Residual Networks.” In: *Computer Vision – ECCV 2016*. Cham: Springer International Publishing, 2016, pp. 630–645. ISBN: 978-3-319-46493-0.

[70] S. Zagoruyko and N. Komodakis. “Wide Residual Networks.” In: *CoRR* abs/1605.07146 (2016). arXiv: 1605.07146.

[71] C. Szegedy, S. Ioffe, and V. Vanhoucke. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.” In: *CoRR* abs/1602.07261 (2016). arXiv: 1602.07261.

[72] G. Carminati et al. “MUPAGE: a fast atmospheric MUon GEnerator for neutrino telescopes based on PArametric formulas.” In: (July 31, 2009). arXiv: 0907.5563v1.

[73] *IN2P3 Computing Centre.* URL: https://cc.in2p3.fr/en/.

[74] C. Sun et al. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era.” In: (July 10, 2017). arXiv: 1707.02968v2.

[75] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[76] F. Chollet et al. *Keras*. https://keras.io. 2015.

[77] A. Paszke et al. "Automatic differentiation in PyTorch." In: *NIPS-W*. 2017.

[78] J. Nickolls et al. "Scalable Parallel Programming with CUDA." In: *Queue* 6.2 (Mar. 2008), pp. 40–53. ISSN: 1542-7730. DOI: 10.1145/1365490.1365500.

[79] F. Seide and A. Agarwal. "CNTK: Microsoft's Open-Source Deep-Learning Toolkit." In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 2135–2135. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2945397.

[80] *RRZE: High Performance Computing at FAU*. Dec. 26, 2019. URL: https://hpc.fau.de/.

[81] S. Chetlur et al. "cuDNN: Efficient Primitives for Deep Learning." In: (Oct. 3, 2014). arXiv: 1410.0759v3.

[82] M. Moser and S. Reck. *OrcaNet*. URL: https://github.com/ViaFerrata/OrcaNet.

[83] R. K. Srivastava, K. Greff, and J. Schmidhuber. "Highway Networks." In: (2015). arXiv: 1505.00387.

[84] G. Huang, Z. Liu, and K. Q. Weinberger. "Densely Connected Convolutional Networks." In: (2016). arXiv: 1608.06993.

[85] *Tensorflow API documentation*. URL: https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer.

[86] *NuFIT 3.2 (2018)*. Dec. 12, 2019. URL: www.nu-fit.org.

[87] S. Adrián-Martínez et al. "Intrinsic limits on resolutions in muon- and electron-neutrino charged-current events in the KM3NeT/ORCA detector." In: *J. High Energ. Phys. (2017) 2017: 8* (Nov. 29, 2016). DOI: 10.1007/JHEP05(2017)008. arXiv: 1612.05621v2.

[88] S. Bourret and L. Quinn. *Sensitivity of ORCA to the neutrino mass ordering and oscillation parameters*. June 2018. DOI: 10.5281/zenodo.1300771.

[89] R. C. Geary. "The ratio of the mean deviation to the standard deviation as a test of normality." In: *Biometrika* 27.3-4 (Oct. 1935), pp. 310–332. ISSN: 0006-3444. DOI: 10.1093/biomet/27.3-4.310. eprint: http://oup.prod.sis.lan/biomet/article-pdf/27/3-4/310/660121/27-3-4-310.pdf.

[90] B. Strandberg and L. Nauta. *MONA: Mass Ordering Nikhef Analysis*. 2019. URL: https://git.km3net.de/common/analysis/MONA.

[91] W. Verkerke and D. Kirkby. "The RooFit toolkit for data modeling." In: (June 2003). arXiv: physics/0306116.

[92] I. Esteban et al. "Updated fit to three neutrino mixing: exploring the accelerator-reactor complementarity." In: *Journal of High Energy Physics* 2017.1 (Jan. 2017), p. 87. DOI: 10.1007/JHEP01(2017)087.

[93] L. Maderer. "Sensitivity to tau-neutrino appearance with the first seven strings of KM3NeT/ORCA." Master thesis. 2019.

[94] C. Andreopoulos et al. "The GENIE Neutrino Monte Carlo Generator." In: *Nucl. Instrum. Meth.* 614 (2010), pp. 87–104. DOI: 10.1016/j.nima.2009.12.009. arXiv: 0905.2517v2.

[95] I. Antcheva et al. "ROOT — A C++ framework for petabyte data storage, statistical analysis and visualization." In: *Computer Physics Communications* 180.12 (2009), pp. 2499–2512. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2009.08.005. URL: http://www.sciencedirect.com/science/article/pii/S0010465509002550.

[96] A. Aurisano et al. “A Convolutional Neural Network Neutrino Event Classifier.” In: *2016 JINST 11 P09001* (Apr. 5, 2016). DOI: 10.1088/1748-0221/11/09/P09001. arXiv: 1604.01444v3.

[97] D. Guest, K. Cranmer, and D. Whiteson. “Deep Learning and its Application to LHC Physics.” In: *Ann. Rev. Nucl. Part. Sci.* 68 (2018), pp. 161–181. DOI: 10.1146/annurev-nucl-101917-021019. arXiv: 1806.11484.

[98] M. Huennefeld. “Deep Learning in Physics exemplified by the Reconstruction of Muon-Neutrino Events in IceCube.” In: *PoS* ICRC2017 (2018), p. 1057. DOI: 10.22323/1.301.1057.

[99] M. Erdmann, J. Glombitza, and D. Walz. “A Deep Learning-based Reconstruction of Cosmic Ray-induced Air Showers.” In: *Astropart. Phys.* 97 (2018), pp. 46–53. DOI: 10.1016/j.astropartphys.2017.10.006. arXiv: 1708.00647.

[100] I. Shilon et al. “Application of Deep Learning methods to analysis of Imaging Atmospheric Cherenkov Telescopes data.” In: *Astropart. Phys.* 105 (2019), pp. 44–53. DOI: 10.1016/j.astropartphys.2018.10.003. arXiv: 1803.10698.

List of Figures

List of Tables

Appendix

A.1 Time windows for CNN image generation

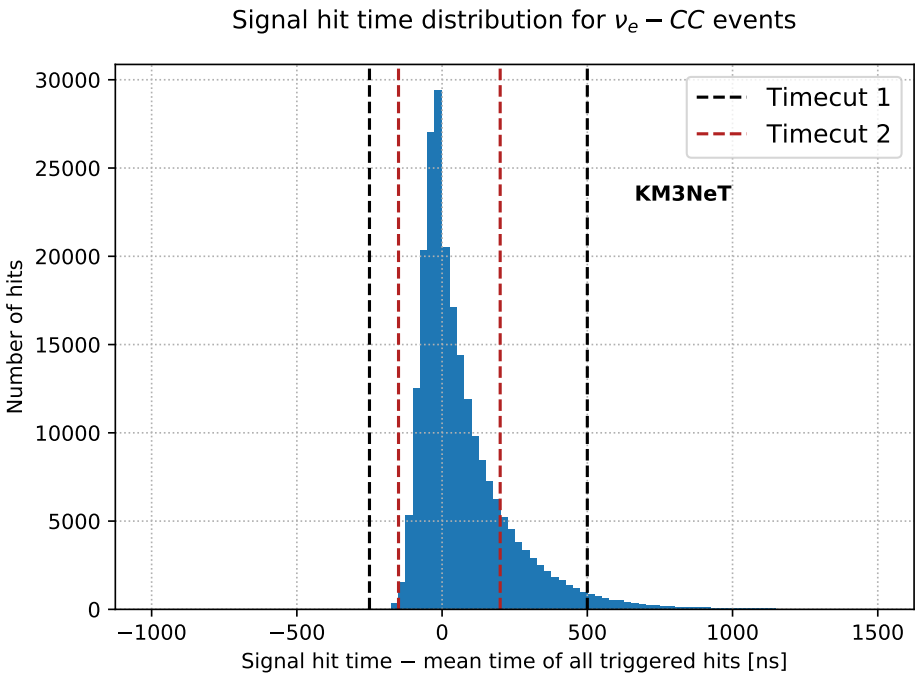

Fig. A.1: Time distribution of ν_e^{CC} signal hits centered with the mean time of the triggered hits for single events. For this distribution, about 3000 ν_e^{CC} events have been used. The dashed lines in black and red visualize possible timecuts.

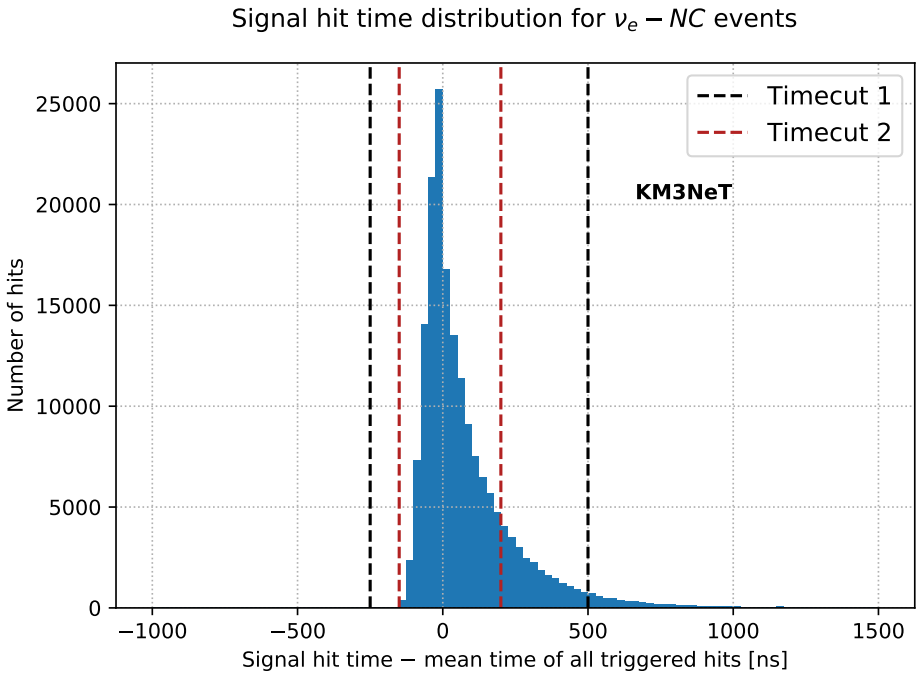

Fig. A.2: Time distribution of ν_e^{NC} signal hits centered with the mean time of the triggered hits for single events. For this distribution, about 3000 ν_e^{NC} events have been used. The dashed lines in black and red visualize possible timecuts.

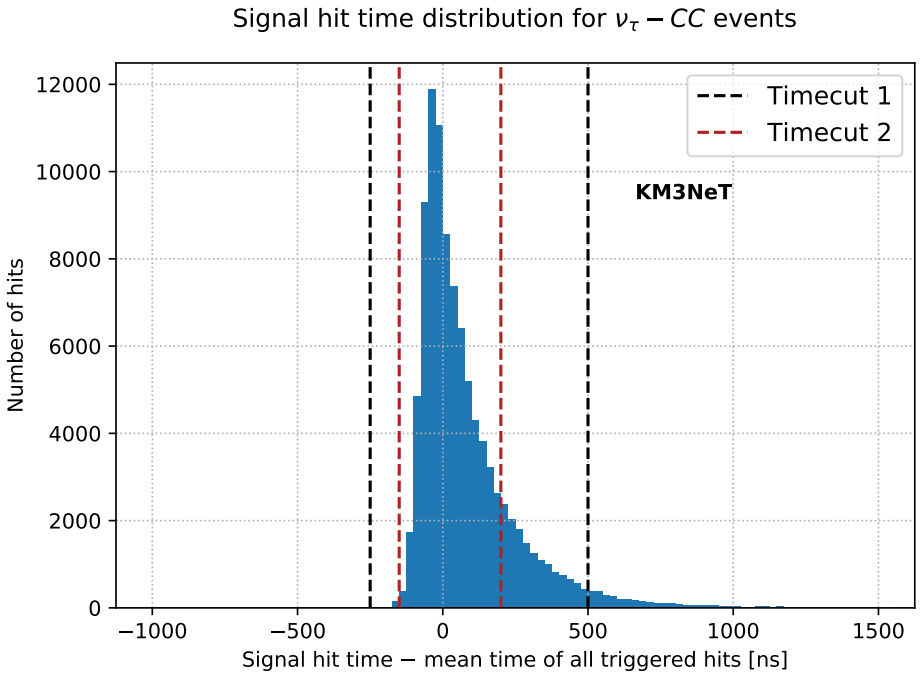

Fig. A.3: Time distribution of ν_τ^{CC} signal hits centered with the mean time of the triggered hits for single events. For this distribution, about 3000 ν_τ^{CC} events have been used. The dashed lines in black and red visualize possible timecuts.

A.2 Track-shower classifier

A.2.1 XYZ-T/P network using timecut 1/2

Building block / layer	Output dimension		
XYZ-T Input using timecut 1 or 2	$11 \times 13 \times 18$	$\times$	60
XYZ-P Input using timecut 1 or 2	$11 \times 13 \times 18$	$\times$	31
Stacked XYZ-T + XYZ-P Input	$11 \times 13 \times 18$	$\times$	91
Convolutional block (64 filters, $\delta = 0.1$)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$)	$11 \times 13 \times 18$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$, $\rho = (2, 2, 2)$)	$5 \times 6 \times 9$	$\times$	64
Convolutional block (128 filters, $\delta = 0.1$)	$5 \times 6 \times 9$	$\times$	128
Convolutional block (128 filters, $\delta = 0.1$)	$5 \times 6 \times 9$	$\times$	128
Convolutional block (128 filters, $\delta = 0.1$)	$5 \times 6 \times 9$	$\times$	128
Convolutional block (128 filters, $\delta = 0.1$, $\rho = (2, 2, 2)$)	$2 \times 3 \times 4$	$\times$	128
Flatten	3072		
Dense 128 + ReLu	128		
Dropout ($\delta = 0.1$)	128		
Dense 32 + ReLu	32		
Dense 2 + Softmax	2		

Table A.1: Network structure of the single input, track-shower classifier CNN with XYZ-T/P input and either timecut 1 or timecut 2. The symbol δ specifies the dropout rate and the symbol ρ specifies the stride for the maximum pooling operation used in the respective convolutional block.

A.2.2 YZT-X network using timecut 1/2

Building block / layer	Output dimension		
YZT-X Input using timecut 1 or 2	$13 \times 18 \times 60$	$\times$	11
Convolutional block (64 filters, $\delta = 0.1$)	$13 \times 18 \times 60$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$)	$13 \times 18 \times 60$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$, $\rho = (1, 1, 2)$)	$13 \times 18 \times 30$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$)	$13 \times 18 \times 30$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$)	$13 \times 18 \times 30$	$\times$	64
Convolutional block (64 filters, $\delta = 0.1$, $\rho = (2, 2, 2)$)	$6 \times 9 \times 15$	$\times$	64
Convolutional block (128 filters, $\delta = 0.1$)	$6 \times 9 \times 15$	$\times$	128
Convolutional block (128 filters, $\delta = 0.1$)	$6 \times 9 \times 15$	$\times$	128
Convolutional block (128 filters, $\delta = 0.1$)	$6 \times 9 \times 15$	$\times$	128
Convolutional block (128 filters, $\delta = 0.1$, $\rho = (2, 2, 2)$)	$3 \times 4 \times 7$	$\times$	128
Flatten	10752		
Dense 128 + ReLu	128		
Dropout ($\delta = 0.1$)	128		
Dense 32 + ReLu	32		
Dense 2 + Softmax	2		

Table A.2: Network structure of the single input, track-shower classifier CNN with YZT-X input and either timecut 1 or timecut 2. The symbol δ specifies the dropout rate and the symbol ρ specifies the stride for the maximum pooling operation used in the respective convolutional block.

A.3 Regression of neutrino properties

A.3.1 ν_μ^{CC}

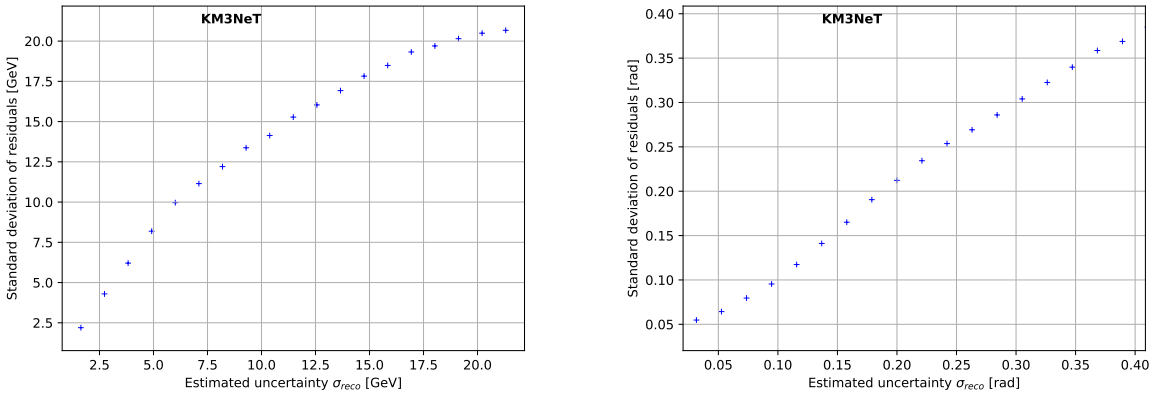

Fig. A.4: Standard deviation of the difference of true MC and reconstructed values versus the uncertainty reconstructed by the CNN event regressor. Left: for the energy reconstruction of ν_μ^{CC} events. Right: for the zenith angle reconstruction of ν_μ^{CC} events.

A.3.2 ν_e^{CC}

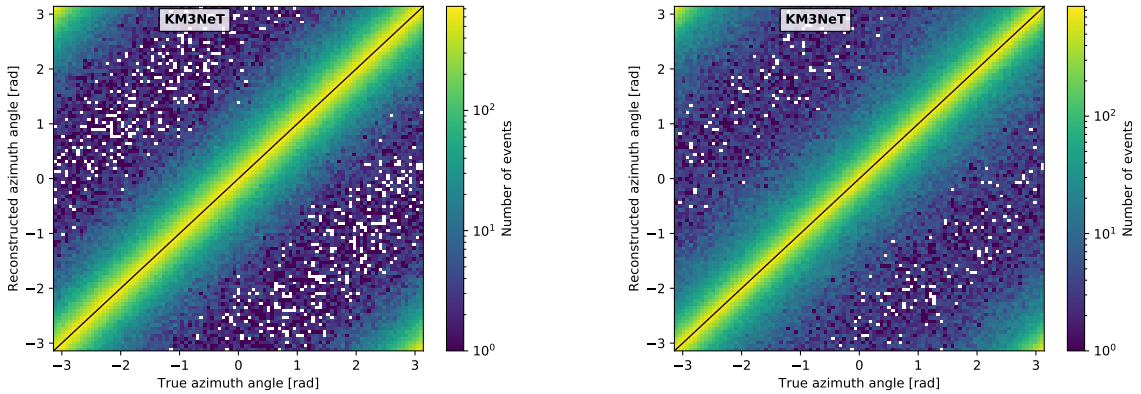

Fig. A.5: Reconstructed azimuth angle vs. true azimuth angle of the CNN event regressor (left) and the TOR reconstruction (right) for ν_e^{CC} events.

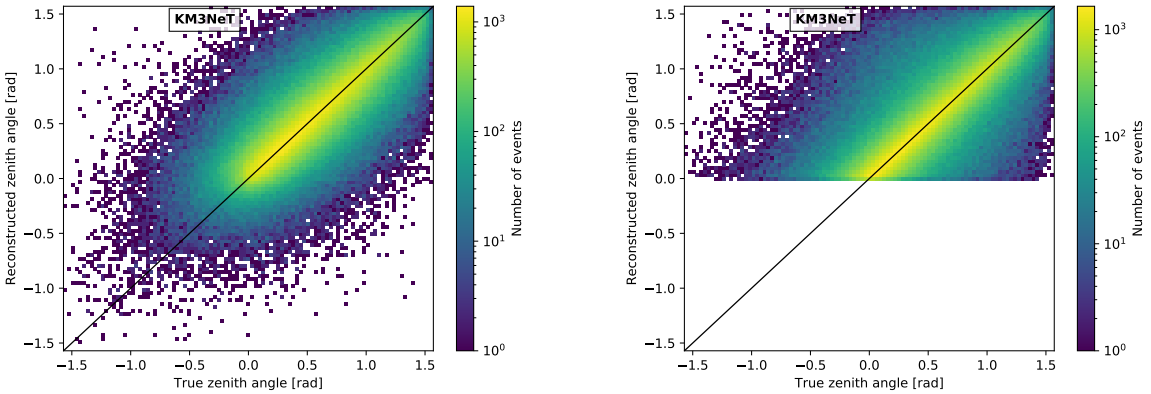

Fig. A.6: Reconstructed zenith angle vs. true zenith angle of the CNN event regressor (left) and the TOR reconstruction (right) for ν_e^{CC} events.

A.3.3 ν_e^{NC}

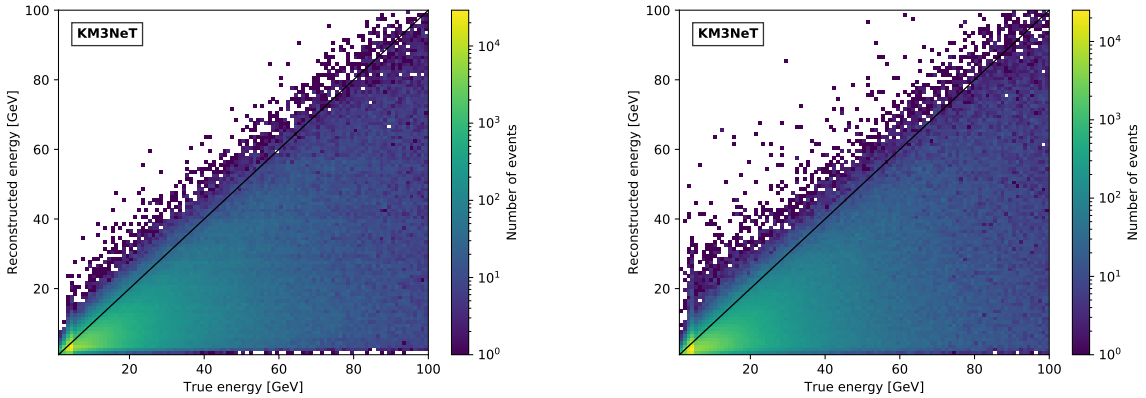

Fig. A.7: Corrected, reconstructed energy vs. true MC neutrino energy for pre-selected ν_e^{NC} events. Left: CNN event regressor. Right: SOR (standard ORCA maximum likelihood reconstruction algorithm for shower-like events).

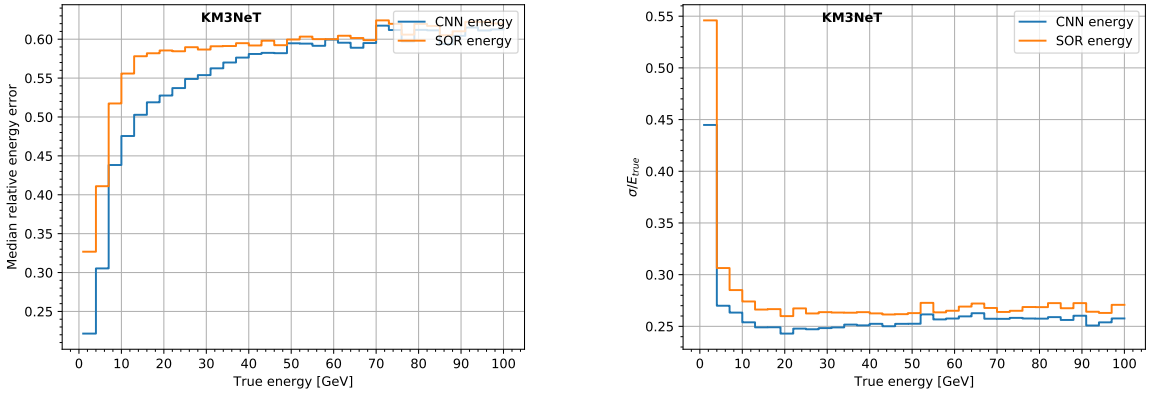

Fig. A.8: Left: median relative error of the reconstructed energy for pre-selected ν_e^{NC} events for the CNN event regressor (blue) and SOR (orange). Right: relative standard deviation of the reconstructed energy for pre-selected ν_e^{NC} events for the CNN event regressor (blue) and SOR (orange).

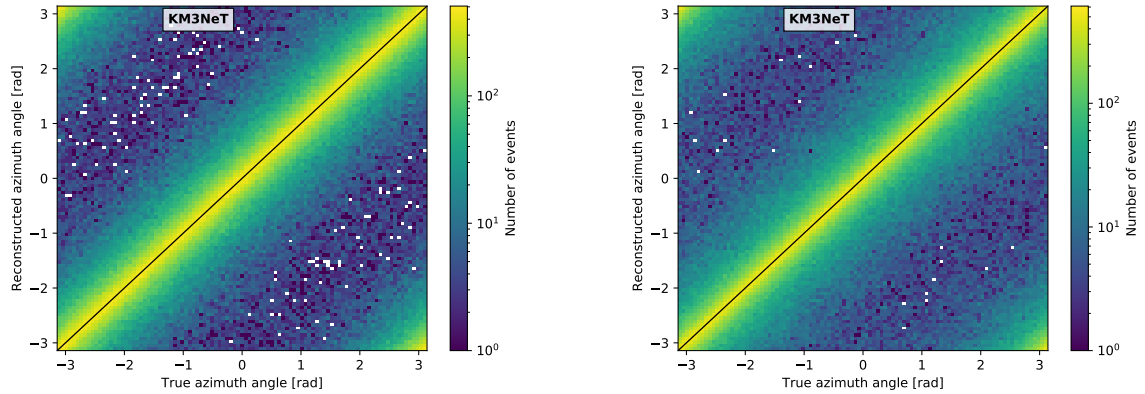

Fig. A.9: Reconstructed azimuth angle vs. true azimuth angle of the CNN event regressor (left) and the TOR reconstruction (right) for ν_e^{NC} events.

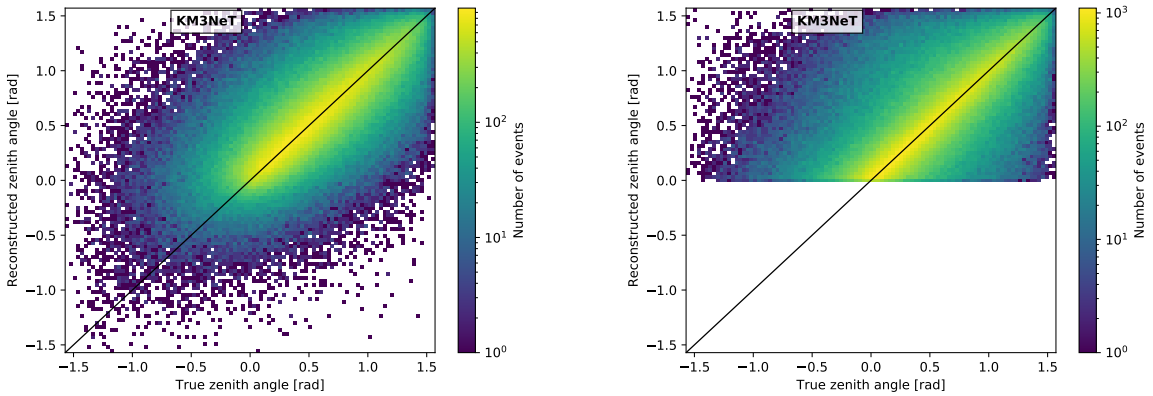

Fig. A.10: Reconstructed zenith angle vs. true zenith angle of the CNN event regressor (left) and the TOR reconstruction (right) for ν_e^{NC} events.

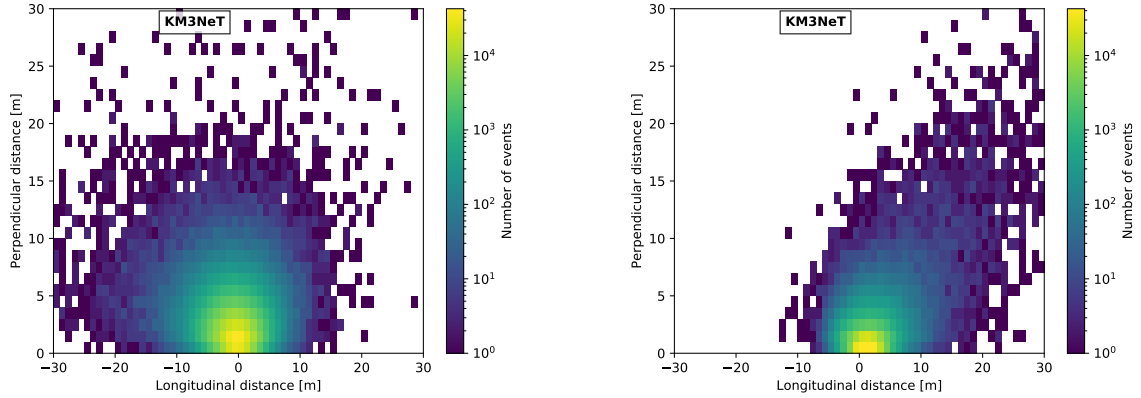

Fig. A.11: Reconstructed neutrino interaction point with respect to the true MC vertex for ν_e^{NC} events. Shown is the perpendicular component versus the longitudinal component of the distance vector. Left: CNN event regressor. Right: TOR.

A.3.4 ν_τ^{CC}

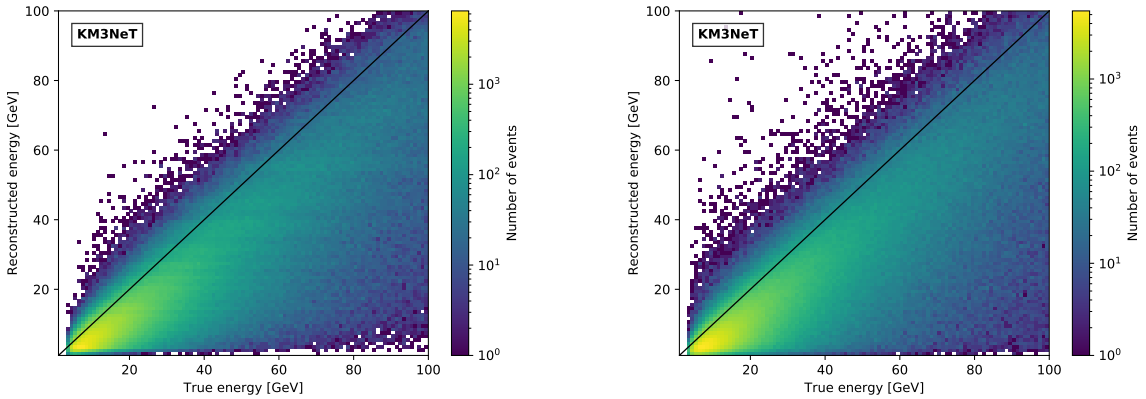

Fig. A.12: Corrected, reconstructed energy vs. true MC neutrino energy for pre-selected ν_τ^{CC} events. Left: CNN event regressor. Right: SOR (standard ORCA maximum likelihood reconstruction algorithm for shower-like events).

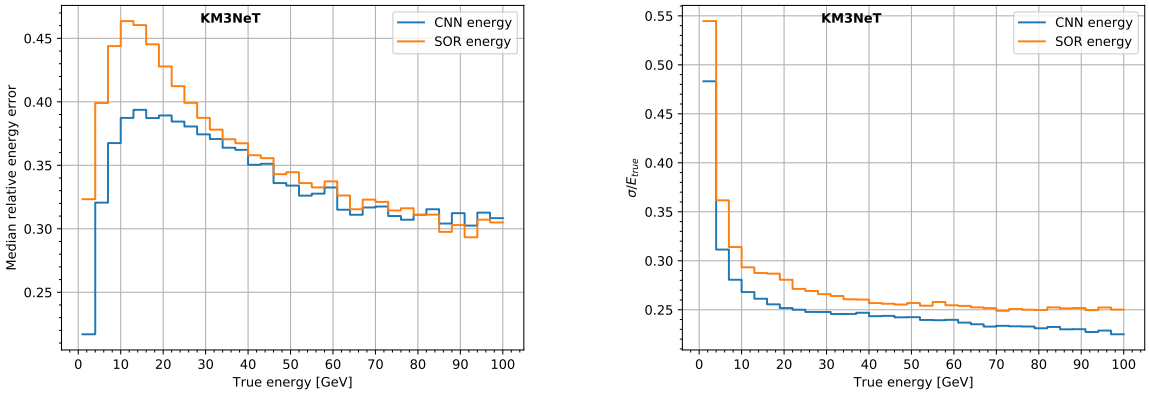

Fig. A.13: Left: median relative error of the reconstructed energy for pre-selected ν_τ^{CC} events for the CNN event regressor (blue) and SOR (orange). Right: relative standard deviation of the reconstructed energy for pre-selected ν_τ^{CC} events for the CNN event regressor (blue) and SOR (orange).

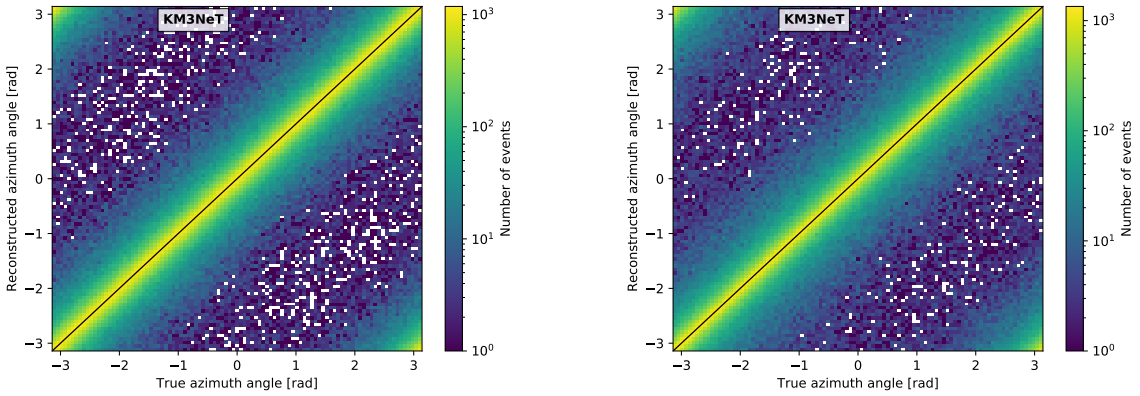

Fig. A.14: Reconstructed azimuth angle vs. true azimuth angle of the CNN event regressor (left) and the TOR reconstruction (right) for ν_τ^{CC} events.

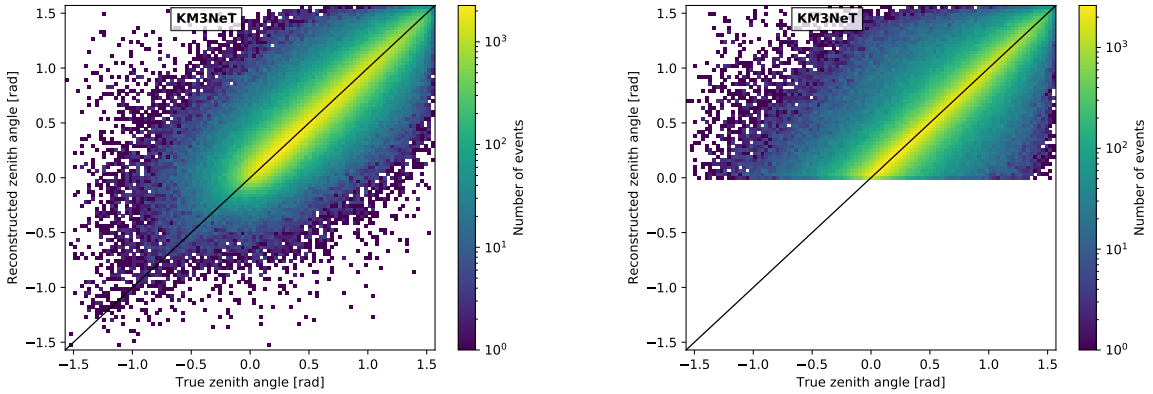

Fig. A.15: Reconstructed zenith angle vs. true zenith angle of the CNN event regressor (left) and the TOR reconstruction (right) for ν_τ^{CC} events.

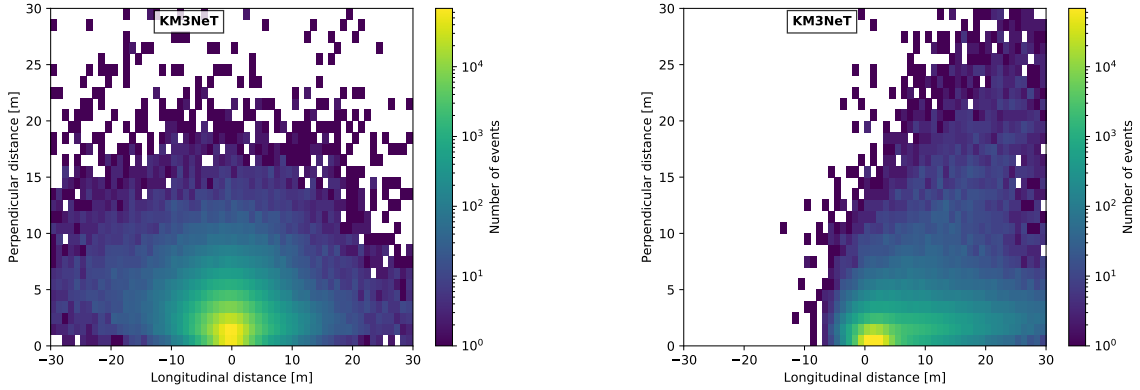

Fig. A.16: Reconstructed neutrino interaction point with respect to the true MC vertex for ν_τ^{CC} events. Shown is the perpendicular component versus the longitudinal component of the distance vector. Left: CNN event regressor. Right: TOR.

Acknowledgements

At this point I would like to thank everyone who supported me during my doctorate. My greatest thanks go to my family and especially to my parents who have always supported me throughout my career: this work would not have been possible without your support over all these years! Thank you so much. I would also like to thank my friends, some of whom have accompanied me on my journey through life since my primary school days. During my school years, my university studies and also during my doctoral thesis I was able to make great friendships, and I am very grateful to you for this.

Apart from this, I'd like to thank the organizers of the International Max Planck Research School for the Physics of Light (IMPRS-PL) for offering me the opportunity to participate in such an interesting PhD program. I'd especially like to thank Dr. Peter Banzer, Heike Schwender and Dorothea Mayer for all their efforts in the IMPRS-PL program.

At last, I would like to thank my dear colleagues in the neutrino group for the great working atmosphere. I would especially like to thank:

Steffen Hallmann, my long-time office colleague and friend, without whom this work would not have been possible! Thank you so much for all the helpful discussions, recommendations and for sharing your in-depth knowledge of KM3NeT. Additionally, thank you for the great atmosphere in our office and for making my life more worthwhile during my PhD, e.g. due to our chit-chats or for organizing the nuBowl.

Dr. Thomas Eberl, for the outstanding support and guidance in my research. You were always open minded for any questions or issues that I had and working with you has been a pleasure!

Stefan Reck, for the great collaboration on OrcaNet and generally for the very helpful machine learning discussions. Thanks a lot!

Dr. Jannik Hofestädt, for the very helpful discussions about the standard ORCA reconstructions and the discussions and proposals concerning my CNN-based reconstruction. Thank you for always having an open door when I had any issues.

Prof. Gisela Anton, my supervisor, for offering to supervise my PhD with such an interesting topic and for your helpful comments on my work. Thank you for always supporting me during my PhD!

Támás Gal, Jonas Reubelt und Johannes Schumann, my colleagues from the neighboring office. Thank you so much for all the chit-chat, the discussions about my work and generally for enriching my life during my PhD. Additionally, I'd really like to thank Tom for teaching me so much about programming and for always being ready to help when I had any, even trivial, problems. And thanks of course as well for joining me in acquiring a boating license!

Dr. Bruno Strandberg, from Nikhef, for the generous help with everything surrounding MONA, thanks a lot!

Dr. Piotr Mijakowski, for agreeing to serve as a second assessor of this thesis.