Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Richtungsrekonstruktion von Elektronen in JUNO mithilfe künstlicher neuronaler Netze

Direction reconstruction of electrons in JUNO using artifical neural networks

— Masterarbeit —

Vorgelegt von:	Hauke Ortwin Schmidt
Matrikelnummer:	6541961
Fakultät:	Mathematik, Informatik und Naturwissenschaften
Fachbereich:	Informatik
Studiengang:	Informatik M.Sc.

Erstgutachterin:	Prof. Dr. Simone Frintrop
Zweitgutachter:	Dr. Björn Wonsak

Hamburg, 20.02.2020

Zusammenfassung

Mit dem JUNO-Experiment wird ein Detektor mit 20.000 Tonnen Flüssigszintillator gebaut, um die Eigenschaften der Neutrinos zu untersuchen. Eine wichtige Information ist dabei die Flugrichtung des Neutrinos, weil sich darüber Rückschlüsse auf die mögliche Quelle schließen lassen. Für eine Sorte Neutrinos, die Elektronneutrinos aus der Sonne, ist dies besonders schwierig, weil die Elektronen, mit denen sie elastisch stoßen, nur wenige Zentimeter weit fliegen. Mit Hilfe der Separation von Čerenkov- und Szintillationslicht und dem Einsatz eines neuronalen Netzes für dreidimensionale Punktwolken wurde eine Rekonstruktion der Flugrichtung durchgeführt. Als Vorlage für das neuronale Netz wurde dabei „Dynamic Graph CNN“ verwendet und an die Aufgabenstellung angepasst. Dabei konnte die Flugrichtung eines Elektrons mit einer Energie von 4 MeV für 68% der Ereignisse mit einer maximalen Abweichung von 37,8° ermittelt werden.

Abstract

The JUNO experiment is a detector with 20,000 tons of liquid scintillator and will be built to research the properties of neutrinos. One important information is the direction of flight of the neutrino, because this allows conclusions about its possible source. For one type of neutrinos, electron neutrinos from the sun, this is particularly difficult because the electrons with which they elastically collide fly only a few centimetres. With the help of the separation of Čerenkov light and scintillation light and the use of a neural network for three-dimensional point clouds, a reconstruction of the flight direction was performed. As a template for the neural network “Dynamic Graph CNN” was used and adapted to the task. The flight direction of an electron with an energy of 4 MeV could be determined for 68% of the events with an maximum error of 37.8°.

Inhaltsverzeichnis

1. Einleitung

Das menschliche Gehirn ist in vielen Teilbereichen heutigen Computersystemen überlegen. Gerade in der Erfassung und Einordnung großer Datenmengen vollzieht es Höchstleistungen. Die Experimentalphysik ist heutzutage auf die effiziente Verarbeitung großer Datenmengen angewiesen. Die Fähigkeiten des menschlichen Gehirns auf Computer zu übertragen und für die Bewältigung von rechenintensiven Problemen zu nutzen, ist ein aktuelles Forschungsfeld.

Mit dem Jiangmen Underground Neutrino Observatory (JUNO) wird ein Neutrinodetektor mit 20.000 Tonnen Flüssigszintillator gebaut. Ziel ist es, die Eigenschaften der Neutrinos zu untersuchen und das Energiespektrum der Oszillation der Neutrinoflavoureigenzustände zu bestimmen. Die Hauptquelle für die Neutrinos sind dabei zwei Kernreaktoren, die in etwa 53 Kilometern Entfernung zum Detektor stehen. Bei Reaktionen von Teilchen mit dem Flüssigszintillator wird Licht ausgesendet, das in Photomultiplier Tubes (PMT), die rund um den kugelförmigen Detektor angeordnet sind, Signale auslöst. Dieses Licht setzt sich aus zwei Bestandteilen zusammen, dem Szintillationslicht und dem Čerenkovlicht. Während das Szintillationslicht in alle Richtungen emittiert wird, wird das Čerenkovlicht als Kegel um die Bewegungsrichtung des Teilchens emittiert.

Mit Hilfe einer Simulationssoftware lassen sich im Vorfeld mögliche Ereignisse simulieren. Die Signale aus der Simulation werden mit den PMT-Nummern, den Signalzeiten, dem Ursprung der Teilchenspur und der Richtung der Teilchenspur gespeichert. Der Ursprung der Teilchenspur wird auch als Vertex bezeichnet. Die daraus resultierenden Daten lassen sich als Punktwolke in drei Dimensionen darstellen. Ein Beispiel für ein Elektron-Ereignis in der Detektormitte ist in Abbildung 1.1 dargestellt. Dabei wurde für alle Photonen eine Zeitkorrektur unter Berücksichtigung der mittleren erwarteten Flugzeit zwischen Vertex und Photomultiplier Tube angewendet und alle Photonen mit weniger als 3,5 Nanosekunden Abweichung dargestellt. Mit dem Auge ist keine Schwerpunktrichtung der früh getroffenen PMTs erkennbar.

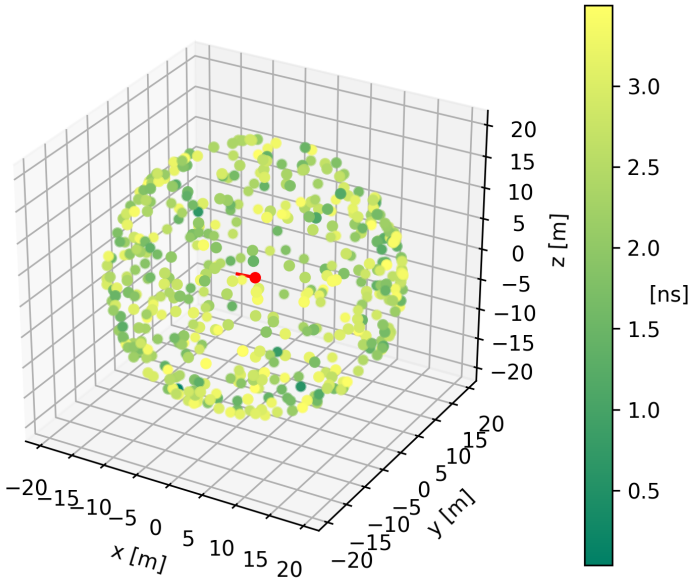

Abbildung 1.1.: Verteilung der Treffer auf den Photomultiplier Tubes für ein Elektron-Ereignis. Die Farbe gibt die Zeit in Nanosekunden seit Eintreten des Ereignisses an. Der Vertex und die Richtung des Ereignisses sind in rot eingezeichnet.

Um aus den Daten die Flugrichtung des Elektrons zu rekonstruieren, wird ein neuronales Netz für Punktwolken angepasst und auf den Daten trainiert. Das Training wird zunächst nur mit den unveränderten Positions- und Zeitdaten aus der Simulation durchgeführt und davon ausgehend das Training und die Aufbereitung der Daten sukzessiv optimiert. Dazu gehört in einem weiteren Schritt die Berücksichtigung der mittleren erwarteten Flugzeit vom Ursprung der Teilchenspur zum Photomultiplier Tube. Um von den simulierten Daten auf die zu erwartenden Messdaten zu schließen, werden weitere Effekte, wie beispielsweise die Streuung der Messzeiten in den PMTs, der sogenannte Transit-Time Spread, auf die Photonen berücksichtigt.

Mit dieser Arbeit wird gezeigt, dass künstliche neuronale Netze auf Punktwolken auch zur Richtungsrekonstruktion in dreidimensionalen Daten effektiv eingesetzt werden können. Die Richtungsrekonstruktion von Elektronen in Flüssigszintillatordetektoren ist bisher kaum untersucht und mit großen Ungenauigkeiten behaftet. Gleichzeitig ist eine effektive Richtungsrekonstruktion wichtig für die Unterscheidung von Untergrund und Signal. Diese Arbeit leistet dazu einen Beitrag, indem im gesamten JUNO-Detektor eine Vorhersage für die Richtung von Elektronen mit nahezu gleichbleibender Qualität erreicht werden kann.

Zur Einordnung in den Anwendungsfall wird in Kapitel 2 ein Überblick über die Physik des Neutrinos und der Lichtentstehung in Flüssigszintillatordetektoren gegeben. Außerdem wird kurz das JUNO-Experiment vorgestellt, um das es in dieser Arbeit im Speziellen geht. Die Idee und die Funktionsweise von künstlichen neuronalen Netzen und deren einzelne Bestandteile folgen in Kapitel 3. Anschließend werden in Kapitel 4 verwandte Arbeiten zur Richtungsrekonstruktion und zur Arbeit auf punktförmigen Datenmengen mit neuronalen Netzen vorgestellt. In Kapitel 5 wird erläutert, wie die Daten für das Training und die Evaluation des in dieser Arbeit modifizierten Netzes simuliert und angepasst wurden. Die einzelnen Schritte der Anpassung des neuronalen Netzes und der Datenaufbereitung werden in Kapitel 6 vorgestellt und evaluiert. Kapitel 7 fasst die Ergebnisse noch einmal abschließend zusammen und gibt einen Ausblick auf die weiteren Optimierungsmöglichkeiten des neuronalen Netzes.
Der in dieser Arbeit entwickelte oder angepasste Programmcode ist unter `https://git.informatik.uni-hamburg.de/3schmidt/directionregressioncnn` verfügbar.

2. Physikalische Grundlagen

In diesem Kapitel werden die physikalischen Grundlagen über das Neutrino erklärt, in dem es in der Anwendung geht. Es wird verdeutlicht, wie Licht in Detektoren entsteht und das Jiangmen Underground Neutrino Observatory (JUNO) vorgestellt.

2.1. Neutrinophysik

Neutrinos entstehen bei vielen Fusions- und Zerfallsreaktionen. Die Entdeckung des Neutrinos beruht auf der Vorhersage, dass beim radioaktiven β-Zerfall neben Elektronen oder Positronen auch noch ein weiteres Teilchen ausgesendet werden muss. Bei einem Zweikörper-Zerfall würde man ein diskretes Energiespektrum erwarten, weil die kinetische Energie des emittierten Teilchens E_{kin_1} dabei durch das Verhältnis seiner Masse m_1 zur gemeinsamen Masse von Teilchen und Rückstoßkern m_2 festgelegt ist [11]:

$$E_{kin_1} = \frac{m_2}{m_1 + m_2} E_0 \tag{2.1}$$

E_0 ist die Bindungsenergie vor dem Zerfall. Da beim β-Zerfall ein kontinuierliches Energiespektrum beobachtet wird, muss zur Einhaltung von Energie- und Impulssatz sowie zur Einhaltung der Drehimpulserhaltung eben dieses weitere Teilchen ausgesendet werden.
Das Standardmodell der Teilchenphysik unterteilt die Elementarteilchen in die Fermionen und die Bosonen. Die Fermionen unterteilen sich weiter in Quarks und Leptonen. Abbildung 2.1 ist zu entnehmen, dass Neutrinos elektrisch ungeladene Elementarteilchen des Standardmodells der Teilchenphysik mit einem Spin ½ sind und zu den Leptonen gehören. Im Standardmodell haben Neutrinos keine Masse und es gibt für jede Leptonenklasse, auch Flavour genannt, ein Neutrino mit entsprechenden Antiteilchen. Es gibt die Flavour Elektron, Myon und Tau. Neutrinos wechselwirken nur über die schwache Wechselwirkung mit den Z-Bosonen und den W-Bosonen. Daher können sie nur indirekt über die bei Reaktionen ausgesendeten Teilchen nachgewiesen werden [24]. Aufgrund der großen Masse

der W- und Z-Bosonen beträgt die Reichweite der schwachen Wechselwirkung nur etwa $2 \cdot 10^{-18}$ Meter [11].

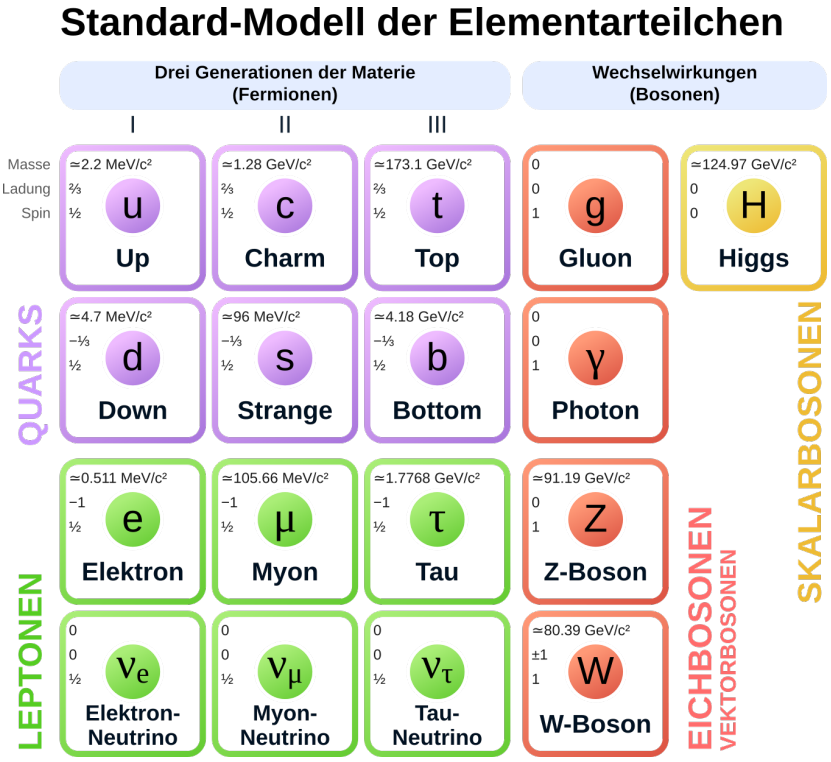

Abbildung 2.1.: Teilchen des Standardmodells der Teilchenphysik [17] (Neutrinomasse entfernt). Die Neutrinos gehören zu den Leptonen unten links und ihre Masse und Ladung im Standardmodell ist Null.

Eine mögliche Quelle für Elektronneutrinos ist die Sonne. In Experimenten mit solaren Neutrinos hat man jedoch weniger Elektronneutrinos gemessen, als erwartet wurden. Dies lässt sich dadurch erklären, dass die Neutrinos ihren Flavour ändern können und sich ein Teil der Elektron-Neutrinos auf dem Weg von der Sonne in Myon-Neutrinos umwandeln. Diese Neutrinooszillation konnte erstmals mit dem Superkamiokande-Detektor, einem Wasser-Čerenkov-Detektor mit 50.000 Tonnen Wasser [7], und Sudbury Neutrino Observatory (SNO), einem Čerenkov-Detektor mit 1.000 Tonnen schwerem Wasser, nachgewiesen werden [4]. Um oszillieren zu können, müssen die Neutrinos aber eine, wenn auch sehr kleine, Ruhemasse haben. Mehr zum Thema Neutrinooszillation findet sich im Buch *Neutrino physics* von Kai Zuber [33].

2.2. Richtungsinformation

Die Flugrichtung der Elektronen liefert zahlreiche Informationen. Wenn solare Elektronneutrinos ν_e elastisch an Elektronen streuen, entspricht ihre Richtung der Richtung der

Neutrinos, da diese bei der Reaktion ihren Impuls teilweise auf das Elektron e^- übertragen. In Abbildung 2.2 ist ein mögliches Feynman-Diagramm eines solchen Streuvorgangs dargestellt.

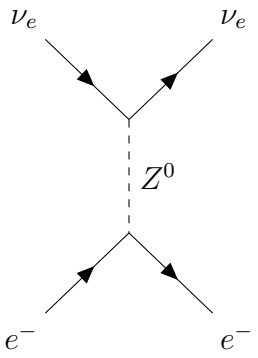

Abbildung 2.2.: Feynman-Diagramm der elastischen Neutrino-Elektron-Streuung unter Austausch eines Z^0-Bosons.

Wenn die Flugrichtung des Elektrons bekannt ist, kann man die Quelle der Neutrinos einschränken. Wenn sie aus der Richtung kommen, in der die Sonne zum aktuellen Zeitpunkt steht, handelt es sich mit einer höheren Wahrscheinlichkeit um solare Neutrinos. Da bei einer Supernova alle Arten von Neutrinos ausgesendet werden, kann auch hier die Richtungsinformation der Elektronen für die Lokalisierung der Supernova behilflich sein. Schlussendlich verbessert die Richtungsinformation auch die Energiebestimmung von Teilchen, da dadurch bessere Startwerte für eine Likelihood-Analyse gewählt werden können. Bei der Likelihood-Analyse wird eine gemessene Menge an Daten mit bereits vorher gemessenen Datenmengen verglichen, für die die Energie bereits bekannt ist. Hingegen gibt die Richtung von Positronen e^+ aus inversen β-Zerfällen kaum Information über die Richtung des Anti-Elektron-Neutrinos $\bar{\nu}_e$, da es sich um einen unelastischen Prozess handelt:

$$\bar{\nu}_e + p \rightarrow e^+ + n \tag{2.2}$$

2.3. Lichtquellen in Flüssigszintillatordetektoren

Fliegen geladene Teilchen durch Materie, so geben sie dabei durch Interaktion mit dem Material, unter anderem durch Stöße, Energie ab. Im Fall von Flüssigszintillatordetektoren wird die Energie wiederum als Licht emittiert. Dabei gibt es zwei Prozesse, bei denen Licht entsteht, zum einen Szintillationslicht, zum anderen Čerenkovlicht.

2.3.1. Szintillationslicht

Flüssigszintillatoren sind eine Mischung aus verschiedenen Kohlenwasserstoffen. Die geladenen Teilchen geben einen Teil ihrer Energie ab, indem sie die Hüllenelektronen des Mediums anregen. Die angeregten Elektronen geben die Energie innerhalb einiger Nanosekunden wieder ab. Bei der Abregung der Moleküle wird die Energie dann als Szintillationslicht isotrop ausgesendet. Mit Hilfe von Beimischungen im Flüssigszintillator wird eine Verschiebung der Wellenlänge des Szintillationslichts erreicht, sodass der Flüssigszintillator für die Photonen durchlässiger ist und mehr im sensitiven Bereich der Photomultiplier Tubes liegt. Der sensitive Bereich der Dynoden-PMTs liegt zwischen 300 nm und 650 nm mit einem Optimalwert bei 420 nm [15]. Bei Energien unter 10 Mega-Elektronenvolt (MeV) fliegen die Elektronen im Detektor nur sehr kurze Strecken in der Größenordnung von einem Zentimeter, sodass die Teilchenspur aufgrund der Auflösung der Photosensoren als annähend punktförmig betrachtet werden kann. Daher lassen sich aus dem Szintillationslicht kaum Richtungsinformationen gewinnen [2]. Wie in Abbildung 2.3 dargestellt, wird das Čerenkovlicht unmittelbar ausgesendet, während das Szintillationslicht aufgrund der Übergangszeiten vom Flüssigszintillator auf die Wellenlängenschieber etwas verzögert und über einen längeren Abklingzeitraum ausgesendet wird.

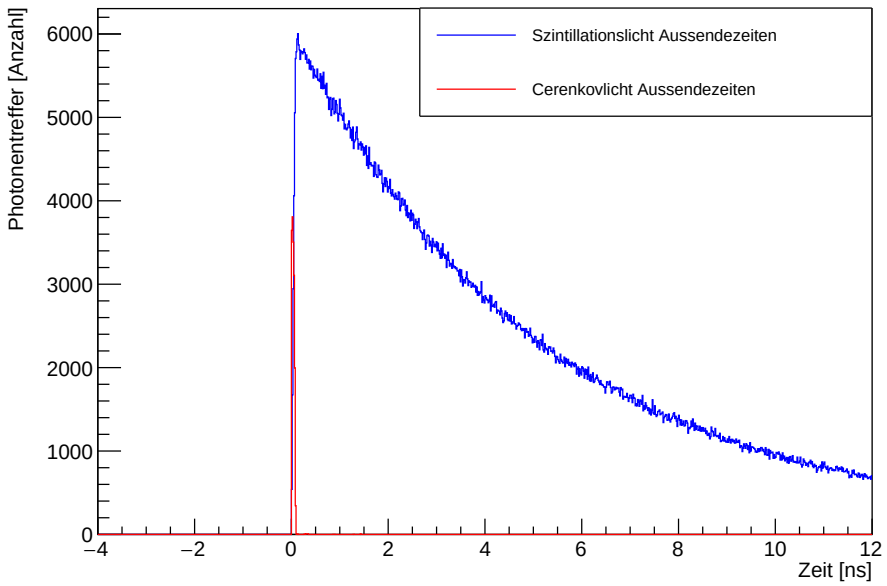

Abbildung 2.3.: Spektrum in der Aussendezeit für Čerenkov- und Szintillationsphotonen für 400 Events mit 4 MeV in der Detektormitte. Die Čerenkovphotonen werden unmittelbar ausgesendet, während die Szintillationsphotonen (blaue Kurve) über eine längeren Zeitraum ausgesendet werden.

Die Wahrscheinlichkeitsdichtefunktion $\Phi_{\text{em}}(t)$ für die Aussendezeit t der Szintillationsphotonen ist abhängig von der Mischung des Szintillators und setzt sich aus der gewichteten Summe der Zerfallsfunktionen zusammen [19]:

$$\Phi_{\text{em}}(t;\tau;\omega) = \sum_{i=1}^{n} \frac{\omega_i}{\tau_i} e^{-\frac{t-t_0}{\tau_i}}, t \geq t_0. \tag{2.3}$$

Mit t_0 wird der Zeitpunkt der Anregung, mit τ_i die mittlere Lebenszeit und mit ω_i das Gewicht der jeweiligen Komponente beschrieben.

2.3.2. Čerenkovlicht

Fliegt ein geladenes Teilchen durch ein elektrisch isoliertes Medium, so erzeugt es dabei an den Molekülen, an denen es vorbei fliegt, eine kurzzeitige Polarisation der Elektronenhülle. Dabei wird ein zeitlich veränderliches elektrisches Dipolmoment induziert, das elektromagnetische Wellen abstrahlt [11]. Für eine Teilchengeschwindigkeit unterhalb der Phasengeschwindigkeit überlagern sich die Wellen verschiedener Atome dabei destruktiv, sodass es zu keiner Lichtemission kommt. Wenn sich das geladene Teilchen dabei hingegen durch das Medium schneller als die Phasengeschwindigkeit des Lichts im Medium bewegt, ist es dabei schneller als die Čerenkovphotonen.

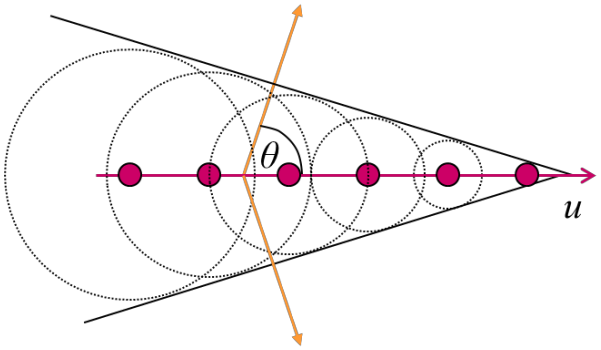

Abbildung 2.4.: Bewegt sich ein Teilchen in Richtung u mit einer Geschwindigkeit größer als die Lichtgeschwindigkeit im Medium, wird Čerenkovlicht (orange Pfeile) mit dem Öffnungswinkel Θ zur Flugrichtung des Teilchens emittiert [5] (bearbeitet).

Deshalb entsteht dabei, wie in Abbildung 2.4 zu erkennen ist, ein Lichtkegel in Bewegungsrichtung, in dem die Photonen konstruktiv interferieren können.

Der Lichtkegel hat den Öffnungswinkel

$$\Theta = \arccos \frac{1}{\beta n} \tag{2.4}$$

mit dem Brechungsindex n des Mediums und $\beta = \frac{v}{c}$, dem Quotienten von Teilchengeschwindigkeit v und Lichtgeschwindigkeit c im Vakuum [24]. Dieses Phänomen ist vergleichbar mit dem Überschallknall beim Dopplereffekt bei Schallwellen, beim dem sich die Amplituden konstruktiv überlagern. Da das Teilchen sehr schnell abgebremst wird, ist es nur kurze Zeit schneller als die Phasengeschwindigkeit des Lichts im Medium und somit entsteht ein so genannter Čerenkovring anstatt eines vollständig ausgefüllten Lichtkreises. Die Anzahl der Čerenkovphotonen hängt dabei von der Geschwindigkeit und damit der Energie des Teilchens ab [2].

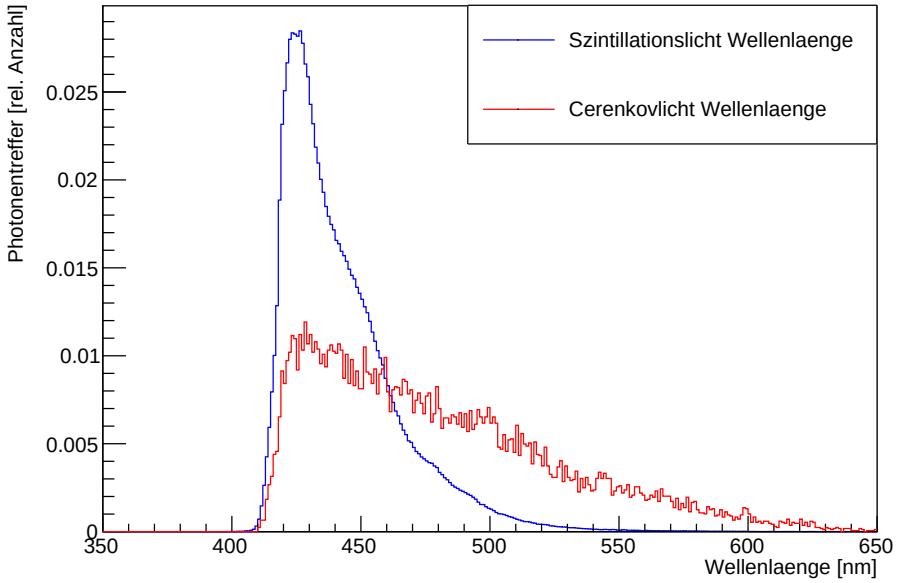

Abbildung 2.5.: Vergleich der detektierten Wellenlängenspektra von Szintillationslicht und Čerenkovlicht aus der JUNO-Simulation für jeweils 400 Elektronenereignisse in der Detektormitte. Dargestellt ist der relative Anteil der jeweiligen Wellenlänge am Spektrum des jeweiligen Lichtprozesses. Während die Wellenlänge für Szintillationslicht steil ansteigt, anschließend auch schnell wieder abfällt, gibt es für Čerenkovlicht einen größeren Anteil an Licht mit größeren Wellenlängen.

Čerenkovlicht mit einer kleineren Wellenlänge als die Absorptionsgrenze des Flüssigszintillators werden absorbiert und als Szintillationslichts wieder ausgesendet [2]. Wie in Abbildung 2.5 dargestellt, liegt die Wellenlänge, unterhalb derer alle Photonen absorbiert werden, bei etwa 410 Nanometern.

Die Ankunftszeit der Photonen hängt zudem von der Phasengeschwindigkeit v_g im Flüssigszintillator ab:

$$v_g(\lambda) = \frac{c}{n(\lambda) - dn(\lambda)/d\log(\lambda)} \tag{2.5}$$

mit dem Brechungsindex n und der Wellenlänge λ.

Wie in Abbildung 2.5 zu erkennen ist, haben die Čerenkovphotonen im JUNO-Detektor beim Auftreffen auf den Photosensoren mit 650 Nanometer eine größere maximale Wellenlänge als die Szintillationsphotonen mit 550 Nanometern. Damit ist im Mittel die Phasengeschwindigkeit der Čerenkovphotonen höher. Durch die höhere Phasengeschwindigkeit und die unmittelbare Aussendung ergibt sich für die Čerenkovphotonen eine etwas frühere Auftreffzeit in den Photosensoren [2].

2.4. Das JUNO-Experiment

Die nachfolgenden Informationen sind, soweit nicht anders angegeben, aus „Neutrino Physics with JUNO“ [6] entnommen.

Das Jiangmen Underground Neutrino Observatory wird nahe der chinesischen Großstadt Jiangmen im Süden Chinas etwa 175 Kilometer westlich von Hongkong gebaut. Im JUNO Experiment sollen mit einem Flüssigszintillatordetektor Interaktionen von Neutrinos mit dem Detektormaterial beobachtet werden. Dabei können die Neutrinos aus verschiedenen Quellen stammen. Hauptquelle sollen die Neutrinos aus zwei Kernkraftwerken sein, die jeweils etwa 53 Kilometer vom Detektor entfernt stehen. Daneben sollen Neutrinos aus den Kernfusionen in der Sonne und aus Quellen im Erdinneren beobachtet werden. Eine weitere mögliche Quelle für Neutrinos stellen Supernovae da, die ebenfalls sehr viele Neutrinos aussenden. Der Detektor steht in einer Experimentierhalle, die sich etwa 700 Meter unter der Erdoberfläche befindet und damit den Detektor von Myonen abschirmt, die durch Reaktionen kosmischer Strahlung in der Atmosphäre entstehen. Die Myonen sorgen im Detektor ebenfalls für Signale, sodass der Detektor während eines Myonendurchfluges in diesem Bereich keine Daten aufzeichnen kann. Die Neutrinos werden vom Gestein oberhalb des Detektors nicht beeinflusst, da sie nur der schwachen Wechselwirkung unterliegen und einen sehr geringen Wechselwirkungsquerschnitt haben.

Der JUNO Detektor besteht neben dem zentralen Flüssigszintillatordetektor noch aus ei-

nem Wasser-Čerenkovdetektor und einem Myonentracker. Der Aufbau ist in Abbildung 2.6 dargestellt. Der Durchmesser der Acrylhülle, in der sich der Flüssigszintillator befindet, beträgt 35,4 Meter. Die ca. 43.000 Photomultiplier Tubes befinden sich auf einem Stahlgerüst mit einem Durchmesser von 40,1 Metern rund um den Flüssigszintillator.

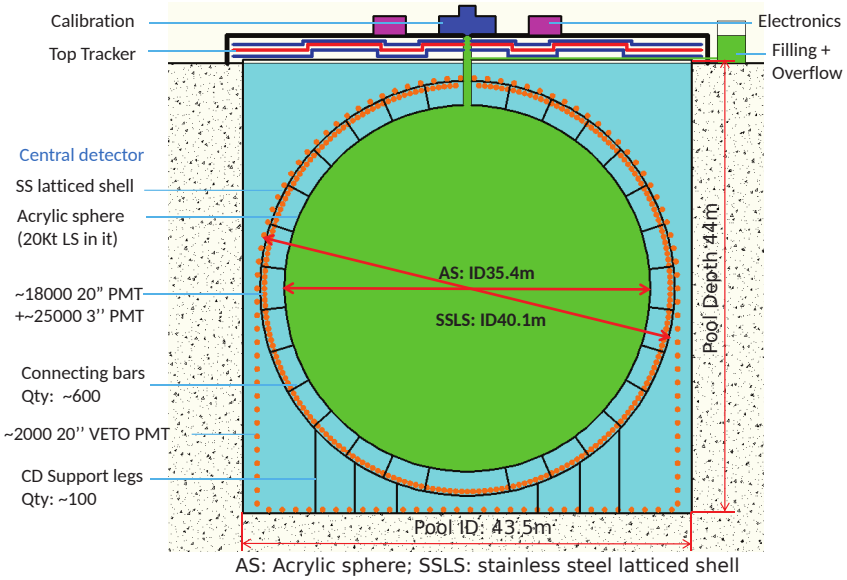

Abbildung 2.6.: Schematischer Aufbau des JUNO Detektors mit dem Flüssigszintillatordetektor (grün) in der Mitte, den PMTs (orange) und dem Wassertank (türkis) rundherum und dem Myonentracker oberhalb davon [17] (Zahlen aktualisiert).

Der Wasser-Čerenkovdetektor ist ein Wassertank rund um den zentralen Detektor herum und fungiert sowohl als Abschirmung für natürliche Radioaktivität aus dem umgebenden Gestein, als auch als Veto-Detektor für kosmische Myonen. Dazu ist der Wassertank mit 2000 Photomultiplier Tubes ausgestattet, die das Čerenkovlicht der Myonen detektieren und so die Spur der Myonen erfassen. Oberhalb des Wassertanks befindet sich der Myonentracker, der die genaue Eintrittsposition der Myonen erfassen soll [6].

2.4.1. Flüssigszintillator

Der Flüssigszintillator im JUNO Detektor besteht aus Linear Alkyl Benzene (LAB), der mit 3 Gramm pro Liter 2,5-diphenyloxazole (PPO) und 15 Milligramm pro Liter p-bis(o-methylstyryl)-benzene (bis-MSB) als Wellenlängenschieber versetzt ist [6]. Für den Flüssigszintillator wird angenommen, dass etwa 10^4 Photonen pro MeV im Detektor deponierter Energie ausgesendet werden. Die ursprünglich im ultravioletten und blauen

Bereich emittierten Photonen werden mit den Wellenlängenschiebern auf eine Wellenlänge von etwa 430 Nanometern gebracht. Ohne Wellenlängenschieber würden die Photonen vom Szintillator wieder absorbiert werden.
Für die Emissionszeit des Flüssigszintillators LAB zusammen mit den Wellenlängenschiebern nach Formel 2.3 wird in der JUNO-Simulation eine schnelle Komponente mit einer Zerfallsdauer $\tau_1 = 4,93ns$ und einem Gewicht $\omega_1 = 0,799$ und eine langsame Komponente mit einer Zerfallsdauer $\tau_2 = 20,6ns$ und einem Gewicht $\omega_2 = 0,201$ angenommen. Mit diesem Flüssigszintillator wird eine Energieauflösung von 3% bei einer Energie von 1 MeV angestrebt, wobei in den Photosensoren etwa 1.100 Photoelektronen pro MeV deponierter Energie erwartet werden. Damit möglichst wenig Photonen auf dem Weg von ihrem Entstehungsort bis zu den Photosensoren absorbiert werden, ist eine hohe Transparenz und damit große Absorptionslänge notwendig. Für JUNO wird eine Absorptionslänge von mindestens 22 Metern bei 430 Nanometern Wellenlänge angestrebt [3].

2.4.2. Signale und Untergrund

Das Hauptsignal, das in JUNO untersucht werden soll, ist der Fluss der Anti-Elektron-Neutrinos aus den Kernkraftwerken. Diese werden im Detektor über den inversen β-Zerfall nachgewiesen, bei dem ein Positron und ein Neutron entstehen. Das Positron gibt seine Energie sehr schnell ab und annihiliert mit einem Elektron unter Abgabe von zwei Gamma-Quanten mit jeweils 511 Kiloelektronenvolt (keV). Das Neutron wird abgebremst und nach etwa 200 Mikrosekunden von einem Proton eingefangen und gibt dabei ein Gamma-Quanten mit 2,2 MeV Energie ab. Die Gamma-Quanten geben ihre Energie mit Hilfe der Compton-Streuung an den Flüssigszintillator ab. Diese Kombination aus dem Signal der zwei ersten Gamma-Quanten und dem zeitverzögerten dritten Signal wird auch als Koinzidenzsignal bezeichnet [6]. Bei der Untersuchung der Oszillation der Reaktorneutrinos gibt es mehrere Untergrundquellen, die ebenfalls Koinzidenzsignale auslösen, die aus den Daten herausgefiltert werden müssen. Die Hauptquellen sind dabei zufällig koinzident auftretende Untergrundsignale, beispielsweise aus radioaktiven Zerfällen im umgebenden Gestein und β-Zerfälle von ^{8}He und ^{9}Li [6]. ^{8}He und ^{9}Li entstehen im Detektor durch Spallation, die durch kosmische Myonen induziert wird, aus dem Kohlenstoff des Szintillators. Für die weiteren untersuchten Neutrinoquellen treten weitere Untergrundquellen auf. Durch verschiedene Zeit-, Orts-, und Energiefilter auf den Daten kann ein Großteil dieser Untergrundsignale herausgefiltert werden.

2.4.3. Photomultiplier Tube

Die Photonen aus Čerenkovlicht und Szintillationslicht werden von Photoelektronenvervielfacherröhren, auf englisch Photomultiplier Tubes (PMTs), registriert. Die PMTs sind Messeinheiten, die aus den eintreffenden Photonen einen Spannungspuls erzeugen. Treffen Photonen auf die Photokathode, werden dort Photoelektronen erzeugt, die in einem elektrischen Feld innerhalb des PMTs beschleunigt werden und mit Hilfe eines Sekundärelektronenverstärkers einen Spannungspuls erzeugen [31]. Bei den Sekundärelektronenverstärkern gibt es verschiedene Bauarten.

Der Dynoden-Verstärker besteht aus mehreren Elektroden, auf die die Elektronen beschleunigt werden und aus denen diese jeweils mehrere Sekundärelektronen herausschlagen. Dadurch erhöht sich von Elektrode zu Elektrode die Anzahl an Elektronen exponentiell, bis diese an der letzten Elektrode einen messbaren Strom über einen Widerstand erzeugen und so das Ausgangssignal generieren [32].

Beim Microchannel-Plate-Verstärker liegt die Beschleunigungsspannung entlang einer Platte an, die mit vielen zwischen 6 und 12 Mikrometer kleinen Kanälen durchbohrt ist. Die Kanäle sind um bis zu 12° gekippt, damit die Elektronen nicht ohne Kollisionen durch die Kanäle durchfliegen können [27]. Fliegt ein Elektron durch einen Kanal, so werden bei Kollisionen mit der Wand Sekundärelektronen herausgelöst [31].

Beim JUNO-Detektor werden rund um den Flüssigszintillatordetektor etwa 18.000 PMTs mit einem Durchmesser von 51 Zentimetern (20 Zoll) und etwa 25.000 PMTs mit einem Durchmesser von 7,6 Zentimetern (3 Zoll) angebracht. Von den großen PMTs sind etwa 5.000 Dynoden-PMTs (R12860) von Hamamatsu und etwa 13.000 Microchannel-Plate (MCP) PMTs von North Night Vision Tech. (NNVT). Bei den kleinen PMTs handelt es sich ebenfalls um Dynoden-PMTs. In Abbildung 2.7 ist der Aufbau eines MCP-PMTs mit beschichteter Glaskuppel (blau) und Sockel mit MCP dargestellt. In der unteren Hälfte der Glaskuppel ist eine Reflexionsbeschichtung, die Photonen, die nicht beim ersten Durchgang durch die Photokathode Elektronen ausgelöst haben, zurück auf die Photokathode lenken, um im zweiten Durchgang ein Photoelektron auszulösen. Die Standardabweichung für die zeitliche Messungenauigkeit der Photomultiplier Tubes, die auch als Flugzeit-Streuung (*engl. Transit-Time Spread, kurz tts*) bezeichnet wird, beträgt für die Dynoden-PMTs zwischen 1 und 1,5 Nanosekunden [18].

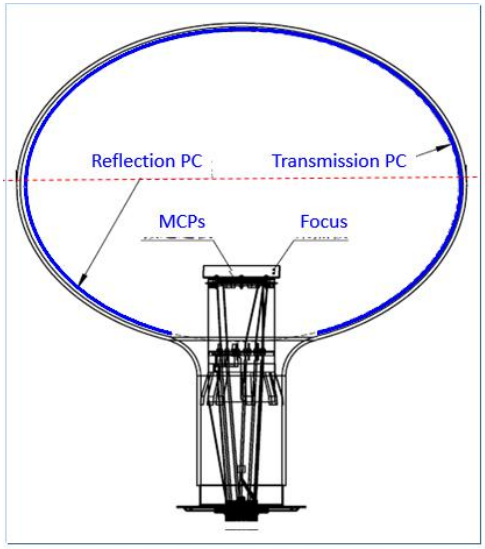

Abbildung 2.7.: Schematischer Aufbau eines MCP-PMTs, wie er im JUNO Detektor eingesetzt wird. Oberhalb der roten Linie ist die Photokathode auf die Glaskuppel aufgedampft [14]. In der unteren Hälfte ist eine Reflexionsschicht aufgetragen, um Photonen, die nicht beim ersten Durchlauf in der Kathode ein Elektron ausgelöst haben, erneut auf die Kathode zu schicken.

Die Streuung entsteht dadurch, dass die Flugzeit der Photonen zwischen der auf die Glaskuppel aufgedampften Kathode und der Dynode je nach Einfallswinkel und Wirkung des elektrischen Feldes innerhalb des PMTs variiert. Für die MCP-PMTs ist die Standardabweichung aktuell noch nicht genau bestimmt, liegt aber etwa im Bereich 5-6 Nanosekunden [14]. Für die in dieser Arbeit beschriebenen Verschmierungen wurde von einer Standardabweichung von 1,274 Nanosekunden für die Dynoden-PMTs, 5,096 Nanosekunden für die MCP-PMTs und 1,911 Nanosekunden für die 3-Zoll-PMTs ausgegangen. Die Werte entsprechen den Annahmen, die in der Elektroniksimulation der JUNO-Software getroffen wurden (Stand Juli 2019).

3. Künstliche neuronale Netze

Künstliche neuronale Netze versuchen die Funktionsweise des menschlichen Gehirns zu imitieren. Die Idee wurde bereits 1943 von Warren S. McCulloch und Walter Pitts eingeführt [21], erlangte aber erst in den letzten Jahren durch die gesteigerte Rechenleistung der Computer, insbesondere durch Grafikkarten, seine heutige Bedeutung.

3.1. Das menschliche Gehirn

Das menschliche Gehirn ist aus Nervenzellen, den Neuronen, aufgebaut, die über Nervenzellfortsätze Informationen austauschen. Die Kontakte zwischen den einzelnen Neuronen verändern sich ständig. Dabei führt ein intensiver Kontakt zwischen zwei Neuronen dazu, dass der Kontakt verstärkt wird. Wenn ein Kontakt lange nicht genutzt wurde, wird er wieder abgebaut [20]. Das Gehirn kann sich dadurch immer wieder an neue Situationen anpassen. Die einzelnen Neuronen geben die Informationen über elektrische Spannungspulse weiter. Das Gehirn ist hierarchisch aufgebaut, wobei die Muster der Spannungspulse von Ebene zu Ebene komplexer werden. So werden die Nervenzellen der untersten Ebene beim Sehen bereits bei Licht aktiviert, während die Nervenzellen der darüber liegenden Ebenen erst bei Licht aus einer bestimmten Richtung, in einer bestimmten Bewegung und mit einer bestimmten Geschwindigkeit aktiviert werden [20].

Ziel eines künstlichen neuronalen Netzes ist es, eine möglichst genaue Abbildung einer Menge an Eingabedaten auf Ausgabedaten zu erreichen. Die Abbildung wird auch als Modell bezeichnet. Damit unterscheidet sich ein künstliches neuronales Netz von einem klassischen Computerprogramm, weil die Gewichte des Modells immer wieder angepasst werden.

3.2. Das Perzeptron

Angelehnt an den Aufbau der Nervenzellen des Gehirns ist die Funktionsweise des Perzeptrons. Dabei gibt es in jeder Schicht (*engl. Layer*) meist mehrere Elemente, die mehrere Eingabewerte x_i erhalten und einen Ausgabewert y erzeugen. Die einzelnen Werte werden mit Gewichten w_i multipliziert, bevor sie zusammen addiert werden, ein Schwellwert s für die Aktivierung subtrahiert und mit Hilfe der Aktivierungsfunktion Θ ein Ausgabewert berechnet wird:

$$y = \Theta\left(\Sigma_i w_i \cdot x_i - s\right) \tag{3.1}$$

Die Aktivierungsfunktion Θ kann unterschiedlich aussehen; die häufig verwendete ReLu-Funktion setzt den Ausgabewert für negative Werte auf 0, positive Werte werden unverändert weitergegeben:

$$\Theta x = \max(0, x) \tag{3.2}$$

Trainiert man das Netz, so werden die einzelnen Gewichte w_i angepasst. Dazu wird die Abweichung des Ausgabewertes y_i vom erwarteten Ausgabewert $\hat{y}_i$ berechnet. Für die Anpassung der Gewichte mit der Lernrate η wird die **Delta-Regel** verwendet [13]:

$$w_{i_{\text{new}}} = w_{i_{\text{old}}} + \eta \cdot (y_i - \hat{y}_i) \cdot x_i \tag{3.3}$$

Eine Verbindung mehrerer Schichten von Perzeptronen hintereinander heißt Multi-Layer Perzeptron (MLP) und bildet die einfachste Form eines neuronalen Netzes. Ist jedes Neuron mit jedem Neuron aus den benachbarten Schichten verbunden, spricht man auch von einer vollständig verbundenen (fully connected) Schicht oder auch *Dense Layer*.

3.3. Arten von Netz-Schichten

Künstliche neuronale Netze lassen sich aus verschiedenen Arten von Schichten aufbauen. Die Eingabedaten werden über mehrere Schichten verarbeitet und weitergegeben.

Convolution Bei der Convolution-Schicht wird ein Filter, oft auch als Kernel bezeichnet, über die Eingabedaten angewendet, wie in Abbildung 3.1 dargestellt. Dabei werden die Pixel

in der Quelle mit dem jeweiligen Filterwert multipliziert und anschließenden aufsummiert und in die Ausgabe geschrieben. Danach wird der Filter um eine Position verschoben und erneut angewendet. Je nach verwendeten Filter kann man spezielle Merkmale, wie zum Beispiel Kanten, aus den Eingabedaten hervorheben.

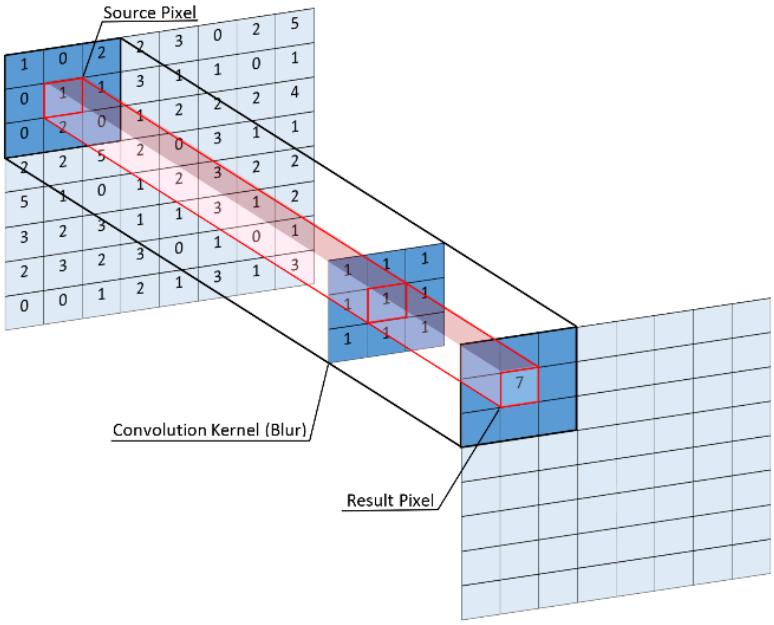

Abbildung 3.1.: Prinzip einer Convolution-Schicht mit einem Filter der Größe 3x3 [16]. Jeder Pixel im Ergebnis berechnet sich aus der Summe der Produkte der Pixel in der Quelle und der Filtermaske.

Um die Größe der Eingabedaten auch in den Ausgabedaten beizubehalten, muss bei mehrdimensionalen Filtern der Rand erweitert werden. Dieses sogenannte *Padding* erfolgt oft mit Nullen oder durch Wiederholung der Randziffern.

Pooling Um die Größe der Daten zur reduzieren, werden Pooling Schichten verwendet. Dabei wird versucht die wichtigen Informationen aus den Daten zu extrahieren.

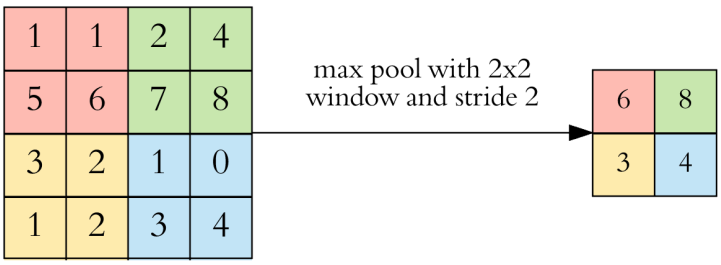

Abbildung 3.2.: Prinzip einer Pooling Schicht mit einem Fenster der Größe 2x2 und einer Schrittweite von 2 [12]. In die Ergebnispixel wird der maximale Wert aus den Pixeln der Quelle für das jeweilige Auswahlfenster eingetragen.

Wie in Abbildung 3.2 zu sehen ist, wird bei einer Max-Pooling Schicht für jedes Auswahlfenster der maximale Wert in die Ausgabe gespeichert und das Auswahlfenster immer um 2 Positionen je Richtung verschoben.
Sehr ähnlich zum Max-Pooling ist das Average-Pooling. Ausgegeben wird der Mittelwert aller Werte im Auswahlfenster.

Reduce Maximum Um die Dimension einer Matrix zu reduzieren gibt es verschiedene Verfahren. Eine davon ist Reduce Maximum, bei der lediglich das maximale Element einer Dimension in die Ausgabe übernommen wird.

L2-Normalisierung Die L2-Normalisierung verkürzt die Länge eines Vektors x mit Hilfe der Quadrate seiner Werte bei Beibehaltung seiner Richtung auf die Länge 1:

$$\vec{x} = \frac{\vec{x}}{\sqrt{\sum_i x_i^2}} \tag{3.4}$$

Für die bessere Vergleichbarkeit der Richtungsvektoren untereinander wird dies genutzt.

Flatten Beim Flatten werden mehrdimensionale Eingabedaten auf eine Dimension reduziert, indem alle Einträge hintereinander in einen Vektor eingefügt werden. Dies ist vor vollständig verbundenen Schichten notwendig, da diese einen eindimensionalen Vektor als Eingabe benötigen, während Convolution-Schichten multidimensionale Ausgaben erzeugen [12].

3.4. Training künstlicher neuronaler Netze

Die Delta-Regel setzt voraus, dass man den erwarteten Ausgabewert für jedes Element kennt. Bei mehrschichtigen Netzen ist dies nur für die Ausgabeschicht gegeben. Daher muss die Regel für mehrschichtige Netze als sogenanntes Backpropagation-Verfahren abgewandelt werden. Dazu wird eine Fehlerfunktion berechnet, die die Abweichung $y_i - \hat{y}_i$ für alle Trainingsdaten angeben soll. Eine mögliche Fehlerfunktion für die Regression von n Trainingsdaten ist die des mittleren quadratischen Fehlers E [13]:

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.5}$$

Backpropagationverfahren Beim Gradientenverfahren wird mit Hilfe der partiellen Ableitung der Fehlerfunktion ein globales Minimum des Fehlers gesucht, um eine möglichst genaue Approximation des zu erlernenden Verhaltens zu erreichen. Dazu werden mit dem Backpropagation-Verfahren die Gewichte w_i des Netzes angepasst [13]:

$$w_{i_{\text{new}}} = w_{i_{\text{old}}} - \eta \cdot \frac{1}{M} \sum_{k=1}^{M} \frac{\partial E}{\partial w_i} \tag{3.6}$$

Die Lernrate η gibt dabei die Geschwindigkeit der Gewichtsanpassung vor und die Batchgröße M die Zahl der zu trainierenden Ereignisse, bevor eine Gewichtsanpassung mit der gemittelten partiellen Ableitung des Fehlers E im Bezug auf das Gewicht w_i stattfindet. Dabei wird die Anpassung der Gewichte an der letzten Schicht begonnen, weil diese unmittelbar mit den fehlerhaften Ausgabedaten verbunden ist und dann schrittweise in Richtung der Eingabeschicht fortgesetzt. Beim Backpropagation-Verfahren wird die Verantwortung für den Fehler zwischen den beteiligten Gewichten aufgeteilt [23]. Werden jeweils mehrere Trainingsdaten vor einer Gewichtsanpassung betrachtet, so reduziert dies größere Schwankungen und erhöht somit die Robustheit des Trainings.
Da das Gradientenverfahren statt in ein globales Minimum auch in einem lokalen Minimum enden könnte, gibt es mehrere Algorithmen, die das Verfahren erweitern, um ein solches Verhalten zu verhindern. Einer davon ist der Momentum-Algorithmus, bei dem eine Linearkombination aus vorheriger Anpassung und dem Gradienten zur Anpassung der Gewichte hinzugefügt wird, um eine Oszillation um ein lokales Minimum zu verhindern und den Impuls in eine Richtung beim Gradientenabstieg beizubehalten [28].

3.5. Overfitting und Dropout

Overfitting tritt auf, wenn ein Modell die Menge der Trainingsdaten sehr gut beschreiben kann, aber nicht generell auf neue Eingabedaten anwendbar ist. Dies lässt sich überwachen, indem nach jeder Trainingsiteration ein Validationsschritt durchgeführt wird, bei dem die Verlustfunktion für einen Datensatz berechnet wird, der nicht Bestandteil des Trainings war. Erhöht sich die Ausgabe der Verlustfunktion für mehrere Trainingsdurchläufe hintereinander, so ist dies ein Hinweis auf Overfitting. Overfitting lässt sich zum einen durch ein rechtzeitiges Beenden des Trainings, zum anderen durch Regularisierung verhindern. Eine Art der Regularisierung sind geringfügige Änderungen an den Trainingsdaten,

wie beispielsweise Drehungen, Vergrößerungen, Verkleinerungen oder Verschiebungen von einzelnen Daten.
Eine weitere Möglichkeit der Regularisierung ist der Dropout. Beim Dropout wird ein zufälliger Teil der Neuronen aus der Berechnung der Ausgabe und dem anschließenden Backpropagation-Prozess ausgeschlossen, um die Robustheit des Netzes gegenüber Overfitting zu verbessern [23]. Dabei wird zudem das Training beschleunigt. Der Anteil an auszulassenden Neuronen lässt sich vorgeben.

3.6. Einsatz von neuronalen Netzen

Neuronale Netze werden oft im maschinellen Lernen zur Verarbeitung von Bild-, Audio- oder Sensordaten eingesetzt. Sie eignen sich hier besonders gut, da sie beim Trainieren lernen, Muster zu erkennen und auf neue Kontexte anzuwenden.
Typische Aufgabentypen für neuronale Netze sind die Klassifikation, bei der die Klassenzugehörigkeit von Eingabedaten hervorgesagt werden soll, und die Regression, bei der für die Eingabedaten eine kontinuierliche Ausgabe erzeugt werden soll [13].

4. Verwandte Arbeiten

In diesem Kapitel werden die bisherigen Arbeiten auf dem Gebiet der Richtungsrekonstruktion in Flüssigszintillatordetektoren und dem Training von neuronalen Netzen auf Punktmengen vorgestellt.

4.1. Bisherige Arbeiten zur Richtungsrekonstruktion in Neutrino-Detektoren

Große Flüssigszintillatordetektoren haben bisher schon einen großen Betrag zum Verständnis der Eigenschaften von Neutrinos beigetragen und werden auch in Zukunft weiter dafür eingesetzt werden. Dabei sind sie im Vergleich zu Wasser-Čerenkov-Detektoren besser skalierbar, weil sie dabei eine um Faktor zwei bessere Energieauflösung erreichen [2]. Allerdings können bei Flüssigszintillatordetektoren für Teilchen mit Energien im niedrigen Mega-Elektronenvolt-Bereich keine ausreichenden Informationen aus dem Szintillationslicht über die Spur der Teilchen gewonnen werden. C. Aberle und andere haben in ihrem Artikel [2] ein Verfahren entwickelt, um für Flüssigszintillatordetektoren die Teilchenrichtung mithilfe der unterschiedlichen Ankunftszeiten von Čerenkovlicht und Szintillationslicht zu extrahieren. Aufgrund der physikalischen Prozesse bei der Lichtentstehung wird das Čerenkovlicht direkt ausgesendet, während das Szintillationslicht leicht zeitverzögert emittiert wird. Mehr zu den physikalischen Prozessen findet sich in Abschnitt 2.3.

Richtungsrekonstruktion mit neuronalen Netzen in JUNO Ein neuronales Netz zur Richtungsrekonstruktion von Niedrigenergie-Ereignissen in JUNO wurde bereits von Yaping Cheng und anderen entworfen und auf 200.000 Ereignissen mit 4 Mega-Elektronenvolt (MeV) Energie trainiert [9]. Bei dieser Arbeit wurden alle Ereignisse in der Detektormitte simuliert und alle Zeiten unterhalb von 98,25 Nanosekunden betrachtet.

Für die Positionen der Photomultiplier Tubes wurde der Detektor in 111 Ringe mit maximal 226 Photomultiplier Tubes aufgeteilt. Die Auftreffzeiten wurden im neuronalen Netz nicht verwendet. Dabei konnte mit idealen Daten ohne Zeitverschmierung an den Photomultiplier Tubes erreicht werden, dass für 68% der Testereignisse (1σ) die maximale Abweichung der rekonstruierten Richtung von der tatsächlichen Richtung des Ereignisses 38,99° beträgt. Unter der Annahme einer Zeitverschmierung von einer Nanosekunde an den Photomultiplier Tubes betrug dort der 1σ-Winkel 56,27° [9].

4.2. Verwandte Arbeiten zu neuronalen Netzen auf Punktmengen

Die Anwendung von Deep Learning auf Punktmengen für die 3D-Klassifikation ist eine oft benötigte Methode, gleichzeitig wird dies oft nur über Netze von 3D-Voxeln oder eine Sammlung von 2D-Projektionen realisiert. Dabei gehen Informationen über die Daten verloren.

PointNet Ein Ansatz zum Arbeiten direkt auf den einzelnen Punkten ist das Netz PointNet [26]. Dazu wird jeder Punkt durch seine dreidimensionalen Koordinaten (x_i, y_i, z_i) dargestellt. Zusätzlich können noch weitere Koordinaten hinzugefügt werden, beispielsweise für Farbe oder Oberflächennormale. Dabei haben die Punkte keine spezielle Reihenfolge. Somit muss das Netzwerk invariant gegenüber Permutationen in den Daten sein.
Wie in Abbildung 4.1 zu erkennen ist, werden die dreidimensionalen Eingabepunkte zunächst in einem Transformationsnetz vorbehandelt. Anschließend berechnet das Netzwerk die folgenden Schritte auf jedem Punkt einzeln, aber mit den gleichen Kantengewichten für alle Punkte. Dies sorgt dafür, dass die Ergebnisse invariant gegenüber Rotationen in den Ausgangsdaten sind. Die folgenden Schritte sind ein Multi-Layer-Perzeptron (MLP), danach eine Merkmalstransformation mit einem separaten Mininetz und ein abschließendes Multi-Layer Perzeptron, bevor mit einer Max-Pooling-Schicht die globalen Merkmale extrahiert werden. Die beiden Anwendungsfälle von PointNet sind die Klassifikation von Objekten und die Segmentation von Teilobjekten oder semantischen Zusammenhängen. Datengrundlage sind in der Regel Punktmengen von Objekten oder Räumen, die beispielsweise mit Hilfe von 3D-Laserscans erstellt werden.

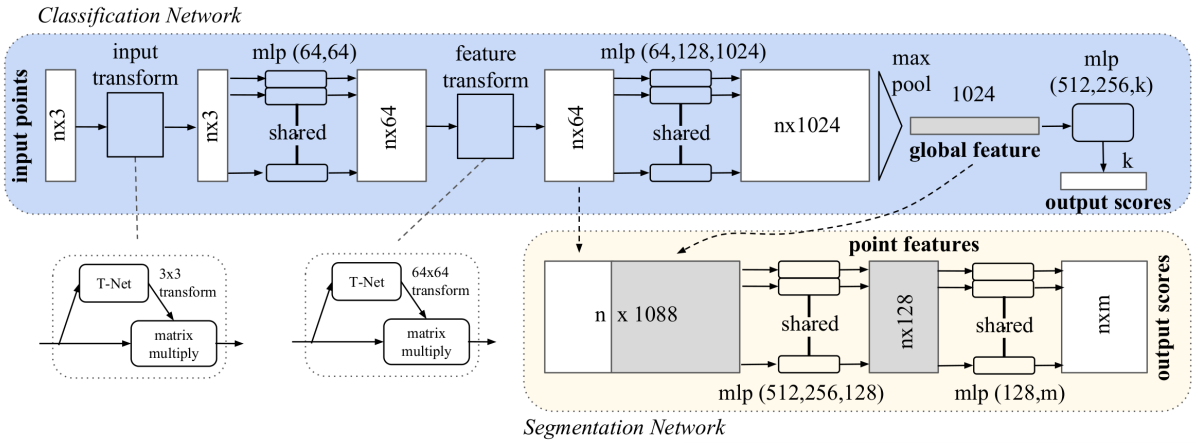

Abbildung 4.1.: Aufbau des PointNet Netzwerkes mit dem Klassifikationsnetz in blau und dem zusätzlichen Segmentationsnetz in gelb. Das Klassifikationsnetz führt auf den Eingabepunkten verschiedene Transformationen aus, berechnet nach jeder Transformation ein Multi-Layer-Perzeptron mit gemeinsamen Kantengewichten und aggregiert anschließend die Merkmale über ein Max-Pooling. Die Segmentation ist eine Erweiterung um zwei weitere Multi-Layer-Perzeptrone mit der Verbindung von globalen und lokalen Merkmalen der Punkte als Eingabedaten [26].

Als Ausgabe gibt das Netzwerk bei der Klassifikation für jede mögliche Klasse einen Wert zurück, der die Wahrscheinlichkeit beschreibt, dass es sich bei dem betrachteten Objekt um ein Objekt dieser Klasse handelt. Bei der Segmentation wird für jeden Punkt und für jede mögliche Unterkategorie ein Wert zurückgegeben, der die Wahrscheinlichkeit beschreibt, dass dieser Punkt zu jener Kategorie gehört.
Das Netzwerk ist in der Lage die Kontur der Objekte anhand einer kleinen Menge an Schlüsselpunkten zusammenzufassen. Im Vergleich zu Netzwerken mit Volumenrepräsentation oder einer Menge an verschiedenen zweidimensionalen Projektionen konnte eine höhere Effizienz in Hinsicht auf die Rechenkosten erreicht werden.

Dynamic Graph CNN Ein auf PointNet aufbauender Ansatz ist der des Dynamic Graph CNN [30]. Dabei werden benachbarte Punkte zu einem Graph verbunden und die Convolution-Operationen auf den Kanten ausgeführt. Dadurch werden Informationen über die Nachbarn in die Berechnung eines Punktes eingeschlossen und so die geometrischen Strukturen berücksichtigt. Im Gegensatz zu Graph Convolutional Neural Networks ist der Graph nicht von Anfang an festgelegt, sondern wir nach jeder Netzwerkschicht dynamisch aktualisiert. Dabei wird für jeden Punkt der gerichtete Graph der k nächsten Nachbarn im Merkmalsraum berechnet, das jeweilige Kantenmerkmal berechnet und anschließend eine symmetrische Aggregationsfunktion, wie Summe oder Maximum, auf Kantenmerkmalen

der verbundenen Nachbarn ausgeführt. Die Autoren bezeichnen diese Methode als Edge Convolution (EdgeConv) [30].

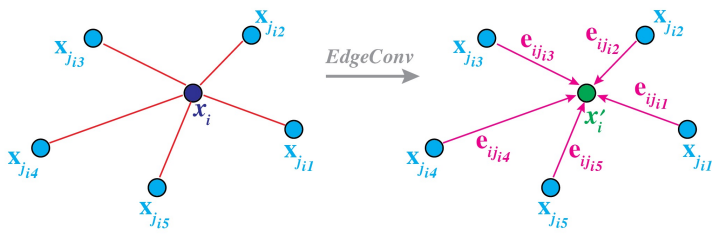

Abbildung 4.2.: Schema der *EdgeConv*-Operation. Die Aggregation der Kantenmerkmale e_{ij} ergibt in Verbindung mit den Kanten aller verbundenen Knoten die Ausgabe der EdgeConv-Operation [30].

Der schematische Ablauf der EdgeConv-Operation ist in Abbildung 4.2 dargestellt. Für jeden Punkt x_i wird die Ausgabe x'_i aus den Kantenmerkmalen e_{ij} der verbundenen Nachbarn x_j errechnet. Das Kantenmerkmal ist eine nicht-lineare Funktion mit einer Menge an lernbaren Parametern. Die Analogie zur Convolution erfolgt, indem man x_i als zentralen Pixel und die verbundenen Nachbarn als Patches betrachtet.
Eine zentrale Eigenschaften von PointNet, die Permutationsinvarianz, bleibt auch bei Dynamic Graph CNN erhalten, da die Aggregation über symmetrische Operationen erfolgt.

5. Simulation und Aufbereitung der Trainingsdaten

Für das Training und die Evaluation können mithilfe einer Simulationssoftware Datensätze von möglichen Elektronenereignissen im Detektor erzeugt werden. Da die Daten in kartesischen Koordinaten vorliegen und benachbarte Punkte in die Berechnung der Richtung mit einbezogen werden sollen, bietet es sich an, auf dem existierenden Netz „Dynamic Graph CNN“ [30] aufzubauen. Durch die Einbeziehung benachbarter Punkte wird das Ziel verfolgt, Punkte, die nah beieinander sind, in der Richtungsrekonstruktion höher zu gewichten. Die Eingabedaten werden bei diesem Netz aus Dateien im HDF5-Format gelesen. Um die Daten für das Training mit Dynamic Graph CNN nutzen zu können, müssen sie daher aus den ROOT-Dateien ausgelesen, die PMT-Nummern durch die PMT-Positionen ersetzt werden und in HDF5-Dateien gespeichert werden.

5.1. Datensimulation

Im ersten Schritt wurden die Daten von 100.000 Ergebnissen von Elektronen mit 3 MeV für das Training und 10.400 Ereignisse mit 4 MeV je zur Hälfte für Validation und Evaluation simuliert. Die Ereignisse wurden alle in der Mitte des Detektors mit verschiedenen Flugrichtungen simuliert. Diese Daten liegen im ROOT-Datenformat [8] vor. Für die Simulation wurde die Detektorsimulation aus der JUNO-Software in der Version J18v1r1 (Revision 3547M vom 23. Juli 2019) verwendet. Zum weiteren Training wurden zudem 110.000 Ereignisse mit einer Energie von 3 MeV entlang der Z-Achse bis nah an den Detektorrand heran simuliert. Bei den Trainingsdaten wurde die Flugrichtung der Elektronen jeweils zufällig gewählt, bei den Daten für Validation und Evaluation wurden in 26 verschiedene, gleichmäßig verteilte Richtungen jeweils 400 Ereignisse simuliert. Für die Evaluation wurden jeweils 5200 Elektron-Ereignisse mit einer Energie von 1 bis 8 MeV

in der Mitte des Detektors und für 4 MeV im Abstand von 50 cm, 1 m, 5 m und 10 m vom Mittelpunkt entlang der Z-Achse erzeugt. Die Energie entspricht dem typischen Energiebereich von Elektronen, die aus Stößen mit solaren Elektronneutrinos (1-5 MeV) und Supernova-Neutrinos mit Energie auch oberhalb von 5 MeV stammen. Ziel ist es, die Rekonstruktion auf eine energieunabhängige Signatur zu trainieren.

5.2. Datenaufbereitung

Da die Richtungsinformation ausschließlich aus dem Čerenkovlicht gewonnen werden kann, sollen die Daten auf diese Signale reduziert werden. Wie in Abbildung 5.1 zu sehen ist, trifft das erste Čerenkovlicht schon kurz vor der mittleren erwarteten Flugzeit auf den Photomultiplier Tubes auf, während das Szintillationslicht zeitverzögert auftrifft. Dies soll genutzt werden, um den Anteil der Čerenkovphotonen am Gesamtsignal zu maximieren. Die Auftreffzeiten in dem PMTs lassen sich als Punkt-Wolke mit der Position des PMTs und der Zeit darstellen.

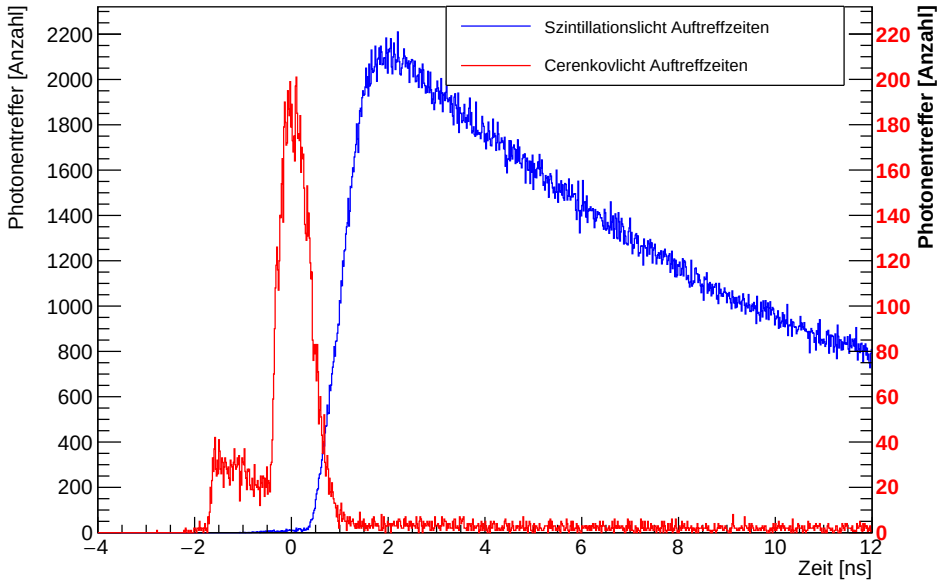

Abbildung 5.1.: Differenz in der Auftreffzeit nach Zeitkorrektur zwischen Čerenkov- und Szintillationsphotonen für 400 Events mit 4 MeV in der Detektormitte. Die Čerenkovphotonen (rote Kurve) treffen ab etwa 2 Nanosekunden vor den durch Szintillation erzeugten Photonen (blaue Kurve) auf die PMTs auf. Die Achseneinteilung auf der rechten Seite gilt für die Čerenkovphotonen und ist um den Faktor 10 vergrößert.

Die simulierten Datensätze wurden mit Hilfe eines in C++ geschriebenen ROOT-Skriptes ausgelesen und mit Hilfe einer Nachschlagetabelle (*engl. lookup-table*) wurde für alle Photonen die mittlere erwartete Flugzeit vom Vertex zum PMT errechnet und von der gemessenen Zeit abgezogen und so eine Zeitkorrektur anwendet. Die Nachschlagetabelle enthält für alle im Detektor möglichen Winkel und Distanzen zwischen Vertex und PMT-Position die mittlere Flugzeit für Szintillationsphotonen. Durch die Zeitkorrektur werden Zeitunterschiede in der Ankunftszeit ausgeglichen, die nur durch die Vertexposition entstehen. Somit muss hierfür die Vertexposition bekannt sein. Hierzu wird die Vertexposition aus der Monte-Carlo-Wahrheit der Simulation verwendet.
Anschließend wurden die korrigierte Zeit, die PMT-Nummer, die Vertexposition und die Flugrichtung als Textdatei abgespeichert. In einem zweiten Schritt wurden die Textdateien in einem Pythonskript eingelesen und die PMT-Nummer durch die Position des PMTs in kartesischen Koordinaten ersetzt. Die modifizierten Daten werden dann als Binärdateien im Hierarchical Data Format (HDF5) mit Hilfe der Pythonbibiliothek h5py [10] abgespeichert. Das HDF5-Format wurde für die effiziente Speicherung großer Datenmengen entwickelt. Dabei können mehrere Datensätze in einer Datei gespeichert werden [29].

6. Richtungsrekonstruktion mithilfe künstlicher neuronaler Netze

Mit Hilfe von TensorFlow wird das künstliche neuronale Netz „Dynamic Graph CNN“ angepasst und auf den Daten aus der Detektorsimulation trainiert. Dabei werden die einzelnen Modifikationsschritte auf ihre Auswirkung auf das Ergebnis evaluiert.

6.1. TensorFlow

Als Framework zum Trainieren des künstlichen neuronalen Netzes wurde TensorFlow [1] in den Versionen 1.13.1 (nur CPU) und 1.14 (GPU) verwendet. Für die GPU Variante wurde Grafikkarte NVIDIA GeForce GTX 980 mit CUDA [22] in der Version 10.1 genutzt. TensorFlow wurde vom Google Brain Team als Open-Source-Bibliothek für das Deep Learning entwickelt und wird durch eine aktive Community stetig weiterentwickelt [13]. Über mehrere Ebenen von Programmierschnittstellen lassen sich künstliche neuronale Netze bauen und trainieren. Die unterste Ebene ist der in der Programmiersprache C++ geschriebene Kernel, der die Befehle auf die einzelnen Hardwarekomponenten, wie Grafikkarte und Prozessor weiterreicht. Die erste Programmierschnittstelle darüber bilden die grundlegenden Datentypen wie Tensoren, Variablen und Graphen, die sich in ein Skript der Programmiersprache Python [25] einbinden lassen, über das sich TensorFlow steuern lässt. Außerdem gibt es in der darüber liegenden Ebene vorgefertigte Schichten, wie Convolution und Pooling (siehe auch Abschnitt 3.3), sowie Methoden zum Training, aus denen die neuronalen Netzen zusammengesetzt werden können. Als oberste Ebene gibt es noch vereinfachte Methoden, die mehrere Aufrufe der mittleren Ebene der Programmierschnittstellen zusammenfassen und ganze Netze für bestimmte Aufgaben bereits zur Verfügung stellen [13].

6.2. Evaluierung der Richtungsrekonstruktion

Für die Ereignisse der Evaluationsdatensätze werden die Richtungsvektoren vorhergesagt. Anschließend wird die Verteilung des Winkels zwischen dem Wert aus der Monte-Carlo-Wahrheit und vorhergesagtem Wert berechnet und der 1σ-Winkel berechnet, wobei 1σ definiert ist als der Winkel, der 68% der Richtungsrekonstruktionen enthält. In Abbildung 6.1 ist die Winkelverteilung der Abweichung von der Monte-Carlo-Wahrheit für die Spurrichtungsrekonstruktion auf dem Evaluationsdatensatz bei einer Teilchenenergie von 4 MeV dargestellt.

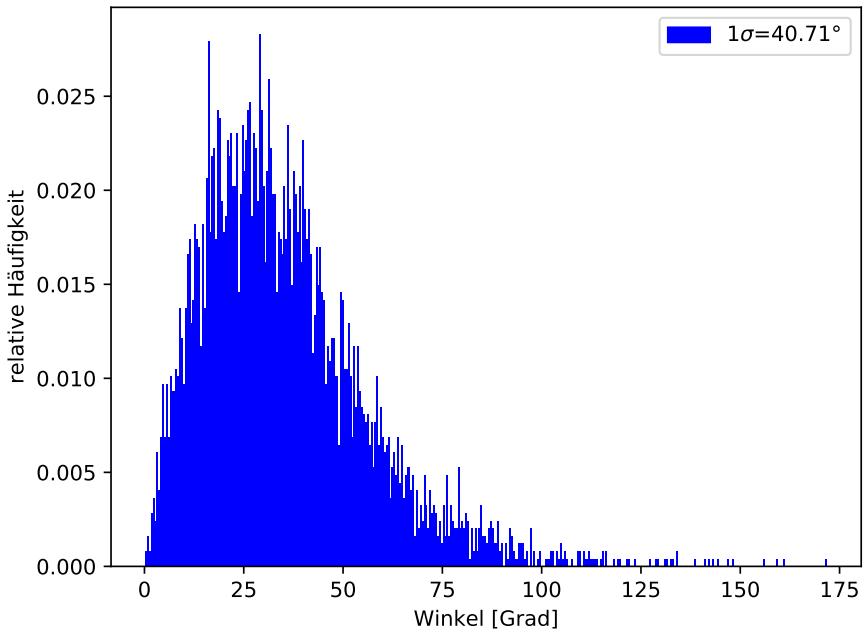

Abbildung 6.1.: Winkelverteilung für die Spurrichtungsrekonstruktion für eine Teilchenenergie von 4 MeV.

6.3. Anpassung des neuronalen Netzes

Gegenüber dem Dynamic Graph CNN Netz, müssen einige Schichten ausgetauscht werden. Dabei wird sich an der Netzstruktur des Netzes von [9] orientiert (siehe Abbildung 6.2). Auf das Dynamic Graph CNN Netz übertragen, ergibt sich dabei eine leicht angepasste Reihenfolge der Schichten mit zunächst einer Berechnung einer Adjazenzmatrix mit den paarweisen Abständen der einzelnen Datenpunkte. Daraus werden die k nächsten Nachbarn bestimmt und eine Kantenabstraktion (*engl. Edge-Feature*) berechnet.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 111, 226, 16)      416
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 37, 76, 16)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 37, 76, 16)        6416
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 13, 26, 16)        0
_________________________________________________________________
conv3 (Conv2D)               (None, 13, 26, 16)        6416
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 5, 9, 16)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 9, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 720)               0
_________________________________________________________________
local4 (Dense)               (None, 384)               276864
_________________________________________________________________
local5 (Dense)               (None, 192)               73920
_________________________________________________________________
dropout_2 (Dropout)          (None, 192)               0
_________________________________________________________________
prediction (Dense)           (None, 3)                 579
=================================================================
Total params: 364,611
Trainable params: 364,611
Non-trainable params: 0
```

Abbildung 6.2.: Convolutional-Neural-Network mit drei Convolution Layern mit jeweils anschließendem Max-Pooling, danach ein Dropout-Layer, ein Flatten-Layer, zwei Dense-Layern, einem weiteren Dropout und einem abschließenden Dense-Layer, wie es als Vorlage genutzt wurde [9]. In der ersten Spalte ist jeweils der Name und die Art der Schicht, in der zweiten Spalte die Größe der Ausgabematrix mit „None“ als Platzhalter für die Anzahl parallel berechneter Ereignisse und in der dritten Spalte die Anzahl der trainierbaren Parameter angegeben.

Name	**Type**	**Output-Shape**
tconv1	Conv2D	(None, 512, 8, 64)
tconv2	Conv2D	(None, 512, 8, 128)
reduce_max	Reduce_max	(None, 512, 1, 128)
tconv3	Conv2D	(None, 512, 1, 1024)
tmaxpool	MaxPool2D	(None, 1, 1, 1024)
tfc1	Fully_connected	(None, 512)
tfc2	Fully_connected	(None, 256)
transform_XYZ/weights	MatMul	(None, 16)
transform_XYZ/biases	BiasAdd	(None, 16)

Tabelle 6.1.: Transformationsnetz, um eine Reihenfolgeunabhängigkeit der Eingabedatenpunkte zu erreichen. Das Netz besteht aus zwei Convolution-Schichten, einer Dimensionsreduktion mit Maximum, einer weiteren Convolution-Schicht, einem Max-Pooling, zwei vollständig verbundenen Schichten und einer Matrixmultiplikation mit Gewichten und die Addition eines konstanten Wertes.

Anschließend wird das Transformationsnetz, wie in Tabelle 6.1 dargestellt, auf das Edge-Feature angewendet und eine erneute Adjazenzmatrix und ein erneutes Edge-Feature berechnet. Die so vorberechnete Menge an Datenpunkten wird dann in das Convolution-Netz, die einzelnen Schichten sind in Tabelle 6.2 dargestellt, gegeben und der Richtungsvektor berechnet. Das neuronale Netz gibt für jede Punktmenge den kartesischen Richtungsvektor aus, normiert auf die Länge 1. Im Gegensatz zum „Dynamic Graph CNN“ werden die Reduce-Maximum-Schichten durch Max-Pooling-Schichten ersetzt und zwischen den einzelnen Convolution Schichten keine neue Adjazenzmatrix und keine Edge-Feature berechnet.

Name	**Type**	**Output-Shape**
dgcnn1	Conv2D	(None, 512, 8, 16)
pool1	MaxPool2D	(None, 255, 4, 16)
dgcnn2	Conv2D	(None, 255, 4, 16)
pool2	MaxPool2D	(None, 127, 2, 16)
dgcnn3	Conv2D	(None, 127, 2, 16)
pool3	MaxPool2D	(None, 63, 1, 16)
dp1	Dropout(0.5)	(None, 63, 1, 16)
flatten1	Flatten	(None, 1008)
local1	Dense	(None, 320)
local2	Dense	(None, 160)
dp2	Dropout(0.5)	(None, 160)
prediction/dense	Dense	(None, 3)
prediction/l2_normalize	L2 normalisation	(None, 3)

Tabelle 6.2.: CNN-Version für Punktwolken des CNNs von Abbildung 6.2. Das Netz besteht aus den drei Convolution Schichten mit Max Pooling und zwei vollständig verbundenen Schichten, sowie zwei Dropout Schichten mit einer Rate von 0,5 vor und hinter den vollständig verbundenen Schichten und einer abschließenden Vorhersage eines normierten Richtungsvektors aus einer vollständig verbundenen Schicht und einer Normierung der Ausgabe.

6.3.1. Anpassung der Verlustfunktion

Die Verlust-Funktion (*engl. loss-function*) wurde von einer für Klassifikationen, und damit diskrete Werte, gut geeigneten Funktion (Softmax Cross Entropy) auf eine Regressionsfunktion für die Abweichung zwischen zwei Richtungsvektoren (Cosine Distance) angepasst. Die relative Genauigkeit der Vorhersage (*engl. accuracy*) wird über die Anzahl der korrekt vorhergesagten Richtungen berechnet. Eine Vorhersage wird dann als korrekt angenommen, wenn der Winkel zwischen Vorhersage und Richtung aus dem Monte-Carlo der Simulation unter 30° beträgt.

6.3.2. Vergrößerung der Tiefe des Netzes

Im Laufe der Masterarbeit wurde das Netz aus Tabelle 6.2 weiter optimiert, indem zunächst eine weitere vollständig verbundene Schicht (local3) mit 80 Ausgabeneuronen als letzte Schicht vor dem zweiten Dropout eingefügt wurde.

In Abbildung 6.3 ist eine Vergleichsmessung der Varianten zu sehen. Dabei sind die Ergebnisse für die ursprüngliche Form in blau dargestellt und die erste Erweiterung in orange. Vergleicht man den 1σ-Fehler für Elektronen-Energien zwischen 1 und 8 MeV, so reduziert sich dieser von ca. 90° auf ca. 50° für 8 MeV. Als Datensatz für das Training wird in diesem Abschnitt ein Datensatz mit allen 210.000 Trainingsereignissen mit je 512 Punkten und einem Zeitschnitt von 3 Nanosekunden verwendet. Für die Vergleichsmessungen werden jeweils 5200 Ereignisse in der Detektormitte mit einer Energie von 1 MeV bis 8 MeV und einen Zeitschnitt von 2,75 Nanosekunden verwendet. Die Position des Vertexes wird in den Daten mit einer Gauß-Funktion mit einer Standardabweichung von 10 cm verschmiert, um realistischere Ergebnisse zu erzeugen. Die Abstände der getroffenen Photomultiplier Tubes zum Vertex werden auf die korrigierte Zeit reduziert und keine Streuung der Flugzeiten innerhalb der PMTs berücksichtigt.

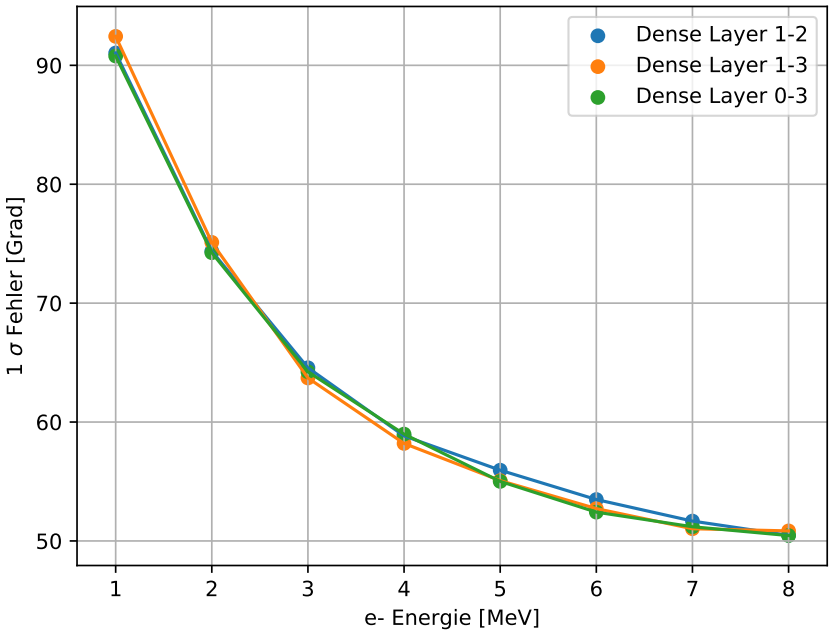

Abbildung 6.3.: 1 σ-Fehler der Spurrichtungsrekonstruktion für Teilchenenergien zwischen 1 und 8 MeV im Vergleich für zwei, drei und vier vollständig verbundene Schichten

In einem weiteren Schritt wurde eine weitere vollständig verbundene Schicht (local0) mit 640 Ausgabeneuronen vor local1 eingefügt. Die Messung hierzu ist in Abbildung 6.3 als grüne Linie eingezeichnet und bietet gegenüber der Version mit nur 3 vollständig verbundenen Schichten eine Verbesserung der Genauigkeit der Spurrichtungsrekonstruktion für Energien zwischen 4 und 6 MeV um etwa 1°.

6.3.3. Variation der Dropout-Rate

Zusätzlich wurde eine Variation der Dropout-Rate von 0,5 untersucht. Dabei wurde, wie in Abbildung 6.4 dargestellt ist, festgestellt, dass eine reduzierte oder erhöhte Dropout-Rate um $\pm\,0,1$ eine 2° bis 3° bessere Genauigkeit herbeiführt. Eine auf 0,6 erhöhte Dropout-Rate ist hierbei nochmal ein wenig besser als eine Dropout-Rate von 0,4.

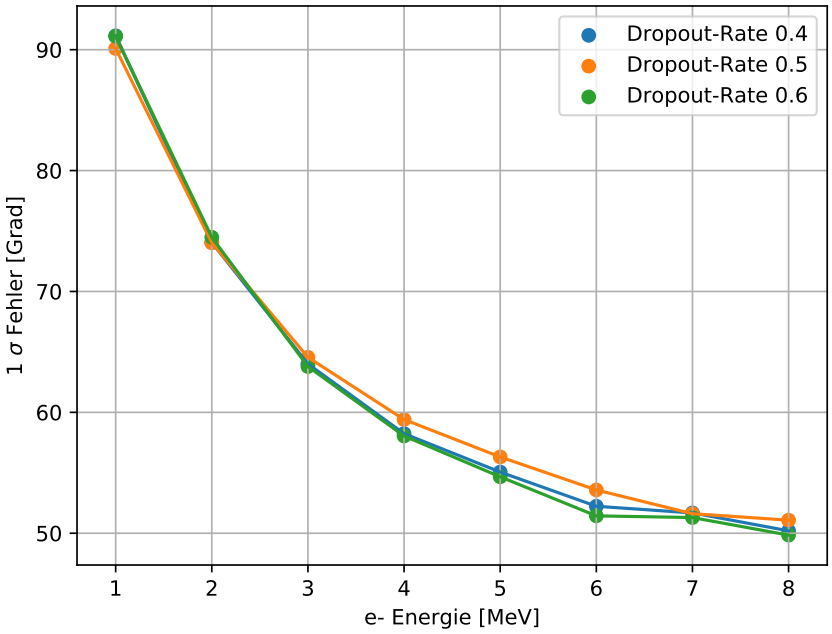

Abbildung 6.4.: 1 σ-Fehler der Spurrichtungsrekonstruktion für Teilchenenergien zwischen 1 und 8 MeV im Vergleich für eine Dropout-Rate von 0,4 , 0,5 und 0,6

6.3.4. Austausch des Pooling-Verfahrens

Außerdem wurde auch getestet, welchen Einfluss ein Ersetzen des Max-Pooling durch ein Average-Pooling hätte. Wie in Abbildung 6.5 zu sehen ist, hat das Average-Pooling dabei gegenüber dem Max-Pooling keinen Vorteil, sondern ist für einzelne Messwerte sogar leicht schlechter.

Name	Type	Output-Shape
dgcnn1	Conv2D	(None, 512, 8, 16)
pool1	MaxPool2D	(None, 255, 4, 16)
dgcnn2	Conv2D	(None, 255, 4, 16)
pool2	MaxPool2D	(None, 127, 2, 16)
dgcnn3	Conv2D	(None, 127, 2, 16)
pool3	MaxPool2D	(None, 63, 1, 16)
dp1	Dropout(0.6)	(None, 63, 1, 16)
flatten1	Flatten	(None, 1008)
local0	Dense	(None, 640)
local1	Dense	(None, 320)
local2	Dense	(None, 160)
local3	Dense	(None, 80)
dp2	Dropout(0.6)	(None, 80)
prediction/dense	Dense	(None, 3)
prediction/l2_normalize	L2 normalisation	(None, 3)

Tabelle 6.3.: Von oben nach unten sind die nacheinander ausgeführten Schichten des modifizierten Netz zu sehen. Es besteht aus den drei Convolution-Schichten mit jeweils anschließenden Max-Pooling und vier vollständig verbundenen Schichten, sowie zwei Dropout-Schichten und einer abschließenden Vorhersage eines normierten Richtungsvektors. Für die einzelnen Schichten ist jeweils ein Name, die Art der Schicht und die Größe der Ausgabematrix angegeben. „None“ steht dabei für die Anzahl an parallel berechneten Ereignissen.

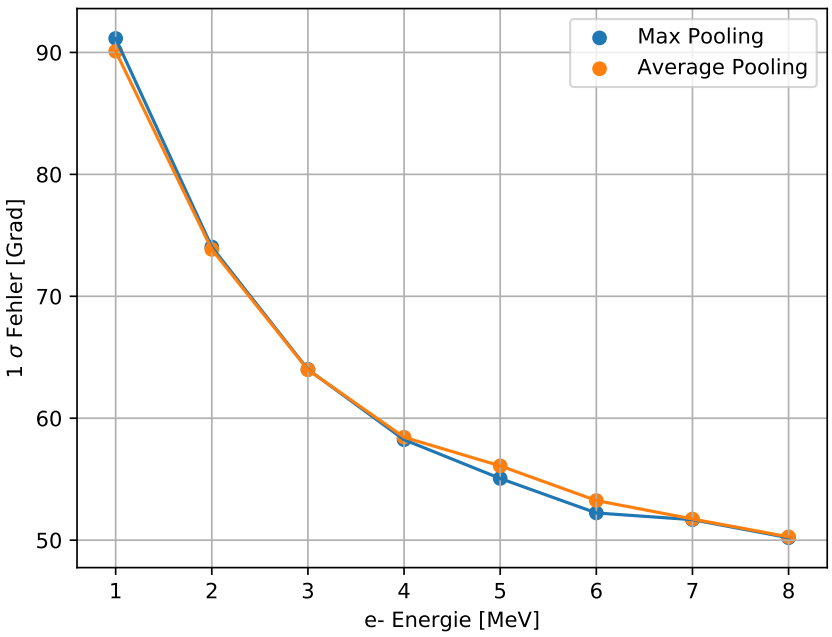

Abbildung 6.5.: 1 σ-Fehler der Spurrichtungsrekonstruktion für Teilchenenergien zwischen 1 und 8 MeV im Vergleich für Average Pooling und Max Pooling

Damit ergibt sich mit Erhöhung der Anzahl der vollständig verbundenen Schichten, der Erhöhung der Dropout-Rate auf 0,6 und Beibehaltung des Max-Poolings das Netz, dass in Tabelle 6.3 dargestellt ist.

6.4. Anpassung der Trainingsmethode

Im Vergleich zur Trainingsmethode des Dynamic Graph CNN wurden einige Parameter angepasst oder hinzugefügt, um das Training für den Anwendungsfall zu optimieren.

6.4.1. Vorzeitige Beendigung des Trainings

Um eine optimale Anzahl an Trainingsdurchläufen zu erreichen, wurde in die Trainingsfunktion zusätzlich ein Abbruchkriterium für das Training eingefügt, sobald sich keine Verbesserung mehr erzielen lässt. Dazu wird nach jedem Trainingsdurchlauf die durchschnittliche Genauigkeit für die Validationsereignisse bestimmt und mit der bisher besten Genauigkeit aus den vorherigen Durchläufen verglichen.

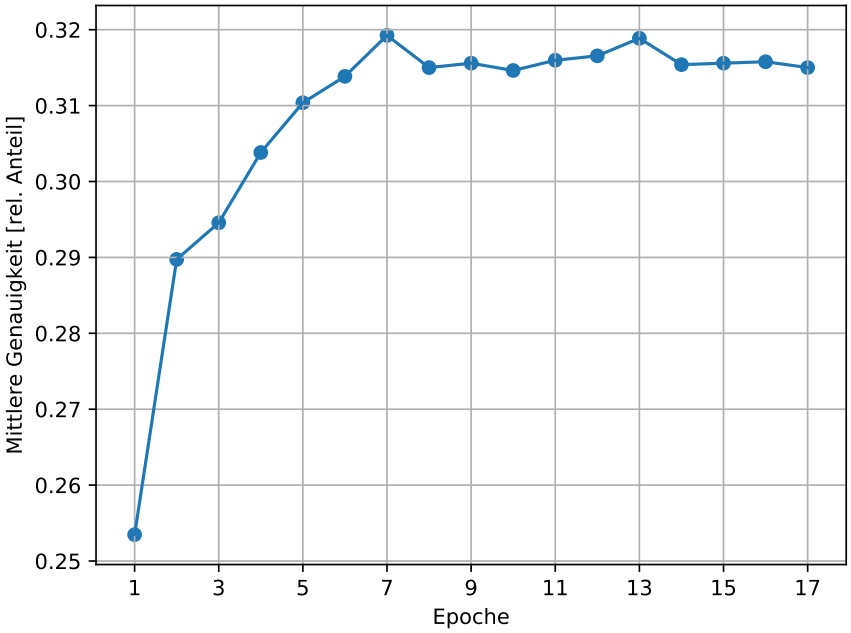

Abbildung 6.6.: Lernkurve für einen Trainingsdurchlauf des neuronalen Netzes. Der beste Wert für die mittlere Genauigkeit wird nach 7 Wiederholungen erreicht. Das Training wurde nach 17 Wiederholungen, also 10 ohne Verbesserung, abgebrochen.

Wenn sich die Genauigkeit über eine vorher festgelegte Anzahl an Wiederholungen nicht mehr verbessert, wird das Training beendet. In der Praxis hat sich eine Anzahl von 10 Wiederholungen ohne Verbesserung bewährt, da sich, wie in Abbildung 6.6 dargestellt, nach einigen schlechteren Durchläufen auch wieder bessere Werte wie in Epoche 13 ergeben können und noch kein Overfitting beobachtet werden kann. Zudem wird das Modell alle 5 Durchläufe gespeichert, um eine frühere Iteration für die Vorhersage der Richtungen nehmen zu können, falls das finale Modell doch schlechtere Ergebnisse liefern sollte oder das Training außerplanmäßig abbricht. Die maximale Anzahl an Wiederholungen wurde auf 52 begrenzt. In der Praxis wurde diese Zahl aber nicht erreicht, da bereits vorher sich keine Besserung der Genauigkeit mehr einstellte.

6.4.2. Verzicht auf Augmentation der Punktwolken

Da sich die Punktwolke der hier untersuchten Anwendung wesentlich von der ursprünglich vorgesehenen Form unterscheidet, wurde auf die bisher eingebauten Rotationen und Verschiebungen der Punkte verzichtet und dies direkt über die Modifikation der Daten durch Hinzufügen von Flugzeitstreuungen und Verschmierungen des Vertexes umgesetzt (siehe auch Abschnitt 6.6).

6.4.3. Trainingskonfiguration

Für das Training wurden immer 32 Ereignisse parallel (*engl. Batch*) berechnet, da dies mit der Anzahl an Punkten je Ereignis eine gute Ausnutzung des verfügbaren Grafikkartenspeichers darstellte. Die Parameter für die Lernrate und deren Anpassung sowie Momentum wurden von der Vorlage übernommen und entsprechen den in der Literatur verbreiteten Werten. Die Lernrate, die die Anpassung der Gewichte des Netzes beeinflusst, wurde zu Beginn auf 10^{-3} gesetzt und mit einem exponentiellen Zerfall mit einer Zerfallsrate von 0,7 alle 200.000 Ereignisse im Laufe des Trainings auf 10^{-5} reduziert. Als Optimierer wurde der Momentum-Optimierer mit einem Momentum von 0,9 gewählt. Um flexibler zwischen einzelnen Datensätzen wechseln zu können, wurde zudem jeweils ein Parameter für die Angabe der Datei mit den zu verwendenden Trainings- bzw. Evaluationsdatensätzen zum Kommandozeilenparser hinzugefügt.

6.5. Optimierung der Datenmodifikation

Nachdem das neuronale Netz angepasst wurde, wurde die Datenmodifikation (siehe auch Abschnitt 5.2) weiter optimiert, um die Qualität der Ergebnisse zu verbessern. Ziel dabei war es insbesondere auch außerhalb der Detektormitte möglichst gleichbleibende Genauigkeiten für die Spurrichtungsrekonstruktion zu erhalten. Wie in Abbildung 6.7 zu sehen ist, nimmt die Genauigkeit der Spurrichtungsrekonstruktion für Ereignisse außerhalb der Mitte des Detektors bei Training lediglich in der Detektormitte (blaue Linie) mit zunehmender Entfernung ab. Wenn man zusätzlich auch entlang der z-Achse trainiert (orange Linie), lässt sich dieser Effekt abschwächen. Erhöht man zusätzlich die Anzahl der Datenpunkte je Ereignis auf 1024 (grüne Linie), lässt sich eine weitere Verbesserung erreichen. Fast konstante Genauigkeiten lassen sich erreichen, wenn zusätzlich der Vertex in den Ursprung des Koordinatensystems verschoben wird und die Datenpunkte mit den gleichen Zeiten auf die gleichen Abstände zum Vertex projiziert werden (rote Linie). Zudem werden alle Signale aus den Daten entfernt, die eine größere Zeit als eine bestimmte Zeitschwelle haben. Als Zeitschwelle wurde 1,25 Nanosekunden gewählt, um einen maximalen Anteil des Čerenkovlichts zu erreichen (siehe auch Abbildung 5.1).

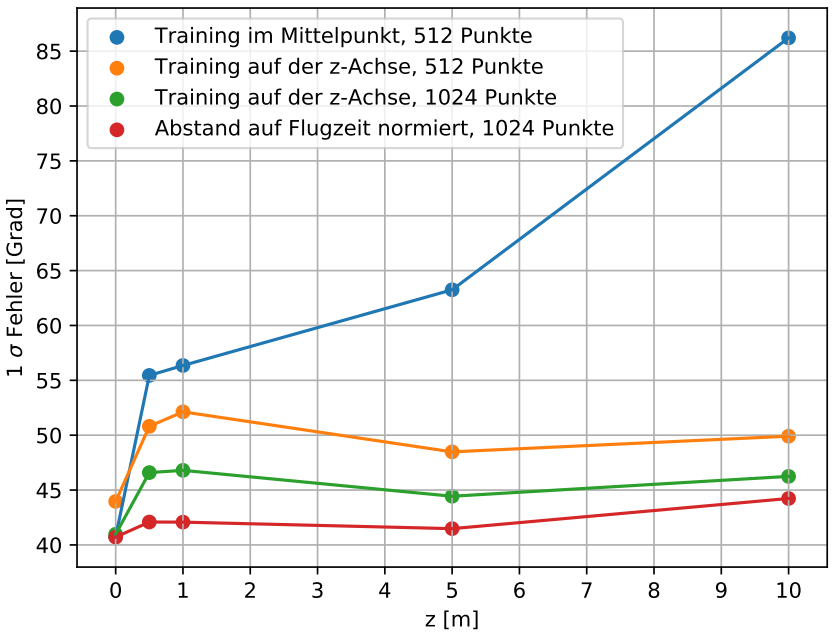

Abbildung 6.7.: Die vertexabhängige Genauigkeit der Spurrichtungsrekonstruktion lässt sich für die verschiedenen Optimierungsschritte der Datenmodifikation mit Hilfe weniger Maßnahme verbessern.

Abschließend wurde der Vertex in den Ursprung des Koordinatensystems gelegt und die Positionen der PMTs auf Einheitskugeln um den Vertex mit der Zeit als Abstand umgerechnet. Wie in Abbildung 6.8 zu sehen ist, führt die Ausweitung des Trainings auf Ereignisse außerhalb der Detektormitte zu einer Verschlechterung der Genauigkeit um etwa 2° auf den Testdaten. Dies lässt sich durch die Erhöhung der Anzahl der Punkte je Ereignis von 512 auf 1024 kompensieren. Die Normierung auf die Flugzeiten wirkt sich nicht auf die energieabhängigen Genauigkeit aus.

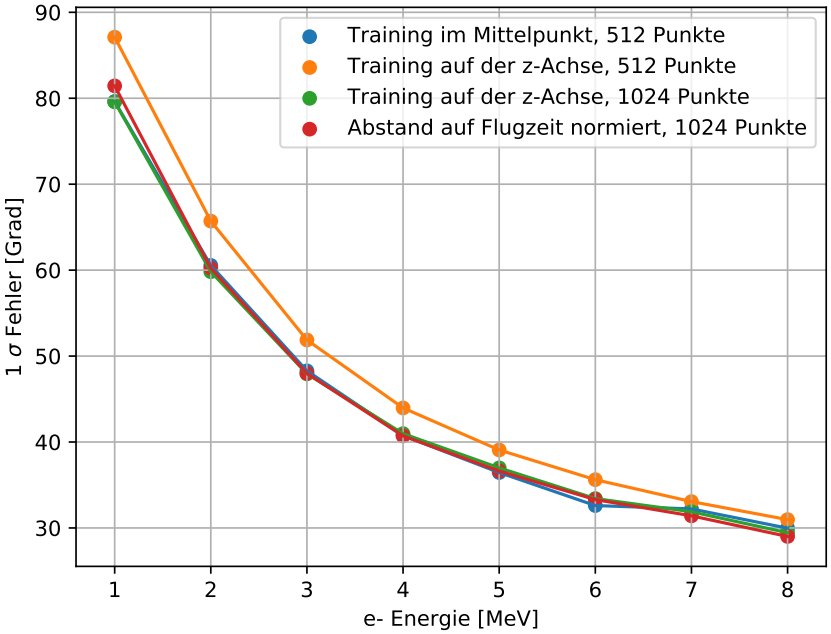

Abbildung 6.8.: Energieabhängie Genauigkeit der Spurrichtungsrekonstruktion für verschiedene Optimierungsschritte

6.6. Untersuchung verschiedener Datensätze

Um realistischere Daten zu erhalten, sollen auch die Auswirkungen verschiedener physikalischer Effekte, die in den Daten zu erwarten sind, untersucht werden.

6.6.1. Berücksichtigung der Flugzeitstreuung

Wie in Abschnitt 2.4.3 beschrieben, haben die PMT eine Flugzeitstreuung mit einer Standardabweichung im Bereich zwischen 1 und 5 Nanosekunden. Um realistische Ergebnisse zu erhalten, muss die Streuung der Flugzeiten (*engl. Transit-Time Spread*) innerhalb der Photomultiplier Tubes auf die Auftreffzeiten angewendet werden.

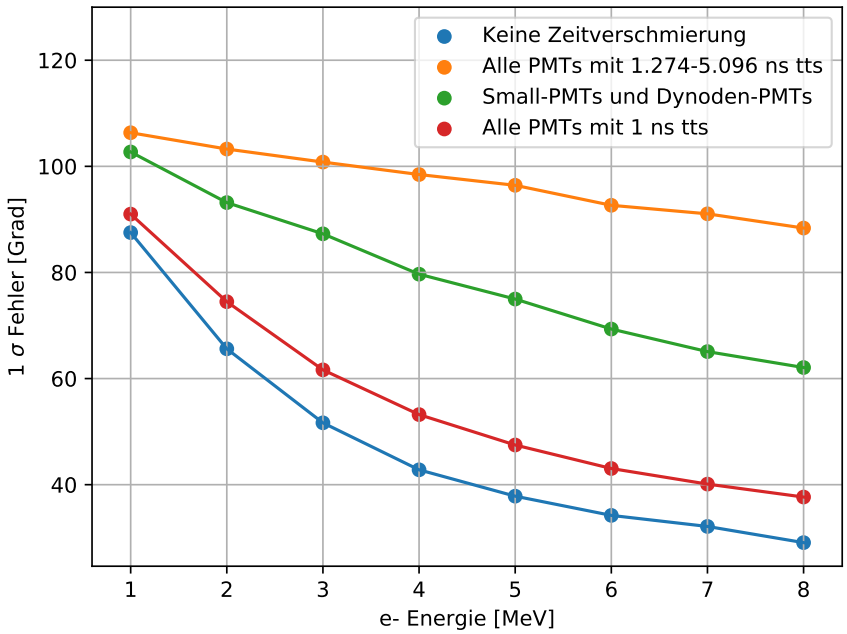

Abbildung 6.9.: Vergleich der Auswirkungen auf die Genauigkeit der Spurrichtungsrekonstruktion für unterschiedliche Annahme für die Flugzeitstreuung

In Abbildung 6.9 sind die Messreihen ohne Flugzeitstreuung und mit Flugzeitstreuung in drei verschiedenen Varianten dargestellt. Dabei ist deutlich zu erkennen, dass bei Nutzung aller PMTs mit den im Abschnitt 2.4.3 genannten Standardabweichungen eine erhebliche Verschlechterung des 1σ-Fehlers auf etwa 98° für eine Teilchenenergie von 4 MeV auftritt. Lässt man die MCP-PMTs mit ihrer sehr hohen Flugzeitstreuung weg, so lässt sich der Effekt deutlich abschwächen. Hierbei wird der 1σ-Fehler für ein 4 MeV Ereignis auf 80° reduziert. Außerdem wurde der fiktive Fall getestet, wenn alle PMTs eine Standardabweichung der Flugzeitstreuung von einer Nanosekunde hätte. Für diesen Fall lässt sich der Fehler noch einmal deutlich reduzieren.

Ein Beispiel für ein modifiziertes Elektron-Ereignis ist in Abbildung 6.10 zu sehen, dabei wurden alle Photonen mit einer Zeit unter 10 Nanosekunden dargestellt. Da auch negative Zeiten auftreten, wurde für die Projektion nur der Betrag der Zeit genutzt werden, um eine Spiegelung im Ursprung zu verhindern. Ein Zentimeter Abstand zum Mittelpunkt entspricht dabei einer Nanosekunde mit einem Offset von einer Nanosekunde. Zusätzlich wurde hier die Streuung der Flugzeiten innerhalb der PMTs zwischen 1,274 Nanosekunden für die Dynoden-PMTs und 5,096 Nanosekunden für die MCP-PMTs berücksichtigt. Mit dem Auge ist keine Häufung früher Auftreffzeiten in Flugrichtung erkennbar.

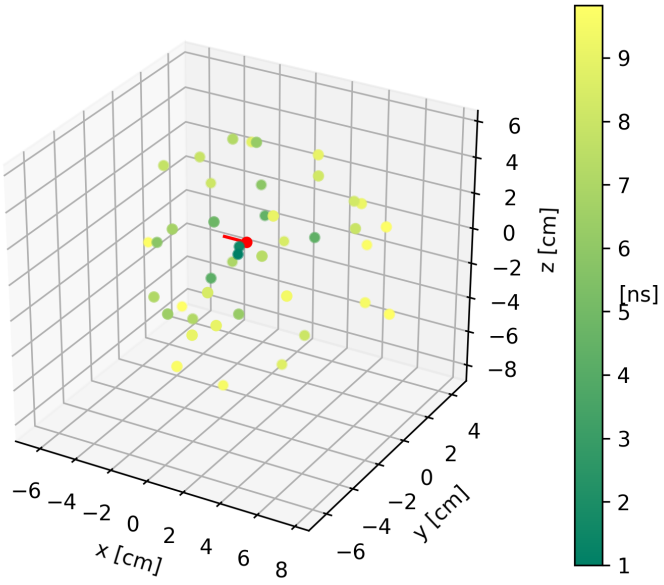

Abbildung 6.10.: Verteilung der Treffer auf den Photomultiplier Tubes für ein Elektron-Ereignis nach Zeitkorrektur. Die Flugzeitstreuung mit einer Standardabweichung zwischen 1,274 und 5,096 Nanosekunden je nach PMT Typ wurde berücksichtigt. Die Farbe gibt die Zeit in Nanosekunden seit Eintreten des Ereignisses an. Der Vertex und die Richtung des Ereignisses sind in rot eingezeichnet.

6.6.2. Berücksichtigung der Unsicherheit bei der Vertexbestimmung

Der Vertex eines Ereignisses lässt sich mithilfe einer Vertexrekonstruktion aus den Daten ermitteln. Dabei ist mit einer Unsicherheit im Bereich von wenigen Zentimetern zu rechnen. Die Unsicherheit ist von der Energie E des Teilchens abhängig und wurde mit

$$\sigma = \frac{10cm}{\sqrt{E}} \tag{6.1}$$

angenommen.

In Abbildung 6.11 ist der Vergleich zwischen einer Messreihe ohne Vertexungenauigkeit und zwei Messreihen mit einer Gaußverschiebung des Vertexes in jede Richtung dargestellt. Dabei ist zu erkennen, dass mit der Standardabweichung aus Formel 6.1 sich die Genauigkeit um etwa 10° für Ereignisse mit einer Energie zwischen 3 und 8 MeV verschlechtert. Bei einer konstanten Standardabweichung von 10 Zentimetern ist eine Verschlechterung von bis zu 20° zu beobachten.

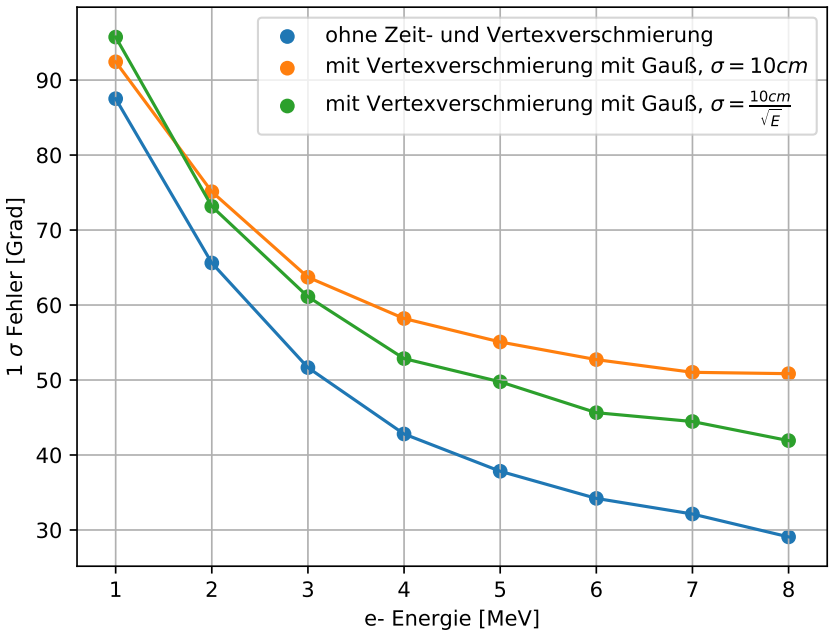

Abbildung 6.11.: Vergleich verschiedener Messreihen mit unterschiedlicher Annahme für Unsicherheit der Vertexposition

6.7. Kombination der Optimierungen

In Abbildung 6.12 ist der 1σ-Fehler für die Abweichung zwischen Monte-Carlo-Wahrheit und der Spurrichtungsrekonstruktion für einen abschließenden Trainings- und Evaluationsdurchlauf sowohl für den idealisierten Fall ohne Zeit- und Vertexverschmierung, als auch für den realitätsnahen Fall mit einer Vertexverschmierung mit $\sigma = \frac{10cm}{\sqrt{E}}$ und nur mit den Dynoden-PMTs mit Flugzeitstreuung dargestellt. Zum einen wurden die Optimierung des neuronalen Netzes und der Datenmodifikation zusammengeführt, zum anderen wurde nochmal eine kleinschrittigere Nachschlagetabelle für die mittleren Flugzeiten verwendet und die Positionen der PMTs angepasst. Als Zeitschwelle wurde 0,75 Nanosekunden gewählt. Ohne Berücksichtigung der Flugzeitstreuung oder Unsicherheit bei der Vertexbestimmung erhält man für ein 4 MeV Ereignis einen 1σ-Fehler von 37,8°. Für den Fall mit Vertexverschmierung und unter ausschließlicher Nutzung der Dynoden-PMTs mit ihrer Flugzeitstreuung von 1,274 Nanosekunden ergibt sich bei einer Energie von 4 MeV einen 1σ-Fehler von 89,4°. Wie in Abbildung 6.13 dargestellt, beträgt ohne Zeit- und Vertexverschmierung bei 4 MeV der 1σ-Fehler für ein Ereignis in 10 Metern Abstand zum Detektormittelpunkt 38,9° und damit nur 1,1° mehr als in der Detektormitte.

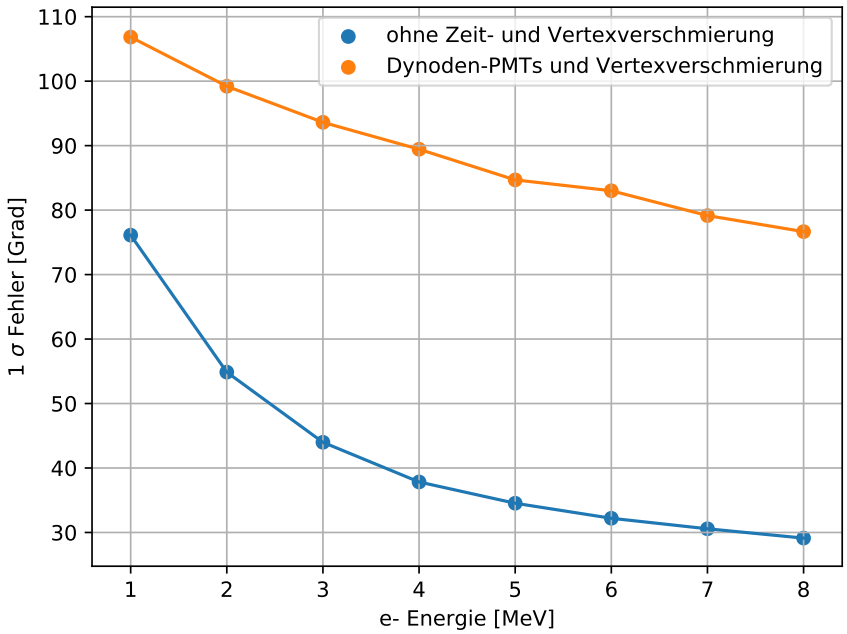

Abbildung 6.12.: Energieabhängie Genauigkeit der Spurrichtungsrekonstruktion für die Kombination der Optimierung für das Netz und die Datenmodifikation

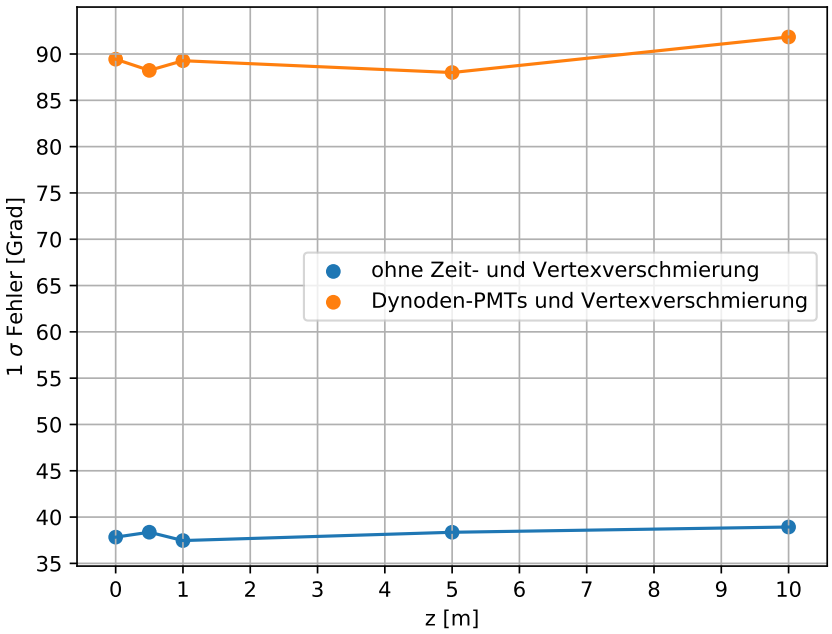

Abbildung 6.13.: Vertexabhängige Genauigkeit der Spurrichtungsrekonstruktion für die Kombination der Optimierung für das Netz und die Datenmodifikation bei einer Teilchenenergie von 4 MeV

7. Zusammenfassung der Ergebnisse und Ausblick

In dieser Arbeit wurde eine Datenaufbereitung und ein neuronales Netz entwickelt, um die Richtung von Elektronen in JUNO zu rekonstruieren. Als Basis für das neuronale Netz wurde ein bestehendes Netz zur Klassifikation von dreidimensionalen Punktwolken genutzt und an die Aufgabenstellung angepasst. Des Weiteren wurde eine Erweiterung der Netzschichten durchgeführt, wodurch für eine Elektronenenergie von 4 MeV eine Verbesserung des Fehlers um etwa 3° erreicht werden kann. Durch die Berücksichtigung der mittleren erwarteten Flugzeiten zwischen Ursprung der Teilchenspur und den Photomultiplier Tubes und die anschließende Normierung des Abstandes der Datenpunkte zum Mittelpunkt anhand der Zeiteinträge lässt sich zudem eine Richtungsrekonstruktion auch außerhalb der Detektormitte erreichen. Dabei konnte die vertexabhängige Zunahme des Fehlers bis auf 1,1° reduziert werden. Die Flugrichtung eines Elektrons aus der Detektormitte mit einer Energie von 4 MeV konnte für 68% der Ereignisse (1σ) mit einer maximalen Abweichung von 37,8° ermittelt werden. Für Elektronen mit 4 MeV in einem Abstand von 10 Metern zur Detektormitte wurde ein 1σ-Fehler von 38,9° berechnet. Unter Berücksichtigung der Unsicherheiten für die Bestimmung des Vertexes und der Auftreffzeiten ergibt sich eine erwartungsgemäße Verschlechterung des 1σ-Fehlers für die Richtungsrekonstruktion. Für eine Standardabweichung von einer Nanosekunde auf den Auftreffzeiten ergibt sich für ein 4 MeV Ereignis ein 1σ-Fehler von 53,2°.

Damit sind die erreichten Genauigkeiten für eine ideale Annahme um etwa 1,1° besser als die bisher erreichten Werte (siehe Abschnitt 4.1).

Um abschließende Aussagen darüber treffen zu können, wie gut die Richtungen der Elektronen und damit auch der Elektronneutrinos im realen JUNO-Detektor bestimmt werden können, müssen die tatsächlichen Flugzeitstreuungen der Photomultiplier Tubes bestimmt werden. In der Simulation wird der zeitliche Übergang vom Flüssigszintillator auf die

Wellenlängenschieber nicht betrachtet, sodass es in der Simulation zu einem sofortigen Beginn der Aussendung der Szintillationsphotonen kommt. Zwar sind für die in JUNO verwendete Mischung die genauen Übergangszeiten noch nicht bekannt, werden aber im Bereich weniger Nanosekunden erwartet. Daher sollte der Zeitunterschied zwischen Szintillationslichts und Čerenkovlicht noch etwas größer ausfallen und eine bessere Separation möglich sein. Außerdem ist der Fehler für die Vertexrekonstruktion idealisiert betrachtet worden, sodass hier eine weitere Untersuchung sinnvoll ist, insbesondere auch im Hinblick auf systematische Fehler.
Zusammenfassend lässt sich aber sagen, dass eine bestmögliche Vertexrekonstruktion und eine möglichst geringe Flugzeitstreuung erreicht werden sollten. Aufgrund ihrer großen Flugzeitstreuung sollte geprüft werden, ob auf die Verwendung der Signale aus den MCP-PMTs verzichtet werden sollte oder ob das Netz darauf trainiert werden kann, Signale aus den verschiedenen PMT-Typen unterschiedlich zu gewichten. Es sollte zudem noch untersucht werden, inwieweit sich eine Ausweitung des Trainings auf weitere Teilchenenergien und eine weitere Verteilung über das komplette Detektorvolumen auf die Genauigkeit der Spurrichtungsrekonstruktion auswirkt. Erste Tests mit einer Variante mit Edge-Features zwischen den Convolution-Schichten auf einem statischen Graphen, der mit den Abständen der Eingabepunkte initialisiert wird, haben gezeigt, dass das Ergebnis noch einmal um bis zu 1,8° verbessert werden kann. Aufgrund der erwarteten Nähe der von Čerenkovphotonen direkt getroffenen PMTs zueinander, scheint dies erfolgversprechend zu sein.

A. Verwendete Datensätze für Grafiken

A.1. Trainingsdurchläufe neuronale Netze

Abbildung 6.1

Trainingsdurchlauf: eval/08_eval_vertexdir

Abbildung 6.3

Dense Layer 1-2: final/14_eval_vertexdir_v5

Dense Layer 1-3: final/13_eval_vertexdir_v6

Dense Layer 0-3: final/18_eval_vertexdir_v14

Abbildung 6.4

Dropout-Rate 0.4: final/6_eval_vertexdir_v14_6

Dropout-Rate 0.5: final/4_eval_vertexdir_v14

Dropout-Rate 0.6: final/5_eval_vertexdir_v14_4

Abbildung 6.5

Max Pooling: final/6_eval_vertexdir_v14_6

Average Pooling: final/8_eval_vertexdir_v15

Abbildung 6.6

Trainingsdurchlauf: eval/22_eval_vertexdir_smeared_v4

Abbildung 6.7 und Abbildung 6.8

Training im Mittelpunkt, 512 Punkte: eval/01_eval_vertexdir
Training auf der z-Achse, 512 Punkte: eval/02_eval_vertexdir
Training auf der z-Achse, 1024 Punkte: eval/07_eval_vertexdir
Abstand auf Flugzeit normiert, 1024 Punkte: eval/08_eval_vertexdir

Abbildung 6.9

Keine Zeitverschmierung: eval/08_eval_vertexdir_512
Alle PMTs mit 1,274-5,096 ns tts: eval/10_eval_vertexdir
Small-PMTs und Dynoden-PMTs: eval/13_eval_vertexdir_mcp
Alle PMTs mit 1 ns tts: eval/14_eval_vertexdir_1ns

Abbildung 6.11

keine Vertexverschmierung: eval/08_eval_vertexdir_512
mit Vertexverschmierung mit Gauß, $\sigma = 10cm$: final/13_eval_vertexdir_v6
mit Vertexverschmierung mit Gauß, $\sigma = \frac{10cm}{\sqrt{E}}$: eval/22_eval_vertexdir_smeared_v4

Abbildung 6.12

Keine Zeit- und Vertexverschmierung: timecutV2/05_eval_vertexdir_v14_4
Dynoden-PMTs und Vertexverschmierung: timecutV2/02_eval_vertexdir_smeared_mcp_-v14_4

Abbildung 6.13

Keine Zeit- und Vertexverschmierung: timecutV2/05_eval_vertexdir_v14_4
Dynoden-PMTs und Vertexverschmierung: timecutV2/02_eval_vertexdir_smeared_mcp_-v14_4

A.2. Simulationsdaten

Abbildung 1.1

TimeCut/user-evt-electron_2_3_MeV_0_0_0_mm_3125_evts_conv.txt Event 5

Abbildung 2.5

Szintillationslicht: user-evt-test_1_4_MeV_0_0_0_mm_400_evts.root
Čerenkovlicht: user-evt-test3_1_4_MeV_0_0_0_mm_400_evts.root

Abbildung 5.1

Szintillationslicht: user-evt-test_1_4_MeV_0_0_0_mm_400_evts.root
Čerenkovlicht: user-evt-test3_1_4_MeV_0_0_0_mm_400_evts.root

Abbildung 6.10

TimeCut/Smeared/user-evt-electron_2_3_MeV_0_0_0_mm_3125_evts_conv.txt
Event 5

Literaturverzeichnis

[1] Abadi, Martín ; Agarwal, Ashish ; Barham, Paul ; Brevdo, Eugene ; Chen, Zhifeng ; Citro, Craig u. a.: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. – URL https://www.tensorflow.org/. – Software verfügbar unter tensorflow.org

[2] Aberle, Christoph ; Elagin, Andrey ; Frisch, Henry J. ; Wetstein, Matthew ; Winslow, Lindley: Measuring directionality in double-beta decay and neutrino interactions with kiloton-scale scintillation detectors. In: *Journal of Instrumentation* 9 (2014), Jun, Nr. 06, S. P06012–P06012. – URL http://dx.doi.org/10.1088/1748-0221/9/06/P06012. – ISSN 1748-0221

[3] Adam, T. ; An, F. ; An, G. ; An, Q. ; Anfimov, N. u. a.: *JUNO Conceptual Design Report*. 2015

[4] Ahmad, Q. R. ; Allen, R. C. ; Andersen, T. C. ; D.Anglin, J. ; Barton, J. C. u. a.: Direct Evidence for Neutrino Flavor Transformation from Neutral-Current Interactions in the Sudbury Neutrino Observatory. In: *Phys. Rev. Lett.* 89 (2002), Jun, S. 011301. – URL https://link.aps.org/doi/10.1103/PhysRevLett.89.011301

[5] Alaeian, Hadiseh: *An Introduction to Cherenkov Radiation*. 2014. – URL http://large.stanford.edu/courses/2014/ph241/alaeian2/. – Zugriffsdatum: 10.12.2019

[6] An, Fengpeng ; An, Guangpeng ; An, Qi ; Antonelli, Vito ; Baussan, Eric u. a.: *Neutrino Physics with JUNO*. J. Phys. G 43 (2016) 030401. 2015

[7] Ashie, Y. ; Hosaka, J. ; Ishihara, K. ; Itow, Y. ; Kameda, J. u. a.: Evidence for an Oscillatory Signature in Atmospheric Neutrino Oscillations. In: *Phys. Rev. Lett.* 93 (2004), Sep, S. 101801. – URL https://link.aps.org/doi/10.1103/PhysRevLett.93.101801

[8] Brun, Rene ; Rademakers, Fons: ROOT - An Object Oriented Data Analysis Framework. In: *AIHENP'96 Workshop, Lausane* Bd. 389, 1996, S. 81–86

[9] Cheng, Yaping ; Ebsen, Lars ; Xu, Yu ; Kun, Zhang: *Investigation of low energy electron track direction reconstruction in LS*. Oktober 2018. – URL `https://indico.cern.ch/event/738555/contributions/3174155/attachments/1736596/2809023/juno_eu_2.pdf`. – Vortrag auf dem JUNO-EU Meeting in Jyvaskyla

[10] Collette, Andrew u. a.: *HDF5 for Python*. – URL `https://www.h5py.org/`

[11] Demtröder, Wolfgang: *Experimentalphysik 4: Kern-, Teilchen- und Astrophysik*. 4. Auflage. Springer Spektrum, Springer-Verlag Berlin Heidelberg, 2014. – ISBN 978-3-642-21475-2

[12] Dertat, Arden: *Applied Deep Learning - Part 4: Convolutional Neural Networks*. 2017. – URL `https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2`. – Zugriffsdatum: 05.12.2019

[13] Deru, Matthieu ; Ndiaye, Alassane: *Deep Learning mit TensorFlow, Keras und TensorFlow.js* -. 1. Aufl. Bonn : Rheinwerk Verlag GmbH, 2019. – ISBN 978-3-836-26509-6

[14] Gao, Feng ; Huang, Guorui ; Heng, Yuekun ; Li, Dong ; Liu, Hulin u. a.: Status of the 20 inch MCP-PMT prototype development for JUNO experiment. In: *Journal of Physics: Conference Series* 888 (2017), sep, S. 012050. – URL `https://doi.org/10.1088/1742-6596/888/1/012050`

[15] Hamamatsu Photonics K.K.: *Photomultiplier tube R12860*. – URL `https://www.hamamatsu.com/eu/en/product/type/R12860/index.html`. – Zugriffsdatum: 24.01.2020

[16] Kiswani, Ahmad: *Convolution and Normalized Cross Correlation on Kepler Architecture*. 2014. – URL `https://sipl.eelabs.technion.ac.il/projects/convolution-and-normalized-cross-correlation-on-kepler-architecture-2/`. – Zugriffsdatum: 05.12.2019

[17] Li, Yu-Feng: *Jiangmen Underground Neutrino Observatory: Status and Prospectives.* 2016. – URL https://arxiv.org/abs/1606.04743

[18] Liao, Dong-Hao ; Liu, Hong-Bang ; Zhou, Yi-Xiong ; Luo, Feng-Jiao ; Wang, Zhi-Min ; Yang, An-Bo ; Xu, Mei-Hang ; Xie, Wan ; Qin, Zhong-Hua: Study of TTS for a 20-inch dynode PMT. In: *Chinese Physics C* 41 (2017), jun, Nr. 7, S. 076001. – URL https://doi.org/10.1088/1674-1137/41/7/076001

[19] Lorenz, Sebastian: *Topological Track Reconstruction in Liquid Scintillator and LENA as a Far-Detector in an LBNO Experiment.* November 2016

[20] Madeja, Michael: *Das kleine Buch vom Gehirn - Reiseführer in ein unbekanntes Land.* Stuttgart : Dt. Taschenbuch-Verlag, 2012. – ISBN 978-3-423-34705-1

[21] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), Dec, Nr. 4, S. 115–133. – URL https://doi.org/10.1007/BF02478259. – ISSN 1522-9602

[22] Nickolls, John ; Buck, Ian ; Garland, Michael ; Skadron, Kevin: Scalable Parallel Programming with CUDA. In: *Queue* 6 (2008), März, Nr. 2, S. 40–53. – URL https://doi.org/10.1145/1365490.1365500. – ISSN 1542-7730

[23] Patterson, Josh ; Gibson, Adam: *Deep Learning - A Practitioner's Approach.* Sebastopol : Ö'Reilly Media, Inc.", 2017. – ISBN 978-1-491-91425-0

[24] Povh, Bogdan ; Rith, Klaus ; Scholz, Christoph ; Zetsche, Frank ; Rodejohann, Werner: *Teilchen und Kerne - Eine Einführung in die physikalischen Konzepte.* 9. Auflage. Springer-Verlag Berlin Heidelberg, 2013. – ISBN 978-3-642-37822-5

[25] Python Software Foundation: *Python Language Reference.* – Verfügbar unter https://www.python.org

[26] Qi, Charles R. ; Su, Hao ; Mo, Kaichun ; Guibas, Leonidas J.: *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.* 2016. – URL https://arxiv.org/abs/1612.00593v2

[27] Qian, Sen: *The R&D and Mass Production of the 20 inch MCP-PMT in China.* Juli 2018. – URL https://indico.cern.ch/event/686555/contributions/2972374/attachments/1682519/2703560/ICHEP2--MCP-PMTs--V3.0.pdf. – Zugriffsdatum:

06.02.2020. – Vortrag auf der International Conference of High Energy Physics in Seoul

[28] Rumelhart, David E. ; Hinton, Geoffrey E. ; Williams, Ronald J.: Learning representations by back-propagating errors. In: *Nature* 323 (1986), oct, Nr. 6088, S. 533–536. – URL `https://doi.org/10.1038/323533a0`

[29] The HDF Group: *The HDF5 Library & File Format*. 2019. – URL `https://www.hdfgroup.org/solutions/hdf5/`. – Zugriffsdatum: 30.01.2020

[30] Wang, Yue ; Sun, Yongbin ; Liu, Ziwei ; Sarma, Sanjay E. ; Bronstein, Michael M. ; Solomon, Justin M.: Dynamic Graph CNN for Learning on Point Clouds. In: *ACM Transactions on Graphics (TOG)* (2019)

[31] Wikipedia: *Photomultiplier — Wikipedia, Die freie Enzyklopädie*. 2018. – URL `https://de.wikipedia.org/w/index.php?title=Photomultiplier&oldid=179186616`. – Zugriffsdatum: 27.01.2020

[32] Wikipedia: *Sekundärelektronenvervielfacher — Wikipedia, Die freie Enzyklopädie*. 2019. – URL `https://de.wikipedia.org/w/index.php?title=Sekund%C3%A4relektronenvervielfacher&oldid=191523991`. – Zugriffsdatum: 27.01.2020

[33] Zuber, Kai: *Neutrino Physics*. 2nd edition. CRC Press, aug 2011. – URL `https://doi.org/10.1201/b11065`

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

bis-MSB	p-bis(o-methylstyryl)-benzene
CNN	Convolutional Neural Network
Conv2D	Convolution in 2 Dimensionen
EdgeConv	Edge Convolution
eV	Elektronenvolt
HDF5	Hierarchical Data Format
JUNO	Jiangmen Underground Neutrino Observatory
keV	Kiloelektronenvolt
LAB	Linear Alkyl Benzene
NNVT	North Night Vision Tech.
MatMul	Matrizenmultiplikation
MaxPool2D	Max-Pooling in 2 Dimensionen
MCP	Micro-Channel-Plate
MeV	Mega-Elektronenvolt
MLP	Multi-Layer-Perzeptron
PMT	Photomultiplier Tube
PPO	2,5-diphenyloxazole
SNO	Sudbury Neutrino Observatory
tts	Transit-Time Spread

Danksagung

Ich möchte mich bei allen bedanken, die an der Entstehung dieser Arbeit mitgewirkt haben. Bei Dr. Björn Wonsak dafür, dass er sich so problemlos dazu bereit erklärt hat, sich gemeinsam mit mir ein Thema für meine Arbeit zu überlegen und ich mich während der ganzen Zeit mit Fragen an ihn wenden konnte. Prof. Dr. Simone Frintrop möchte ich für die Bereitschaft danken, diese Arbeit zu begutachten. A special thanks goes to Dr. Mikko Lauri for the continuous meetings with all the students who are writing a thesis in the CV group.
David Meyhöfer danke ich für eine Unterstützung bei der Einrichtung der JUNO-Simulation und dem Extrahieren der Daten aus den ROOT-Dateien. Malte Stender danke ich für das fleißige Korrekturlesen meiner Arbeit in vielen Zwischenschritten. Ich danke Dr. Henning Rebber dafür, dass er mir immer wieder Fragen zur JUNO-Simulation und und den Lookup-Tables beantwortet hat und mir die Funktionsweise eines PMTs anschaulich erklärt hat. Bei Christian Hellwig, Jan Buerger, Morten Henken und Puja Tehrani möchte ich mich für die angenehme Zusammenarbeit in unserem Büro bedanken. Rosmarie Wirth danke ich für die gemeinsame und kurzweilige Zeit beim Workshop zum Deep Learning in der Astroteilchenphysik in Aachen. Allen Mitglieder der Forschungsgruppe Neutrinophysik möchte ich für die freudige Zeit bei den Meetings, dem gemeinsamen Mittagessen, Grillen, Kuchen essen und Tisch-Kickern danken.
Bei meinen Schwestern Friederike und Sophie sowie meinen Eltern möchte ich mich für die regelmäßigen Gesellschaftsspieleabende und die Unterstützung in der ganzen Zeit bedanken. Ich möchte mich bei Melf Johannsen und Sebastian Döbel ganz herzlich für die Freundschaft seit dem Beginn des Bachelorstudiums bedanken, ohne die so manche Phase des Studiums doch schwer erträglich gewesen wäre. Zudem möchte ich mich bei beiden für das Korrekturlesen der Arbeit vor der Abgabe bedanken. Bei Tim Rolff möchte ich mich für die Tipps, die er mir immer wieder zu meiner Arbeit gegeben hat, bedanken. Bei allen Besuchern der CiS-Stammtische, sowie bei der OE-AG der Informatik möchte ich mich für tolle 6,5 Jahre bedanken, die mein Studium auch außerhalb der Vorlesungen bereichert haben.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, 20.02.2020	
Ort, Datum	Unterschrift

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik und online auf der Website der Arbeitsgruppe Neutrinophysik zu.

Hamburg, 20.02.2020	
Ort, Datum	Unterschrift