Neuronale Netzwerke zur Signal/Untergrund Klassifikation für BabyIAXO

Dongjin Suh

Bachelorarbeit in Physik
angefertigt im Physikalischen Institut

vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität
Bonn

März 2022

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate kenntlich gemacht habe.

Bonn, ...
Datum Unterschrift

1. Gutachter: Prof. Dr. Klaus Desch
2. Gutachter: Dr. Jochen Kaminski

Inhaltsverzeichnis

1 Einleitung

Ein wichtiges Ziel der modernen Physik ist eine einheitliche Theorie zur Beschreibung aller Phänomene in der Welt zu finden. Nach aktuellen Erkenntnissen beruht sich die heutige Physik auf der einen Seite auf die allgemeine Relativitätstheorie von Albert Einstein. Allerdings schränkt sich die Theorie nur auf die Gravitationskraft ein, welche eine der vier fundamentalen Wechselwirkungen sind. Für die anderen drei fundamentalen Kräfte; die starke Kraft, die schwache Kraft und die elektromagnetische Kraft gibt es das sogenannte Standardmodell der Teilchenphysik, welche alle bekannten Elementarteilchen und die drei wichtigen Wechselwirkungen zwischen ihnen beschreibt. Das Standardmodell basiert auf Symmetrien, sog. lokalen Eichsymmetrien, die die Flexibilität der Natur gut beschreiben. Das Standardmodell besagt, dass die CPT-Symmetrie unter allen Umständen invariant ist. Daraus folgt bei einer PT-Verletzung zu einer CP-Verletzung. Für die schwache Wechselwirkung konnte erwartungsgemäß eine CP-Verletzung nachgewiesen werden. Mathematisch ist die starke Wechselwirkung stark mit der schwachen verwandt. Daher erwartet man mathematisch durch die gleichen Formalismen auch hier eine CP-Verletzung, die man aber bislang nicht messen konnte. Das Problem, dass zwar theoretische Überlegungen eine CP-Verletzung forderten aber diese nicht beobachtet wurde, führte zu der Idee, ein neues Teilchen einzuführen. Das hypothetische Teilchen, dass eine Lösung dieses Problem sein kann, um die unbeobachtete starke CP-Verletzung zu erklären, wird als Axion bezeichnet.
Nun kommt man zu dem CAST-Experiment, welches mittlerweile durch das IAXO-Experiment abgelöst wurde, somit auch zum IAXO-Experiment, dessen Ziel auf die Detektion eines möglichen Axion-Teilchens gelegt wurde. Allerdings wird erwartet, dass die Wahrscheinlichkeit zur Detektion eines Axion-Teilchens sehr gering ist. Zudem existieren Untergrundstrahlungen wie kosmische Strahlung und radioaktive Untergrund des Detektormaterials, die unvermeidbar mit am Detektor aufgenommen werden. Die Tatsache der selten erwarteten Vorkommnissen von Axionphotonen und dagegen die hohen Untergrundrate ist nun die Motivation dieser Arbeit. Es soll eine effiziente Methode entwickelt werden, um eine gute Trennung zwischen möglichen photonartigen Signalen eines Axions und untergrundartigen Ereignissen zu erreichen.
Innerhalb der Arbeit werden die neuronale Netzwerke, die potentiell sehr effektive Methode sein können, betrachtet und untersucht. Es werden zwei verschiedene Arten der neuronalen Netzwerke in Betracht gezogen. Es sollen jeweils das ***M**ulti**l**ayer **P**erceptron* (MLP) und ***C**onvolutional **N**eural **N**etwork* aufgebaut und durch Training der Netzwerke optimiert werden. Anschließend kommt die Validierung der optimierten MLP und und CNN, um zu überprüfen, ob die Netzwerke nachvollziehbar verhalten und keine unerwartete Aktivitäten oder verfälschte Ergebnisse zeigen. Zudem wird die Signaleffizienz der beiden Netzwerken bestimmt und zum Schluss sollen die Effizienzen der Netzwerke mit der Signaleffizienz der üblich verwendeten Likelihood-Methode und auch der Unterschied zwischen MLP und CNN verglichen und diskutiert werden.

2 Axionen

Axion bezeichnet ein hypotetisches, noch nicht nachgewiesenes Elementarteilchen, welches eingeführt wurde, um das starke CP-Problem zu lösen. Axionen besitzen kaum Masse und wechselwirken kaum mit sichtbarer Materie, weshalb sie auch oft als ein möglicher Kandidat für die dunkle Materie angesehen werden.
In der Teilchenphysik steht CP für Charge (Ladung) und Parity (Parität) oder Charge-conjugation Parity symmetry: die Kombination aus Ladungskonjugationssymmetrie (C) und Paritätssymmetrie (P). Nach der aktuellen mathematischen Formulierung der Quantenchromodynamik (QCD) könnte es bei starken Wechselwirkungen zu einer Verletzung der CP-Symmetrie kommen.
Die CP-Symmetrie besagt, dass die Physik unverändert bleiben sollte, wenn Teilchen mit ihren Antiteilchen vertauscht wurden und dann auch linkshändige und rechtshändige Teilchen vertauscht wurden. Dies entspricht dem Durchführen einer Ladungskonjugationstransformation und dann einer Paritätstransformation. Dazu lässt sich der folgende Term aus der Lagrange-Dichte der Quantenchromodynamik (QCD) schreiben, welcher die CP-Symmetrie der starken Wechselwrikung verletzt:

$$\mathcal{L} = \Theta \frac{\alpha_s}{4\pi} G_a^{\mu\nu} \tilde{G}_{a\mu\nu} \tag{1}$$

wobei $G_a^{\mu\nu}$ und $\tilde{G}_{a\mu\nu}$ Feldstärketensor des Gluon-Feldes und α_s die Kopplungskonstante der starken Wechselwirkung sind. Die Größe von Θ beschreibt das Ausmaß der CP-Verletzung.
Eine CP-Verletzung bei der schwachen Wechselwirkung ist bereits bestätigt. Jedoch wurde nie eine Verletzung der CP-Symmetrie in irgendeinem Experiment beobachtet, das nur die starke Wechselwirkung beinhaltete. Eine starke CP-Verletzung würde zu einem magnetischen Moment des Neutrons führen. Da die konstituierenden Quarks des Neutrons geladen sind und u- und d-Quarks unterschiedliche Ladungen besitzen, erwartet man naiverweise, dass sich die Ladungen nicht gegenseitig perfekt aufheben. Dies würde zu einem von Null verschiedenes elektrisches Dipolmoment führen, obwohl das Neutron elektrisch neutral ist. Sofern aber das Neutron ein elektrisches Dipolmoment hat, ist die Zeit- (T) und Paritätstransformation (P) nicht mehr symmetrisch zum Ausgangszustand. Das würde wiederum bedeuten, anhand des gemessenen Dipolmomentes eines Neutrons lässt sich die starke CP-Verletzung überprüfen. Bisher konnte das Dipolmoment nicht nachgewiesen werden. Die obere Grenze liegt derzeit bei $d_n < 1,3 \cdot 10^{-26}$ e·cm.[1]

Folglich legt dieses Ergebnis eine obere Schranke für Θ fest:

$$\Theta \leq 10^{-10} \tag{2}$$

Ein solch kleiner Wert für Θ verlangt eine unnatürliche Feinabstimmung (fine tuning). Um dieses Problem zu lösen, führten Roberto Peccei und Helen Quinn das Peccei-Quinn-Mechanimus[2] ein. Sie schlagen eine neue Symmetrie in Form einer U(1)-Phasentransformation vor, welche spontan unterhalb einer bestimmten Energieskala f_a gebrochen wird und deren Brechung den Wert Null für Θ generiert. Bei diesem Bruch der Symmetrie wird ein sogenanntes Pseudo-Goldstone Boson hervorgerufen, welches den Name Axion erhalten hat.

Axionen können über die Gluonen an zwei Photonen koppeln, dieser Prozess wird als der inverse Primakoff Effekt[3] bezeichnet. Beim inversen Primakoff-Effekt wechselwirkt ein Axion mit einem virtuellen Photon eines elektrischen oder magnetischen Feldes. Für die solaren Axionen, die im Zentrum der Sonne entstehen können,

gilt eine ähnliche Verteilung wie die Photonen im Kern der Sonne, mit den sie gekoppelt werden. Es entstehen dann aufgrund der kinetischen Energie ein Photon im weichen Röntgenbereich. Das erwartete Spektrum der solaren Axionen ist in der Abb. 1 zu sehen.

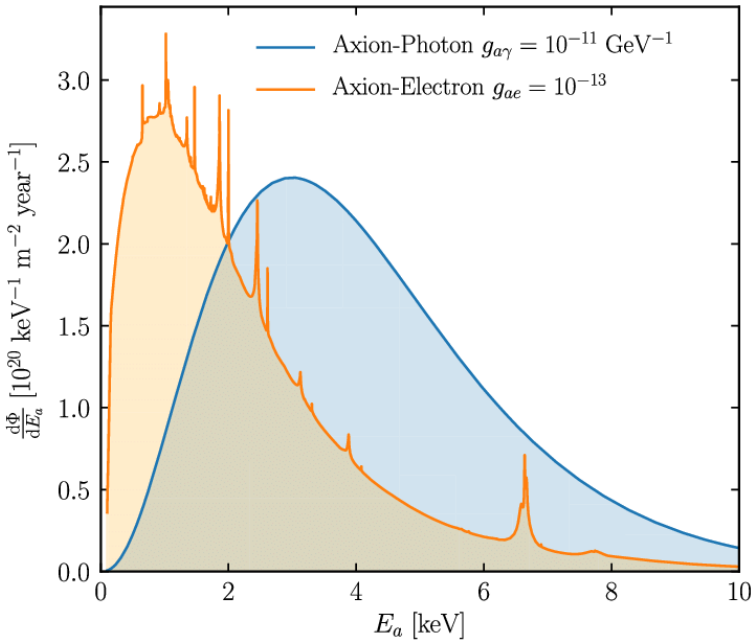


Abbildung 1: Der auf der Erde erwartete solare Axionenfluss und seine Komponenten aufgrund der Axion-Elektronen-Kopplung (Bremsstrahlung, Compton und Axiorekombination) in Orange und der Axionphotonen-Kopplung (Primakoff) in Blau.[4]

3 The CERN Axion Solar Telescope - CAST

The CERN Axion Solar Telesccope (CAST) ist ein Experiment, um nach hypothetischen Teilchen, dem sogenannten Axion, zu suchen. Um ein mögliches Axion zu detektieren, muss zunächst eine Axionquelle geben. Da heiße und thermische stellare Plasmen sehr effiziente Axionquellen sein können[5], stellt die Sonne eine mögliche Axionquelle dar. Sie könnten im Inneren der Sonne produziert werden, wenn durch Röntgenstrahlen Elektronen und Protonen in Gegenwart starker elektromagnetischer Felder abgestreut werden (Primakoff Effect)[6].

Dafür wird das sogenannte Axion-Helioskop-Prinzip verwendet. Wird auf der Erde ein transversales Magnetfeld auf die Sonne ausgerichtet, dann können darin theoretisch von der Sonne emittierte solare Axione in reelle Photonen umgewandelt werden. Analog zur Produktion der Axione im solaren Plasma spielt hier der Primakoff-Effekt die entscheidende Rolle. Passiert ein Axion ein transversales Magnetfeld, so kann es durch den zeitlich invertierten Primakoff-Effekt in ein reelles Photon umgewandelt werden. Da die Ausbreitungsrichtung des Axions und die Magnetfeldrichtung in einem Winkel von 90° zueinander stehen müssen, muss das Magnetfeld der Sonne nachgeführt werden. Dadurch wird eine möglichst lange Beobachtungszeit ermöglicht. Die Umwandlung eines Axions in ein reelles Photon erfolgt unter Erhaltung des Impulses und der Energie des Axions. Die Energieverteilung der Photonen, die das Magnetfeld verlassen, entspricht also der Energieverteilung der ursprünglichen solaren Axione. Diese können mit geeigneten Detektionssystemen für Röntgenstrahlung am Ende des Magnetfeldes nachgewiesen werden. Also können in einem transversalen Magnetfeld die Sonnenaxionen oder andere axionähnliche Teilchen (ALPS) kohärent zurück in Röntgenphotonen im keV-Bereich umgewandelt werden.[7] Das Experiment besteht somit darin, die solaren Axionen aus der Sonne zu detektieren. Das Experiment ist aufgebaut mit einem 9,26 m langen Prototype des Dipolmagneten des Large Hadron Colliders, der ein zur Ausbreitungsrichtung der solaren Axione transversales und homogenes Magnetfeld von maximal 9,5 T erzeugt. Dieser Magnet ist auf einer beweglichen Plattform montiert, die eine Bewegung von ±8° vertikal und ±40° horizontal erlaubt und an beiden Enden sowohl mit gasgefüllten als auch mit Festkörper-Röntgendetektoren mit niedrigem Hintergrund gekoppelt, sodass er die Sonne fast 1,5 Stunden lang sowohl bei Sonnenaufgang als auch bei Sonnenuntergang beobachten kann. [7] Den Rest des Tages werden Hintergrunddaten aufgezeichnet.

Da Axionen eine Masse im < 1 eV Bereich haben, ist deren Energie äquivalent zur Energie der Photonen aus denen sie erzeugt werden (jedenfalls im Falle von solaren Axionen). Die Photonen im Kern der Sonne folgen eine Schwarzkörper Strahlung, weshalb die Axionen, die dort generiert werden eine ähnliche Verteilung aufweisen. In Hinblick auf die 15 Mio. K Temperatur im Kern der Sonne ergibt das also eine Energie der erwarteten Axionen von ungefähr 3-4 keV (Siehe Abb. 1). Die so erzeugten Axionen können dann in dem Magneten durch Interaktion mit den virtuellen Photonen des Magnetfeldes zu Photonen zurück konvertiert werden. Diese tragen dann erneut die Energie dieser 3-4 keV. So erwartet man die solaren Axionen im weichen Röntgeenbereich, weshalb an beiden Enden des Magneten hochempfindlichen Detektoren angebracht sind, die im Energiebereich der Röntgenstrahlung (0,5 keV bis 20 keV) sensitiv sind.

3.1 Detektoren

Drei verschiedene Arten von Detektoren wurden entwickelt, um die Röntgenstrahlen zu detektieren, die durch die Umwandlung der Axionen im Inneren des Magneten entstehen: eine Zeitprojektionskammer (TPC), ein

CCD und ein MikroMegas-Detektor.

Es wurde festgestellt, dass die TPC für Röntgenstrahlen von Sonnenuntergangs-Axionen empfindlich ist. Auf der anderen Seite des Magneten, den Axionen des Sonnenaufgangs zugewandt, arbeiten gleichzeitig ein CCD-Detektor und ein Micromegas-Detektor. Der CCD-Detektor arbeitet in Verbindung mit einem Spiegelsystem, um die aus den Magnetbohrungen kommende Röntgenstrahlung zu fokussieren. Ein Micromegas-Pixeldetektor mit höherer Unterdrückungsfähigkeit soll das CCD ersetzen. Das Spiegelteleskop fokussiert die von der Magnetbohrung induzierten Röntgenstrahlen auf einen Submillimeterpunkt. Dies ermöglicht die Verwendung eines sehr kleinen Detektors und die Verbesserung des erwarteten Signal-Hintergrund-Verhältnisses um etwa zwei Größenordnungen. [8]

MicroMegas Detektor: Collar und Giomataris[9] schlug zuerst die Vorteile der Verwendung des MicroMegas für Messungen mit niedriger Schwelle und niedrigem Hintergrund vor, wie sie für das CAST-Experiment erforderlich sind. Es handelt sich hierbei um Gasdetektoren, die zum effizienten Nachweis von Photonen mit einer Energie zwischen 1 keV und 10 keV optimiert sind. Die wesentlichen Vorteile dieser Detektoren sind deren niedriger Hintergrund, ihre sehr gute Ortsauflösung, eine hohe Nachweiswahrscheinlichkeit für Röntgenphotonen und die niedrigen Herstellungskosten. Technologisch ist das Micromegas-Konzept eine Weiterentwicklung des Vieldrahtproportionalzählers, wobei das Drahtgitter des Vieldrahtproportionalzählers durch eine mikrostrukturierte Kupferfolie mit einem Lochdurchmesser von ungefähr 25 µm ersetzt wurde. Bei der Herstellung der Detektoren wurde speziell darauf geachtet, dass nur Materialien mit intrinsisch geringer natürlicher Radioaktivität verwendet wurden. [10]

GridPix Detektor: Aufgrund von sehr geringen Konversionswahrscheinlichkeit zu Röntgenphotonen, die wichtigsten Anforderungen an den Detektor sind sowohl eine hohe Detektionseffizienz als auch eine gute Hintergrundunterdrückung. Aus diesem Grund kommt es zur Anwendung eines GridPix basierten Detektors. Das GridPix Detektor verwendet ein Timepix ASIC[11] als Ausleser und zur Gasverstärkung wird Integrated Grid (InGrid)[12, 13] eingesetzt. Das InGrid wird direkt auf die Oberseite des Chips durch photolithografische Nachbearbeitungstechniken produziert, die kleine Größen und präzise Ausrichtung zulassen. Auf diese Weise wird jedes Gitterloch des Netzes direkt über den Pixel platziert. Der gesamte Aufbau aus Pixelchip und InGrid wird dann GridPix genannt.

3.2 BabyIAXO

Das CAST-Experiment soll nun langfristig vom Nachfolgerexperiment IAXO (International Axion Observatory) abgelöst werden. IAXO ist ein Axion-Helioskop der neuen Generation, dessen Hauptziel es ist, Axionen (oder andere ähnliche Partikel) zu erkennen, die möglicherweise in großen Mengen vom Sonnenkern emittiert werden. Im großen und ganzen ist das IAXO ein Nachfolger des CAST und somit ist der allgemeine experimentelle Aufbau und Prinzip nicht viel anders von CAST.
IAXO wird einen riesigen 20 m langen toroidalen supraleitenden Magneten mit acht Spulen und acht Bohrungen mit 60 cm Durchmesser zwischen den Spulen verwenden. Dieser Magnet wird ähnlich wie bei einem herkömmlichen Teleskop auf einer beweglichen Struktur platziert, um den Magneten auf die Sonne auszurichten. Am Ende der Magnetbohrungen wird eine speziell gebaute Röntgenoptik die vermeintlichen axioninduzierten Photonen in kleinen Bereichen ($0.2\,\mathrm{cm}^2$) mit einer Brennweite von etwa 5 Metern fokussieren. Jeder

der Brennpunkte wird von verschiedenen Detektortypen mit extrem niedrigem Hintergrund abgebildet. Dadurch wird erwartet, dass man mit IAXO eine verbesserte Sensitivität um Faktor 10000 bis 20000 gegenüber CAST erreicht.[14]
Das BabyIAXO ist ein experimentelles Zwischenstadium vor IAXO. Mit BabyIAXO soll eine Leistungsüberprüfung für IAXO und gleichzeitig auch signifikante wissenschaftliche Ergebnisse erreicht werden. BabyIAXO ist als kleines IAXO konzipiert: Es wird aus zwei 10 m langen Magnetspulen bestehen, die zwei Bohrungen mit je 70 cm Durchmesser bieten. Die verwendeten Röntgenoptik und Detektoren werden ähnliche wie für IAXO erwartete Dimensionen haben.

3.3 Daten aus einem GridPix artigen Detektor

Für die Detektion werden gasgefüllte Detektoren unter anderem für weiche Röntgenstrahlung im Bereich 0 - 10 keV. Dabei funktioniert es so, dass in einer gasgefüllten Kammer, an die ein elektrisches Feld angelegt ist, geladene Teilchen durch Ionisation primäre Elektronen erzeugt werden können. Diese primären Elektronen können dann durch das elektrische Feld in Richtung Anode driften, wo sich die Auslese befindet. Die verwendeten Detektoren haben unterhalb der Anode zur Auslese einen sogenannten GridPix. Dies ist eine Kombination eines Timepix (ein 256x256 Pixel ASIC) mit einer Gasverstärkungszone. Zusammen erlaubt uns dies jedes primäre Elektron einzeln aufzuzeichnen. Während sie das elektrische Feld durchqueren, kommt es durch Interaktion mit dem Gas zu einer Diffusion, sodass die primären Elektronen von ihrem Ursprungsort statistisch fluktuiert verteilt werden. Nun können Teilchen wie Myonen, die in der oberen Atmosphäre erzeugt werden, den Detektor nahezu ungehindert durchqueren und auf ihrer gesamten Weg durch die Gaskammer ionisieren. Auf der anderen Seite haben die Photonen ein einzelnes Photoelektron als Ursprung, welches dann nur in direkter Nähe eine zur Energie propotionale Anzahl weiterer Elektronen-Ion Paare erzeugt. Das bedeutet, dass die Myonen einen länglich verstreuten Spur hinterlässt, während der Ursprung eines Photons eher punktförmig aussieht. Anhand dieser Variation haben verschiedene Arten von detektierter Strahlung nun verschiedene geometrische Eigenschaften und genau diese prinzipielle, geometrische Unterscheidung zwischen verschiedenen Teilchen ermöglicht einem eine Möglichkeit zur Trennung zwischen Signal und Untergrund.

4 Artificial Neural Networks

Künstliche neuronale Netzwerke (in Engl. Artificial Neural Network - ANN) sind eine Modellbildung von Netzen aus künstlichen Neuronen, angelehnt an einem biologischen Vorbild. Allerdings geht es hier mehr um eine Abstraktion von Informationsverarbeitung, weniger um das Nachbilden biologischer neuronaler Netze. Also sind ANNs Computersysteme, die von den biologischen neuronalen Netzen inspiriert sind.

Ein ANN basiert auf eine Ansammlung von Knotenpunkte, die miteinander verbunden sind. Diese Knotenpunkte werden als künstliche Neuronen bezeichnet, welche die Neuronen in einem biologischen Gehirn darstellen. Jedes Neuron ist verbunden mit anderen Neuronen und jede Verbindung besitzt ein gewisses Gewicht, die die Stärke des Einflusses eines Neurons auf einen anderen bestimmt. Alle Neuronen bekommen eine Eingabe und produziert anhand der Eingabe Werte eine Ausgabe, welche als neue Eingabe an anderen Neuronen weitergegeben werden können. Die Neuronen empfangen eine oder mehrere Eigaben und alle Eingaben werden aufsummiert, wobei die Eingaben in einem Neuron in der Regel individuell gewichtet ist. Die Summe aller unterschiedlich gewichteten Eingaben wird dann durch eine Aktivierungsfunktion geleitet, welche üblicherweise eine nicht-lineare Funktion ist. Die Rückgabe der Funktion kann dann als Ausgabe eines Neurons entweder an nächsten Neuronen weitergegeben werden oder als endgültige Ausgabe interpretiert werden, falls das Neuron in der sogenannten Ausgabe-Ebene befindet.

Die Neuronen in einem Netzwerk sind typischerweise in mehreren Schichten organisiert, insbesondere beim *Deep Learning*. Neuronen einer Schicht verbinden sich nur mit Neuronen der unmittelbar vorhergehenden und unmittelbar folgenden Schichten. Die Schicht bzw. Ebene, die externe Daten empfängt, ist die Eingabeschicht. Die Schicht, die das endgültige Ergebnis erzeugt, ist die Ausgabeschicht. Dazwischen befinden sich null oder mehr verborgene Schichten. Sichtbare und unsichtbare Netzwerke werden ebenfalls verwendet. Zwischen zwei Schichten sind mehrere Verbindungsmuster möglich. Sie können „vollständig verbunden“ sein, wobei sich jedes Neuron in einer Schicht mit jedem Neuron in der nächsten Schicht verbindet. Sie können sich bündeln, wobei sich eine Gruppe von Neuronen in einer Schicht mit einem einzelnen Neuron in der nächsten Schicht verbindet, wodurch die Anzahl der Neuronen in dieser Schicht reduziert wird. Neuronen mit nur solchen Verbindungen bilden einen gerichteten azyklischen Graphen und werden als *feedforward network*[15] bezeichnet. Alternativ werden Netzwerke, die Verbindungen zwischen Neuronen in derselben oder vorherigen Schichten ermöglichen, als *recurrent network*[16] bezeichnet.

4.1 Aktivierungsfunktion

Eine Aktiverungsfunktion bestimmt den Ausgabewert eines Neurons aus dem gegebenen Eingabewert. Wenn eine Aktivierungsfunktion fest definiert ist, gilt die Funktion für den Verarbeitungsprozess aller Neuronen in einem Netzwerk.

Die verschiedenen Aktivierungsfunktionen, die in folgenden eingeführt werden sollen, sind zunächst in der Abb. 2 zusammen graphisch dargestellt:

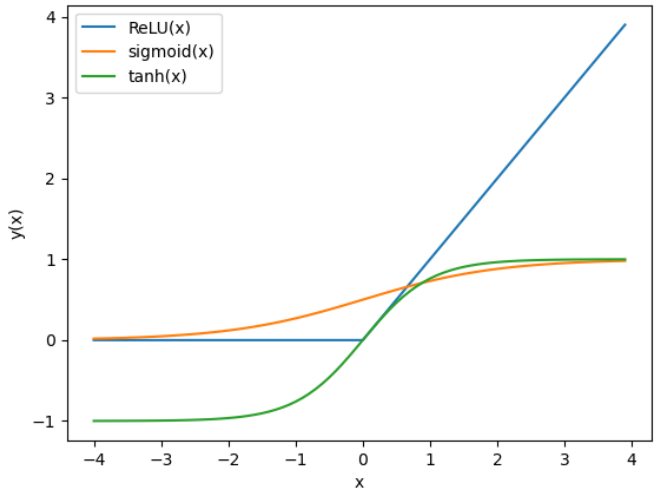


Abbildung 2: Graphische Darstellung der Aktivierungsfunktionen ReLU, Sigmoid, und Tangens hyperbolicus

- Sigmoid Funktion:
  Die Sigmoid-Funktion ist eine besondere Form der logistischen Funktion und ist wie folgt definiert:

$$\text{sig}(x) = \frac{1}{1 + \exp(-x)} \tag{3}$$

  Die Sigmoid-Funktion besitzt eine S-förmige Kurve und verwandelt Zahlen aus reellen Bereich in einem Wert in dem Intervall zwischen 0 und 1. Auch wenn also die Eingabe in die Funktion entweder eine sehr große negative Zahl oder eine sehr große positive Zahl ist, liegt die Ausgabe immer zwischen 0 und 1. Aus diesem Grund wird die Sigmoid-Funktion auch als *squashing function* bezeichnet. Die Sigmoid-Funktion kann als Aktivierungsfunktion in neuronalen Netzen verwendet werden. Wenn die Aktivierungsfunktion für ein Neuron eine Sigmoid-Funktion ist, ist dies eine Garantie dafür, dass die Ausgabe dieser Einheit immer zwischen 0 und 1 liegt. Da die Sigmoid-Funktion eine nicht-lineare Funktion ist, wäre die Ausgabe dieser Einheit eine nicht-lineare Funktion der gewichteten Summe der Eingänge. Ein solches Neuron, das eine Sigmoid-Funktion als Aktivierungsfunktion verwendet, wird als Sigmoideinheit bezeichnet.

- ReLU Funktion:
  ReLU steht für **Re**ctified **L**inear Activation **U**nit und ist eine Aktivierungsfunktion eines künstlichen Neurons. ReLU ist nichts anderes als eine sehr einfache nicht lineare Transformationsfunktion:

$$\text{ReLU}(x) = \max(x, 0) \tag{4}$$

  Bei einem gegebenen Element x ist die Funktion als das Maximum dieses Elements und 0 definiert. Also behält die ReLU-Funktion nur positive Elemente bei und verwirft alle negativen Elemente, indem sie die entsprechenden Aktivierungen auf 0 setzt. Sie ist seit Jahren in der Regel die übliche Wahl für eine Aktivierungsfunktion, sowohl aufgrund der einfachen Implementierung als auch der guten Leistung bei einer Vielzahl von Vorhersageaufgaben und dafür erstaunlichen Güte.[17]

- Tangens hyperbolicus (Tanh) Funktion:
  Ähnlich wie bei Sigmoid Funktion wandelt die Tanh Funktion ihre Eingaben zu Elementen innerhalb

eines Intervalls zwischen -1 und 1 und nimmt als Wert alle reellen Zahlen an. Die Funktion ist folgendermaßen definiert:

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \tag{5}$$

4.2 Backpropagation

Dieser Abschnitt ist angelehnt an dem Buch „Theorie der Neuronalen Netze“[18] von Raúl Rojas.

Das *Backpropagation* ist nichts anderes als das Variieren der Gewichte entlang des größten Gradienten. Dazu berechnet *Backpropagation* den Gradienten im Gewichtsraum eines *Feedforward*-Neuronalnetzwerks in Bezug auf eine Verlustfunktion. Zunächst wird die Ausgabe eines Neurons dargestellt. Für die Ausgabe o_j eines Neurons j und die über alle Eingabewerten mit unterschiedlichen Gewichtungen summierte Eingabe y_j in das Neuron j gilt:

$$o_j = \varphi(y_j) \tag{6}$$

$$y_j = \sum_{i=1}^{n} x_i w_{ij} \tag{7}$$

dabei ist φ die Aktivierungsfunktion, n ist die Anzahl der Eingaben, x_i entspricht dem Eingabewert i und w_{ij} die Gewichtung zwischen Eingabewert i und Neuron j.
Dann wird die Verlustfunktion L (loss function) definiert, welche die Abweichung zwischen dem erwarteten und dem errechneten Wert wiederspiegelt:

$$L = \frac{1}{2} \sum_{i=1} n(t_i - o_i)^2 \tag{8}$$

mit n die Anzahl der Muster, die dem Netzwerk gegeben werden, t_i der gewünschte bzw. erwartete Ausgabewert, o_i der errechnete Ausgabewert. Das Ziel des *Backpropagation*-Algorithmus ist nun die Minimierung der Verlustfunktion L. Dabei sollen die Gewichtungen der Neuronenverbindungen in Abhängigkeit auf die Abweichung der Ausgabe verändert werden. Also wird die Abweichung von der Ausgabe zur Eingabe zurück propagiert, weshalb auch das Verfahren als *Backpropagation* bezeichnet wird.
Die Minimierung der Abweichung geschieht durch Differentation der Verlustfunktion L. Die partielle Ableitung bzw. Gradient der Funktion L erhält man durch das Anwenden der Kettenregel:

$$\frac{\partial L}{w_{ij}} = \frac{\partial L}{\partial o_j} \frac{\partial o_j}{\partial y_j} \frac{\partial y_j}{\partial w_{ij}} \tag{9}$$

Die Änderung des Gewichts kann dann mit der folgenden Formel ausgedrückt werden:

$$\Delta w_{ij} = -\eta \frac{\partial L}{\partial w_{ij}} = -\eta \delta_j o_i \tag{10}$$

$$\text{mit} \quad \delta = \left\{ \begin{array}{ll} \varphi'(y_j)(o_j - t_j) & \text{falls j Ausgabeneuron ist,} \\ \varphi'(y_j) \sum\limits_k \delta_k w_{jk} & \text{falls j verdecktes Neuron ist.} \end{array} \right\} \tag{11}$$

wobei Δw_{ij} die Änderung des Gewichts w_{ij} der Verbindung von Neuron i zu Neuron j,
η eine feste Lernrate,
δ_j das Verlustsignal des Neurons j,
t_j die erwartete Ausgabe der Neurons j,
o_i die Ausgabe des Neurons i,
o_j die errechnete Ausgabe des Neurons j und
k der Index der nachfolgenden Neuronen von j sind.

Die Variable δ_j geht dabei auf die Unterscheidung der Neuronen ein: Liegt das Neuron in einer verdeckten Ebene, so wird seine Gewichtung abhängig von der Abweichung der Ausgabe geändert, die die nachfolgenden Neuronen erzeugen, welche wiederum ihre Eingaben aus dem betrachteten Neuron beziehen.
Die Bezeichnung *Backproagation* kommt daher, dass man Gradienten im *reverse mode* berechnet, was für den Fall eines neuronalen Netzwerks mit N Eingaben und M Ausgaben, bei dem $N \gg M$ gilt, massiv viel effizienter ist als, als der *forward mode*.

4.3 Verlustfunktion

Nach der Berechnung der Ausgabewerte wird verglichen, wie stark die von dem Netzwerk prognostizierten Ausgabewerte von dem tatsächlich erwarteten Wert abweicht. Eine Verlust-Funktion berechnet also die Abweichung zwischen den beiden Werten. Aus der Verlust-Funktion können wir die Gradienten ableiten, die zum Aktualisieren der Gewichte verwendet werden. Der Durchschnitt über alle Verluste ergibt die Kosten. Es gibt viele Funktionen, um den Verlust basierend auf dem prognostizierten und tatsächlichen Wert je nach Problem zu ermitteln.
Das *cross entropy loss* ist die am häufigsten verwendete Verlustfunktion für Klassifizierungsprobleme. Das *cross entropy loss*, der dann später verwendet wird, lässt sich ausdrücken als:

$$\text{loss}(x, y) = -\sum x \log y \tag{12}$$

Das *cross entropy* ist ein Maß für die Differenz zwischen zwei Wahrscheinlichkeitsverteilungen. Man möchte die Differenz zwischen der Wahrscheinlichkeitsverteilung y berechnen, die durch den Datengenerierungsprozess (das erwartete Ergebnis) erzeugt wird, und der Verteilung x, die durch unser Modell dieses Prozesses dargestellt wird. Die Differenz zwischen beiden Verteilungen x und y wird dann als Verlust (engl. *Loss*) bezeichnet. Sie steigt exponentiell an, je mehr die Vorhersage vom tatsächlichen Ergebnis abweicht.
Das Cross-Entropy-Funktion hat eine Vielzahl von Varianten, von denen der häufigste Typ die ***B**inary **C**ross **E**ntropy* (BCE) ist. Der BCE-Verlust wird hauptsächlich für binäre Klassifizierungsmodelle verwendet; das heißt, Modelle mit nur zwei Klassen.

4.4 Hyperparameter

Ein neuronales Netzwerk beinhaltet eine große Menge an wählbaren und veränderlichen Parameter. Hyperparameter sind sich unverändernde Parameter die festgesetzt werden müssen, bevor das Netzwerk trainiert wird. Je nach Art eines Netzwerkes kann die Anzahl der Parameter sowie deren Einfluss auf das Training stark variieren. Die Initialisierung der Hyperparameter spielt beim Trainieren eines neuronalen Netzwerkes

eine sehr große Rolle, denn schon eine Veränderung von einen der vielen Parameter kann die Genauigkeit und Effizienz des Netzwerks sehr stark verbessern oder aber auch verschlechtern. Zunächst möchte man auf die Hyperparameter eingegehen, die für alle Netzwerke allgemein grundlegend sind:

Lernrate (*learning rate*): Dieser Hyperparameter bezieht sich auf den Schritt der Backpropagation, wenn Parameter gemäß einer Optimierungsfunktion aktualisiert werden. Grundsätzlich stellt es dar, wie wichtig die Änderung des Gewichts nach einer Neukalibrierung ist. Man möchte so gut wie möglich die Abweichung (Loss) minimieren, das erfolgt idealerweise dadurch, dass die Gewichte w in die Richtung des Minimums verschoben werden. Die korrespondierende Funktion ist die Gleichung (10). Das η in der Gleichung ist das *learning rate* und es teilt dem Algorithmus mit, wie wichtig der Einfluss des Gradienten auf das Gewicht sein soll. Das Problem eines kleinen η besteht darin, dass das NN sehr langsam konvergiert (falls es konvergiert), und wir könnten auf das Problem des sogenannten „verschwindenden Gradienten“ stoßen. Auf der anderen Seite, wenn η sehr groß ist, verfehlt das Risiko das Minimum und tritt in das Szenario „explodierender Gradient“ ein.

Epoche: Sie stellt dar, wie oft das Algorithmus über den gesamten Datensatz trainieren soll. Eine Epoche bedeutet, dass jedes Element der Daten im Trainingsdatensatz Gelegenheit hatte, durch das Modell zu gehen. Eine Epoche besteht normalerweise aus mehreren Batches.

Batch: Die Größe des Batches ist ein Hyperparameter, der die Anzahl der zu bearbeitenden Datenelemente definiert, bevor die internen Modellparameter wie die Gewichte aktualisiert werden. Da eine Epoche zu groß ist, um sie auf einmal in den Computer einzugeben, teilt man sie in mehrere kleinere Batch auf.

Aktivierungsfunktion: Die Wahl einer Aktivierungsfunktion kann auch Einfluss auf den Optimierungsprozess haben. Eine genaue Erklärung über die Aufgabe sowie die Arten der Aktivierungsfunktionen sind in Abschnitt Abschnitt 4.1 zu finden.

4.5 Multilayer Perceptron

Ein **M**ulti**l**ayer **P**erceptron (MLP) ist ein *Feedforward ANN*, welches aus mehrere Ebenen von Neuronen besteht. Ein MLP besteht aus mindestens drei Ebenen: Zunächst gibt es die Eingabe-Ebene und die Ausgabe-Ebene. Der wesentliche Unterschied zu anderen neuronalen Netzwerken ist dann eine oder mehrere weitere Ebenen zwischen der Eingabe- und Ausgabe-Ebene, die Ebenen zwischen Eingabe- und Ausgabe-Ebene werden als versteckte Ebenen (engl. *hidden layers*) bezeichnet. Da MLPs vollständig verbunden sind, verbindet sich jedes Neuron in einer Ebene mit einem bestimmten Gewicht mit jedem Neuron in der folgenden Ebene. Also ist ein MLP eine Aufstapelung von mehreren vollständig verbundenen Neuron-Ebenen, jede Ebene gibt ihre Information weiter an die darüber liegende Ebene, bis man an die Ausgabe-Ebene erreicht und Ausgaben generieren. Jede Eingabe beeinflusst jedes Neuron in der versteckten Ebene, und jede dieser wiederum beeinflusst jedes Neuron in der Ausgabe-Ebene.

Die Berechnungen, die an jedem Neuron in der Ausgangs- und verborgenen Ebene stattfinden, sind wie

folgt:

$$\vec{o}(x) = \vec{G}(\vec{b}_2 + \vec{W}_2\vec{h}(x)) \tag{13}$$

$$\vec{h}(x) = \vec{s}(\vec{b}_1 + \vec{W}_1 x) \tag{14}$$

mit Bias Vektoren $\vec{b}_1$, $\vec{b}_2$; Gewichtungsmatrix $\vec{W}_1$, $\vec{W}_2$ und die Aktivierungsfunktionen $\vec{G}$ und $\vec{s}$. In der Eingabeebene ist für jede Eingabevariable ein Neuron plus zusätzlich ein sogenanntes Bias-Neuron. Dieses hat immer den Ausgabewert Eins. Mithilfe eines solchen Neurons, kann der Ausgabewert der weiteren Neuronen verschoben werden, indem die Gewichtung dieses Neurons verändert wird. Dies liegt darin begründet, dass die Ausgabe eines Neurons eine gewichtete Linearkombination aller Eingänge ist. In jeder versteckten Ebene ist ein weiteres Bias-Neuron enthalten. Die Ausgabe-Ebene besteht aus einem einzelnen Neuron, welches die Gesamtausgabe der neuronalen Netzwerks darstellt.

Die Neuronen im MLP werden mit dem *Backpropagation*-Lernalgorithmus trainiert. MLPs sind so konzipiert, dass sie jede stetige Funktion approximieren und Probleme lösen können, die nicht linear trennbar sind. Die wichtigsten Anwendungsfälle von MLP sind Musterklassifikation, -erkennung, -vorhersage und -näherung.

4.6 Convolutional Neural Network

Ein ***C**onvolutional **N**eural **N**etwork* (CNN) ist eine Klasse von künstlichen neuronalen Netzwerken, die am häufigsten zur Analyse visueller Bilder verwendet werden. CNN ist eine regularisierte Version von *Multilayer Perceptrons* (MLP). Da MLP eine vollständig verbundenes Netzwerk ist, ist auch jedes Neuron in einer Ebene mit allen anderen Neuronen in der nächsten Ebene verbunden. Diese vollständige Verbindung kann oft dazu führen, dass es zu einer Überanpassung des Modells an die Daten kommt. Für die Regularisierung oder Vermeidung von Überanpassung werden beim MLP typischerweise die Parameter während des Trainings verändert oder einzelne Verbindungen zwischen Neuronen gelöscht. Dagegen verwendet CNN einen anderen Ansatz zur Regularisierung: Sie nutzen das hierarchische Muster in Daten und stellen Muster mit zunehmender Komplexität zusammen, indem sie kleinere und einfachere Muster verwenden, die in ihre Filter eingeprägt sind. Das bedeutet, dass ein CNN im Vergleich zu einem Standard-MLP eine komplexere Struktur hat.

Ein CNN besteht zunächst genauso wie beim MLP aus einer Eingabe-Ebene, Ausgabe-Ebene und mehreren versteckten Ebenen. In einem CNN umfassen die versteckten Ebenen die Neuronen-Ebene, die Faltungen (engl. convolution) durchführen. Diese Ebene wird als *convolutional layer*[19] bezeichnet. Der Name *Convolutional Neural Network* weist also darauf hin, dass das Netzwerk eine mathematische Operation namens *convolution*, auf Deutsch „Faltung“, verwendet. *Convolutional Networks* sind eine spezialisierte Art von neuronalen Netzwerken, die in mindestens einer ihrer Ebenen eine Faltung anstelle einer allgemeinen Matrixmultiplikation verwenden. Der Aufbau eines typischen *Convolutional Neural Network*s ist als Skizze in der Abb. 3 zusammengefasst.

Ein *Convolutional Layer* korreliert die Eingabe und den Kernel und fügt eine skalare Abweichung hinzu, um eine Ausgabe zu erzeugen. Die beiden Parameter einer *Convolutional Layer* sind der Kernel/Filter (Faltungsmatrizen) und der skalare Bias. In der Regel liegt die Eingabe als zwei- oder dreidimensionale Matrix (z.B. die Pixel eines Graustufen- oder Farbbildes) vor. Dementsprechend sind die Neuronen im *Convolutional Layer* angeordnet. Die Aktivität jedes Neurons wird über eine diskrete Faltung (daher der Name *Convolutional Layer*) berechnet. Dabei wird schrittweise eine vergleichsweise kleine Faltungsmatrix (Filterkernel) über

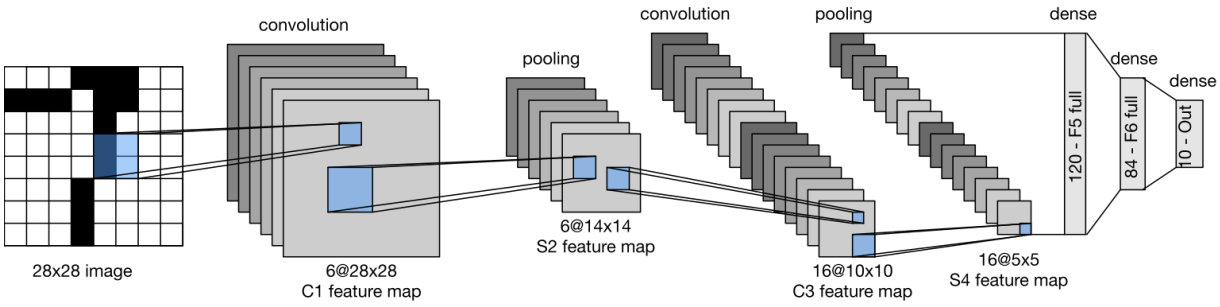


Abbildung 3: Die Architektur eines typischen *Convolutional Neural Network* Modells *LeNet* und sowie der Datenfluss in *LeNet* dargestellt. Die Eingabe ist eine handgeschriebene Ziffer, die Ausgabe eine Wahrscheinlichkeit über 10 mögliche Ergebnisse. [19]

die Eingabe bewegt. Die Eingabe eines Neurons im *Convolutional Layer* berechnet sich als inneres Produkt des Filterkernels mit dem aktuell unterliegenden Bildausschnitt. Dementsprechend reagieren benachbarte Neuronen im *Convolutional Layer* auf sich überlappende Bereiche (ähnliche Frequenzen in Audiosignalen oder lokale Umgebungen in Bildern).

Beim Trainieren von Modellen auf der Grundlage von *Convolutional Layer* initialisiert man die Kernel normalerweise zufällig, genau wie bei einem vollständig verbundenen Netzwerk. Die Neuronen in einer Ebene sind nicht mit der gesamten nächsten Ebene verbunden, sondern nur mit den Neuronen in der Umgebung. Das bedeutet, dass die Auswirkung der Neuronen nur lokal begrenzt ist und wenn die lokale Region klein ist ist, wird der Unterschied zum vollständig verbundenen Ebenen umso größer. Das *Convolutional Layer* enthält auch mehrere Filter innerhalb des Layers, und dies ist ein vordefinierter Hyperparameter. Die Anzahl der Filter innerhalb einer Ebene gibt die Tiefendimension des Ausgabevolumens an, die von dem *Convolutional Layer* als Eingabe für die nächste Ebene erstellt werden. Die Ausgabe des *Convolutional Layers* wird dann als *feature map* bezeichnet, da sie als gelernte Repräsentationen (*features*) in den räumlichen Dimensionen (z.B. Breite und Höhe) für die nachfolgende Ebene angesehen werden kann. Das Stapeln der *feature maps* für alle Filter entlang der Tiefendimension bildet das vollständige Ausgabevolumen des *Convolutional Layer*. Jeder Eintrag im Ausgabevolumen kann somit auch als Ausgabe eines Neurons interpretiert werden, das einen kleinen Bereich in der Eingabe betrachtet und mit Neuronen in derselben *feature map* die Gewichtungsparameter teilt (*shared weights*). Aus den *shared weights* folgt unmittelbar, dass Translationsinvarianz[19] eine inhärente Eigenschaft von CNNs ist. Diese *shared weights* ermöglichen das Erkennen von Merkmalen in den Daten unabhängig von ihrem Standort auf der Eingabeschicht. Dies macht es besonders nützlich in der Bildanwendung, da Objekte immer erkannt werden können, egal wo auf dem Bild sie erscheinen.

Ein weiteres wichtiges Bestandteil eines CNNs ist das *Pooling* bzw. *Pooling Layer*, welches eine nicht lineare *down-sampling* ist. Das Ziel des *Pooling* ist es, überflüssige Information zu verwerfen, um die die räumliche Größe des Bildes, sowie die Anzahl der Parameter und Berechnung im Netzwerk zu reduzieren und damit auch die Überanpassung zu kontrollieren. Der Hauptgrund für die Verwendung von *Pooling Layers* lässt sich auf zwei Punkte fassen. Zunächst kann die Verwendung von *Pooling* die Leistung des Netzwerks erhöhen,

denn *Pooling* erlaubt die Größe des Netzwerks massiv zu verringern. Der zweite Punkt ist das Hervorheben von wichtigen Informationen. Die verschiedenen *Pooling* Arten wie *Maxpooling* und *Average Pooling* haben alle gemeinsam, dass sie das '*Signal to Noise*' ratio der Informationen im Netzwerk quasi erhöhen, indem sie weniger relevante Information loswerden.
Wie *Convolutional Layer* bestehen Pooling-Operatoren aus einer Filter mit fester Form, das entsprechend seiner Schrittweite über alle Regionen in der Eingabe geschoben wird und eine einzelne Ausgabe für jede Position berechnet, die von dem Filter mit fester Form durchquert wird (manchmal als *pooling-window* bezeichnet). Im Gegensatz zur *cross-correlation* Berechnung der Eingaben und Kernel in dem *Convolutional Layer* enthält das *Pooling Layer* jedoch keine Parameter (es gibt keinen Kernel). Stattdessen sind Pooling-Operatoren deterministisch und berechnen normalerweise entweder den Maximal- oder den Durchschnittswert der Elemente im Pooling-Filter. Diese Operationen werden als *Maximum-Pooling* (kurz Max-Pooling) und *Average-Pooling* bezeichnet. Gegeben sei ein Bild der Größe N x N, eine sehr gängige Form des Max-Poolings ist ein Layer mit Filtern der Größe 2 x 2. Bei einem Downsampling mit einem 2x2-Filter über das gesamte Bild, an jeder Stelle, auf die der Pooling-Filter trifft, berechnet er den maximalen oder durchschnittlichen Wert des Eingabe-Subtensors im Filter. Der Effekt ist, dass hervorstechende Eigenschaften in den Daten propagiert werden, während weniger wichtige Eigenschaften des Bildes entfernt werden. Man sieht, dass das Bild sowohl nach dem *Pooling Layer* als auch nach *Convolutional Layer* nicht mehr die originale Pixelgröße N x N besitzt. Wir können die Operation ändern, um eine gewünschte Ausgabeform zu erreichen, indem wir die Eingabe auffüllen (Padding) und die Schrittweite der Filterverschiebung anpassen (Stride).

Ein CNN besteht also zunächst aus einer Eingabe-Ebene, welche dann direkt mit mehreren *Convolutional Layers* verbunden sind. Die die Eingabematrizen werden hier je nach Gewichtung gefaltet. Dem *convolutional layer* folgen dann *Pooling Layers* und es werden vergleichsweise weniger relevante Informationen verworfen. Zum Schluss wird an den *Pooling Layers* die vollständig verbundene Ebene angehängt, um die am Ende zwei Ausgaben zu generieren und sie bilden gemeinsam das *Convolutional Neural Network*.

5 Klassifikation von Signalen

Eine der Hauptaufgaben des CAST-, BabyIAXO-Experiments besteht darin, die photonartigen Signale eines möglichen Axions von allen anderen Untergrundsignalen zu trennen, die am Detektor aufgenommen wurden. Aufgrund der selten erwarteten Vorkommnissen von Axionphotonen sowie der hohen Untergrundrate durch kosmischer Strahlung und radioaktiven Untergrund des Detektormaterials usw. muss eine effiziente Methode entwickelt werden, um eine gute Trennung von signal- und untergrundartigen Ereignissen zu erreichen. Durch Anwendung eines neuronales Netzwerks wird nun erwartet, dass man eine sehr gute Klassifikationsmethode entwickeln kann, um die beiden Arten von Ereignissen genau und effizient zu trennen.

5.1 Vorbereitung und Verarbeitung der Daten

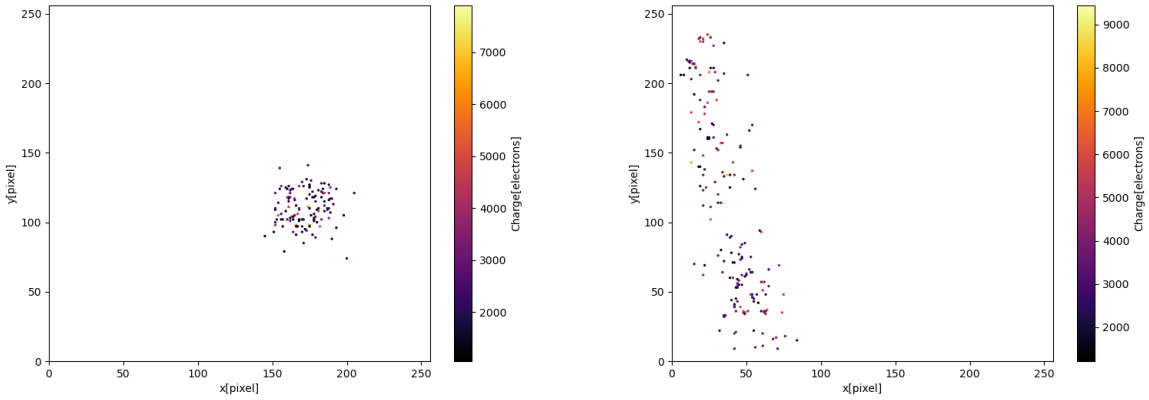
Um ein neuronales Netzwerk zur Klassifikation von signal- und untergrundartigen Ereignissen zu bauen, benötigt man die entsprechenden zu klassifizierenden Daten, um das Netzwerk passend zu trainieren. Also muss zunächst mit dem Detektor jeweils ein Datensatz mit signalartigen Ereignissen und einer mit untergrundartigen Ereignissen aufgenommen werden. Die Untergrundmessungen wurden direkt aus dem Detektor entnommen, welcher die Untergrunddaten aufzeichnet, während er nicht auf die Sonne ausgerichtet ist. Um signalartige Ereignisse zu erhalten, werden Kalibrationsmessungen mit verschiedenen Quellen in entsprechenden Energiebereich durchgeführt. Dazu wird mit einer Röntgenröhre im Vakuum Elektronen auf das Target beschleunigt, wodurch Bremsstrahlungen erzeugt werden. Die in Form einer Bremsstrahlung erzeugten Röntgenstrahlungen tragen eine Energie der zugehörigen Röntgenlinie, welche sich je nach Targetmaterial unterscheidet. Diese Röntgenstrahlungen bzw. Röntgenphotonen werden dann mit dem Gasdetektor detektiert. Da durch die Axionen erzeugte Photonen eine Energie von 3-4 keV tragen, verwendet man Kalibrationstargets, die Röntgenstrahlungen in ähnlichen Energiebereich erzeugen können. Die verwendeten Targets für die Kalibration sind: Silber (Ag), Aluminium (Al), Kohlenstoff (C), Kupfer (Cu) (0,9 kV), Kupfer (2 kV), Kupfer-Nickel (Cu-Ni), Moscovium-Chrom (Mn-Cr), Titan (Ti)

Da zur Auslese des Detektors ein GridPix verwendet wird, erhält man von dem Detektor dementsprechend als Rohdaten Pixel Daten mit 256x256 Elektronladungen für jedes Ereignis. Jede Elektronladung wird dann aufgezeichnet. In der Abb. 4 sind die Bilder eines typischen Röntgenphotonereignisses und eines Untergrundereignisses zu sehen.

Für das CNN können die rohen Pixel Daten direkt verwendet werden, in dem sie als ein Bild interpretiert werden.

Für das MLP müssen die Rohdaten, die aus dem Chip ausgelesen werden, zunächst weiter verarbeitet werden. Es werden die geometrischen Eigenschaften des Pixelbildes aus dem Detektor-Chip berechnet und diese Werte der geometrischen Eigenschaften dienen dann als Eingangsdaten eines MLPs.
Folgende geometrische Eigenschaften werden dabei berechnet und für das MLP berücksichtigt:

- **Exzentrizität (eccentricity)**
- **Ladungsenergie (energyFromCharge)**
  Es gibt die Energie der Gesamtladung des Ereignisses an.



(a) Röntgenphoton

(b) Untergrund

Abbildung 4: (a) zeigt ein typisches Röntgen Ereignis. Es hat nur einen Punkt am Ursprung, die zufällige Streuung verursacht eine radial symmetrische Diffusion, wodurch eine sphärische Form erzeugt wird. (b) ist ein typisches Untergrund Ereignis. Es wird induziert, indem ein kosmisches Teilchen das Gasvolumen durchquert und Teilchen auf dem gesamten Weg ionisiert. Dies führt zu einer spurartigen Form, da die Diffusion nur orthogonal zur Bewegungsrichtung sichtbar ist.

- **kurtosis Longitudinal / kurtosis Transverse (Wölbung)**
  Unter Kurtosis versteht man die Steilheit einer Wahrscheinlichkeitsverteilung. Damit kann beschrieben werden, wie spitz oder flach der Gipfel der Verteilung ist. Es wird entlang der langen Achse (Tranverse) und entlang der kurzen Achse (Longitudinal) berechnet.

- **Length**
  Es gibt die Länge einer Spur entlang der langen Achse an.

- **skewness Longitudinal / skewness Transverse (Schiefe)**
  Skewness gibt die Schiefe einer Verteilung an. Sie beschreibt die Art und Stärke der Asymmetrie. Einmal wird die Skewness entlang der langen Achse (Tranverse) und dann entlang der kurzen Achse (Longitudinal) angeschaut.

- **Hits**
  Anzahl der aktiven Pixel im Cluster

- **RMS Longitudinal / RMS Transverse**
  RMS ist das quadratische Mittel und ist derjenige Mittelwert, der berechnet ist als Quadratwurzel des Quotienten aus der Summe der Quadrate der beachteten Zahlen und ihrer Anzahl. RMS wird einmal entlang der langen Achse (Tranverse) und dann entlang der kurzen Achse (Longitudinal) berechnet.

- **Fraction In Transverse RMS** Anteil aller Pixel, die innerhalb eines Kreises mit Radius des transversen RMS um das Zentrum des Clusters sind.

- **Rotation Angle (Rotationswinkel)**

5.2 Implementation/Bildung/Aufbau von neuronalen Netzwerke

Für die Klassifikation der Signal- und Untergrunddaten werden die neuronalen Netzwerke, die dafür verwendet werden sollen, aufgebaut. Da in dieser Arbeit einmal das MLP und das CNN betrachtet werden soll, werden beide Arten des Netzwerks separat gebaut und trainiert. Sowohl das MLP als das CNN wird in der Programmiersprache Python implementiert, verwendet wird dazu ein *open source machine learning framework*, basierend auf *Pyorch* Bibliothek. Mit Hilfe des *Pytorch framework* ist man in der Lage viele Funktionen, die für den Aufbau eines neuronalen Netzwerks notwendig sind, einfach als eingebaute Funktion verwenden ohne diese einzeln zu implementieren, wodurch man recht bequem und schnell auch kompliziertere Netzwerke aufbauen kann.

5.2.1 Einlesen und Verarbeiten der Daten

Für das Training sowie für die spätere Validierung der Netzwerke werden Daten verwendet, die in CERN aufgenommen wurden. Aufgrund von sehr seltenen Vorkommnissen der Axion Teilchen wird erwartet, dass am CAST aufgenommene Signaldaten zum aller größten Teil Untergrundereignisse sind. Damit das Netzwerk die Signal und Untergrundereignisse unterscheiden kann, benötigt man Trainingsdaten, die sowohl signalartige als auch untergrundartige Daten enthalten. Um signalartige Daten zu erhalten, wurden aus verschiedenen Quellen mit einer Röntgenröhre die Röntgenphotonen aufgenommen, sodass man Kalibrationsdaten mit unterschiedlichen Energien hat. Für die Untergrunddaten wurde dann aus der Untergrundmessung am CAST entnommen.

Sowohl die Kalibrations- als auch Untergrunddaten sind in Form einer HDF5[20]-Datei gespeichert und zuerst müssen die Daten in das Programm, wo das Netzwerk dann ausgeführt wird, eingelesen werden. Diese erfolgt einfach über das h5py-Paket, sodass man die Daten einer HDF5-Datei auslesen und beispielsweise in einem numpy-array speichern kann. Zur Rekonstruktion der verwendeten Daten sind die entsprechenden Methoden und Code in dem Github Repository *TimepixAnalysis*[1] zu finden. Der nächste Schritt, der vorgenommen werden muss, ist das Schneiden der Kalibrationsdaten. Wenn man die Daten der geometrischen Eigenschaften als Histrogramm aufzeichnet, erkennt man, dass die Daten eine statistische Verteilung annimmt. Alle Ereignisse, die sehr stark von der Verteilung abweichen, sollen entfernt werden, damit das Netzwerk nicht mit Daten trainiert werden, die wenig aussagekräftig sind. Somit wird die Wahrscheinlichkeit, dass das Netzwerk später nach dem Training ein Untergrundereignis als Photonsignal klassifiziert, verringert. Für das Schneiden der Daten wird *Pandas DataFrame*[2] eingesetzt. Ein *DataFrame* hat einen Zeilen- und einen Spalten-Index. In den Spalten werden die einzelnen geometrischen Eigenschaften eingetragen und die Zeilen entsprechen dann den Ereignissen. Für jeden Spalt bzw. Eigenschaft wird dann eine Schnittbedingung definiert und eine Zeile bleibt nur dann erhalten, wenn in allen Spalten die jeweilige Bedienung erfüllt ist. Alle anderen Zeilen bzw. Ereignisse, die mindestens in einem Spalt die Schnittbedingung nicht erfüllen, werden verworfen. Für die Schnittbedingung wird wie bereits erwähnt ein Histogramm für jede geometrische Eigenschaft erstellt und für jede Eigenschaft die untere und obere Grenze so gewählt, dass nur der Bereich der Verteilung erhalten bleibt den restlichen Bereich außerhalb der Verteilung ausschneidet.

[1] https://github.com/Vindaar/TimepixAnalysis

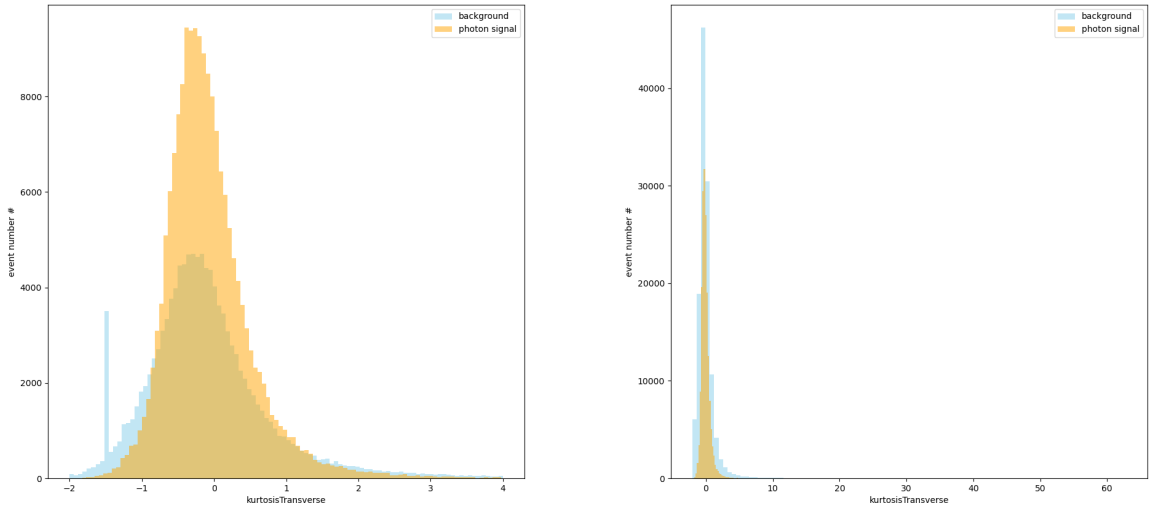
[2] Pandas DataFrame ist eine Struktur, die zweidimensionale Daten und die entsprechenden Bezeichnungen enthält. DataFrames werden häufig in der Datenwissenschaft, im maschinellen Lernen, im wissenschaftlichen Rechnen und in vielen anderen datenintensiven Bereichen eingesetzt.

Die verwendeten Schnittwerte sind in Tabelle 1 eingetragen:

geom. Eigenschaft	Schnittwert unten	Schnittwert oben
Exzentrizität	1	10
Ladungsenergie	0	15
kurtosis Longitudinal	-3	5
kurtosis Transverse	-2	4
Length	0	17
skewness Longitudinal	-2	2
skewness Transverse	-2	2
Fraction In Transverse RMS	0	0,5
RMS Longitudinal	0	5
RMS Transverse	0	2
Rotation Angle	0	3.5
Hits	0	500

Tabelle 1: Die Schnittwerte der geometrischen Eigenschaften auf die Kalibrations- und Untergrunddaten

Beispielsweise ist die Verteilung der geometrischen Eigenschaft **kurtosisTransverse** von Silber (Ag) ohne und mit dem Schnitt in Abb. 5 abgebildet:



(a) Ohne Schnitt

(b) Mit Schnitt

Abbildung 5: Signal- und Untergrundverteilung der geometrischen Eigenschaft 'kurtosis Transverse' mit und ohne Schnittanwendung auf die Kalibrationsdaten von Silber

Die Histogramme für die restlichen geometrischen Eigenschaften der Silberquelle lassen sich in Anhang Abb. 31 bis 34 finden. Dieser Schnitt wurde neben Silber auch für alle vorhandenen Kalibrationsquellen durchgeführt, wobei für alle Quellen, die identischen Schnittwerte eingesetzt wurden.

Nun müssen die Daten so umformiert werden, dass sie dem Netzwerk gegeben werden können. Zuerst muss zu jedem Ereignis ein sogenanntes Label angehängt werden. Anhand des Labels kann das Netzwerk identifizieren, ob ein Ereignis in Wirklichkeit ein Signal oder Untergrund ist. Mit Hilfe des Labels kann dann die Verlustfunktion und auch die Klassifikationsgenauigkeit eines Netzwerks später ermittelt werden. Da man

hier nur eine binäre Klassifikation hat, kann jedem Ereignis entweder das Label 0 oder 1 zugeordnet werden, je nach dem welches Label man für signalartige Ereignisse wählt. Für diesen Vorgang wir eine eigene Klasse 'Dataset' definiert, welche als Eingabe einen Datensatz und ein Label nimmt. Die Klasse addiert dann bei allen Ereignissen des Datensatzes das mitgebene Label hinzu, so wird ein Ereignis ursprünglich aus einem 1D-Array mit den geom. Eigenschaften als Elemente zu einem 2D-Tensor mit dem 1D-Array als das erste Element und in dem zweiten Eintrag das Label. Angenommen man gibt der Klasse einen Signaldatensatz mit 10 geom. Eigenschaften und das Label 0 mit, dann erhält man einen Datensatz zurück, in der jedes Element bzw. Ereignis ein 2D-Tensor in der Form (10,1) ist. Analog werden dann durch die Klasse alle Ereignisse eines Untergrunddatensatzes mit dem Label 1 markiert.
Dann läuft der Datensatz zum Schluss durch eine DataLoader Klasse aus dem Pytorch Bibliothek. Die DataLoader Funktion serviert die Daten in mehreren Batches (Stapel) auf, wobei die Datenanzahl innerhalb eines Batches als Batchgröße mitgegeben wird.

5.2.2 Neuronale Netzwerke

Multi Layer Perceptron (MLP): Nun muss das Modell definiert werden und seine Eingaben und Parameter (Gewicht und Bias) mit seinen Ausgaben in Beziehung setzen. Für das Modell wurde eine MLP-Klasse definiert, für die einzelne Ebene werden die eingebaute Module aus dem Pytorch-Bibliothek verwendet. Das Modell hat insgesamt vier Ebenen mit einer Eingabe-Ebene und drei weiteren vollständig verbundene Ebenen. Die Eingabe-Ebene bekommt zuerst 10 Eingabewerte, wobei die Anzahl der Eingaben der Anzahl der geometrischen Eigenschaften entsprechen soll. Die ersten zwei vollständig verbundenen Ebenen sind die versteckte Ebene (*hidden layer*), die jeweils 500 und 300 versteckte Einheiten bzw. Neuronen enthält und die ReLU-Aktivierungsfunktion anwendet. Die letzte Ebene ist dann die Ausgabe-Ebene, welche am Ende zwei Ausgaben generiert.
Für die Berechnung von Verlust (loss) wird eine aus dem torch Bibliothek importierte Funktion *BCEWithLogitsLoss* verwendet. Die Loss-Funktion *BCEWithLogitsLoss* ist eine Mischung aus Sigmoid- und *cross entropy loss* Funktion, welche für eine binäre Klassifikationsprobleme geeignet ist.

Convolutional Neural Network (CNN): Hier wird nach dem sogenannten *LeNet*-Modell[21] angerichtet, das zum eines der frühsten *Convolutional Neural Network* (CNN) Strukturen gehört. Grob gesehen besteht ein LeNet aus zwei Teilbereichen: Der erste Teil besteht aus convolutional Block, der zweite Teil basiert dann auf einem Block von mehreren vollständig verbundenen Ebenen, um schließlich eine Ausgabe-Ebene mit benötigter Anzahl an Ausgabeneuronen zu erhalten. Eine beispielhafte Architektur des *LeNet* lässt sich in der Abb. 3 finden.
Ebenfalls wie beim *Multilayer Perceptron* wird hier das Modell aus eingebaute Funktionsmodule von Pytorch-Bibliothek verwendet. Der erste Block aus drei *Pooling Layers* und drei *Convolutional Layers*, wobei die Layers beginnend mit einem *Pooling Layer* immer abwechselnd angeordnet sind. Verwendet wird das *Maximum-Pooling Layer* mit einer Filtergröße von 2x2 und stride 2. Jedes *Convolutional Layer* verwendet einen Kernel mit der Größe 5x5 und eine Aktivierungsfunktion. Diese Ebenen (Layers) bilden räumlich angeordnete Eingaben auf eine Reihe von zweidimensionalen *feature maps* ab, wodurch üblicherweise die Anzahl der Kanäle erhöht werden kann. Allerdings haben die Daten, die hier verwendet werden nur die elektrische Ladungen als Graustufen-Einträge und keine Farben (RGB-Kanäle), sodass die Anzahl der Kanäle immer auf 1 gehalten wird.

Um die Ausgabe vom Convolution Block an den Block mit vollständig verbundenen Ebenen zu übergeben, müssen die Daten abgeflacht werden. Das heißt, man nimmt diese vierdimensionale Eingabe und wandelt sie in die zweidimensionale Eingabe um, die von vollständig verbundenen Ebenen erwartet wird. Wie bereits in der Implementation von MLP eingegangen: Die zweidimensionale Darstellung, die in die vollständig verbundene Ebene eingeht, verwendet die erste Dimension, um die flache Vektordarstellung jedes Ereignis mit den geometrische Eigenschaften zu geben, und die zweite, um die Ereignisse im Batch mit entsprechendem Label zu indizieren.
Für die Verlust-Funktion wird wie beim MLP eine Mischung aus Sigmoid- und *cross entropy loss* Funktion gewählt, da es beim CNN ebenfalls um ein binäres Klassifikationsproblem handelt. Es wird die Klasse *BCEWithLogitsLoss* aus dem *torch*-Bibliothek verwendet.

6 Training der neuronalen Netzwerke / Bestimmung der Parameter im Hinblick auf die Optimierung der Netzwerke

Damit die Verbindungen in künstlichen neuronalen Netzen passend geknüpft werden, um die Aufgabenstellung zu lösen, müssen die Netze zunächst trainiert werden. Durch das Trainieren von Netzwerk werden die Modellparameter wie die Gewichte der einzelnen Neuronenverbindungen so optimiert, damit man eine möglichst hohe Genauigkeit bei der Klassifikation erreicht.
Bei neuronalen Netzen unterscheidet man typischerweise zwischen einer Trainingsphase und einer Testphase. In der Trainingsphase lernt das neuronale Netz anhand der vorgegebenen Daten. Dementsprechend werden in der Regel die Gewichte zwischen den einzelnen Neuronen modifiziert.
In der Testphase werden hingegen keine Gewichte verändert. Statt dessen wird hier auf Grundlage der bereits modifizierten Gewichte aus der Trainingsphase untersucht, ob das Netz etwas gelernt hat. Zum Schluss kommt dann die Validierungsphase, wenn das Netzwerk komplett zu Ende trainiert wurde und sich keine Änderungen oder Optimierungen mehr vornimmt.

Dementsprechend werden die vorhandenen Daten, die verwendet werden sollen, in Trainings-, Test- und Validierungsdaten aufgeteilt, wobei zum größten Teil aller Datenmenge für das Training eingesetzt wird. Das Netzwerk ist so implementiert, dass nach jeder Epochendurchlauf die Klassifikationsgenauigkeit von Trainingsdaten und Testdaten bestimmt werden. Wenn das Netzwerk alle Epochen durchlaufen hat und der Trainingsprozess beendet ist, werden die Validierungsdaten durch das vollständig trainierte Netzwerk klassifiziert und wiederum die Genauigkeit der Klassifikation (validation accuracy) bestimmt.

Außerdem betrachtet man noch die Ausgabeverteilung der beiden Ausgabeneuronen, um die Klassifikation eines Netzwerks genauer untersuchen zu können. Die Ausgabeverteilung zeigt die rohen Ausgabewerte, bevor sie eindeutig zum Signal oder Untergrund klassifiziert werden. Je stärker die Verteilung der Signalereignisse mit der Verteilung der Untergrundereignisse überlappt, desto schwieriger bzw. ungenauer wird dann die Klassifikation, wodurch man anhand der Ausgabeverteilungen auf die Effizienz und Leistung eines Netzwerks schließen kann.

Für den Optimierungsgrad des Netzwerks wird zum einem an die Klassifikationsgenauigkeit und zum anderen an die Signal- und Untergrundverteilung der beiden Ausgabeneuronen orientiert und bewertet. Die Genauigkeit (Accuracy) des Netzwerks und die Ausgabeverteilung einer der beiden Ausgabeneuronen von Validierungsdaten sowie der Verlust-Wert (Loss) nach dem Training werden als Vergleichsmaßstab genutzt.

Um ein Netzwerk erfolgreich zu trainieren und die bestmögliche Optimierung zu erzielen, spielt die Wahl der Hyperparameter(Siehe Abschnitt 4.4) eine entscheidende Rolle. Je nach dem wie die Hyperparameter festgesetzt werden, kann das Trainingsergebnis ganz unterschiedlich werden. Also kann man sagen, dass die Hauptaufgabe beim Konstruieren und Trainieren eines Netzwerkes darin besteht, die passenden bzw. die optimalsten Hyperparameter zu finden. Hierbei sei es aber zu erwähnen, dass das Finden des optimalen Wertes für die einzelnen Hyperparameter sehr intuitiv ist. Das heißt, es ist sehr schwer die Parameterwerte auf logische und nachvollziehbare Art und Weise zu optimieren, sondern die Parameter werden intuitiv festgelegt und man versucht die einzelnen Parameter zu variieren, bis man einen optimalen Hyperparameter findet, bei der das Netzwerk die beste Leistung bringt. Ein weiteres Problem, auf das man hier stößt, ist die viele Anzahl der Hyperparameter und die Korrelation. Es ist nicht so, dass die verschiedenen Hyperparameter eine separierte Einwirkung auf das Netzwerk hat. Die Hyperparameter stehen in Korrelation zueinander und ein

Hyperparameter kann bei der Wahl eines anderen Parameters Einfluss haben, um den optimalsten Wert zu finden. Allerdings ist es sehr schwer zu sehen, wie sie miteinander korrelieren und es gibt auch kein fester Vorgang wie man die Hyperparameter bestimmt. Man versucht so oft wie möglich die Hyperparameter zu variieren und mit möglichst vielen Kombinationen das Netzwerk zu trainieren. Um diesen Bestimmungsprozess dennoch anschaulich in dieser Arbeit darzustellen, werden im voraus die Hyperparameter gefunden und festgesetzt, die im optimalen Bereich liegen. Diese werden dann als Standardparameter gesetzt und die einzelnen Hyperparameter werden nacheinander separat variiert, während für die anderen Hyperparameter immer die Standardwerte beibehalten werden. Somit kann man sehen, wie sich ein einzelner Hyperparameter auf das Verhalten des Netzwerkes auswirkt und möglicherweise kann das Netzwerk durch die zusätzliche Anpassung eventuell noch weiter optimiert werden.

6.1 Optimierung des Multilayer Perceptron (MLP)

Das Training des Netzwerks MLP wird auf dem CPU durchgeführt. Das Quellcode zum Training des *Multilayer Perceptron*s ist auf *Github Repository ANN*[1] verfügbar.
Die Hyperparameter des Multilayer Perceptrons (MLP) und die entsprechend festgelegten Werte sind zunächst in der Tabelle 2 angeordnet:

Hyperparameter	als Standard verwendeter Wert/Funktion
Epochenanzahl	100
Batchgröße	100
learning rate	0.05
Aktivierungsfunktion	ReLU-Funktion
Anzahl Inputvariablen/ verwendeter geom. Eigenschaften	10 (wird später im Abschnitt 7.1 separat behandelt)
Anzahl versteckter Ebenen/Einheiten (hidden layout)	2 versteckte Ebenen mit jeweils 500 und 300 Einheiten (hidden units)
Anzahl Trainingsdaten	140000

Tabelle 2: Auf der linken Seite der Tabelle sind die relevanten Hyperparameterarten des MLP, auf der rechten Seite sind die dazugehörige zum Trainieren verwendete Hyperparameter angeordnet.

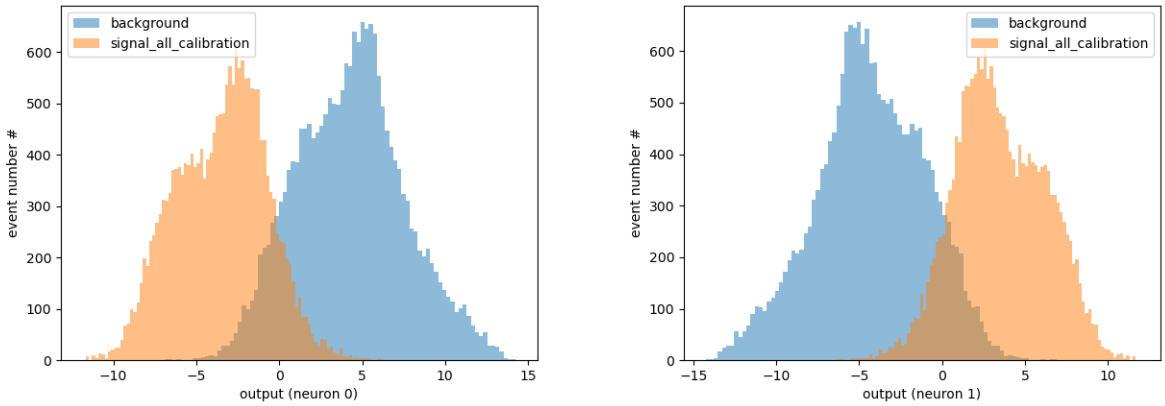
Es sei noch einmal zu erwähnen, dass diese Standardwerte nicht zufällig ausgesucht wurden, sondern sie wurden durch mehrfaches Training des MLPs als Optimalwert gefunden. Es werden in allen folgenden Tabellen die Genauigkeit mit 'acc' (Accuracy) abgekürzt dargestellt.
Die Ergebnisse der Klassifikationsgenauigkeiten sowie die Ausgabeverteilungen nach dem Training des MLPs mit den festgelegten optimalen Hyperparameter(siehe Tabelle 2) sind in der Tabelle 3 und Abb. 6 zu sehen:

Das Netzwerk besitzt zwei Ausgabeneuronen, und zwar Neuron 0 und Neuron 1, weshalb man auch insgesamt zwei verschiedene Ausgabeverteilungen erhalten, wobei die Verteilungen einfach nur spiegelverkehrt aussehen. In weiteren sollen nur die Ausgabeverteilungen des Ausgabeneuron 0 betrachtet und verglichen

[1] https://github.com/dongjinsuh/ANN/blob/main/train_mlp.py

loss	train acc	test acc	validierung acc
0,1154711	0,9015	0,8970	0,8977

Tabelle 3: Die Klassifikationsgenauigkeiten des MLP mit den Standard Hyperparameterwerte



(a) Neuron 0

(b) Neuron 1

Abbildung 6: Signal- und Untergrundverteilungen von Ausgabeneuronen mit Optimalparameter

werden.

Epochen: Es wird allgemein erwartet, dass die Genauigkeit des Netzwerks mit steigender Anzahl an Epochen ebenfalls steigt. Ab einer bestimmten Epochenzahl aber sollte sich die Genauigkeit sättigen und ab diesen Punkt wird keine signifikante Verbesserung der Genauigkeit nicht mehr erwartet. Zwar kann durch die größere Epochendurchläufe die Genauigkeit immer noch um ein kleines Stück erhöhen, allerdings ist die Steigung nur noch sehr klein und außerdem erhöht sich dann auch die Laufzeit des MLPs. Es wird also ab einem Punkt nicht mehr sehr effizient, für diese minimale Genauigkeitsverbesserung so viel Laufzeit zu investieren.
Um eine optimale Epochenzahl zu bestimmen, wird zu Beginn eine sehr große Epochenanzahl gewählt und die Genauigkeit sowie die Verluste werden nach jeder Epochendurchgang gespeichert und die Gesamtverläufe abgebildet. Die Verläufe für 400 Epochen sind in Abb. 7 und Abb. 8 zu sehen, die Ergebnisse nach dem Trainingsabschluss finden sich in der Tabelle 4.

Epochenzahl	loss	train acc	test acc	validierung acc
400	0,1008	0,9148	0,9033	0,9014
100	0,1155	0,9015	0,8970	0,8977

Tabelle 4: Die Klassifikationsgenauigkeiten nach dem Trainieren des MLP für verschiedene Epochenzahl

Man sieht auf jeden Fall, dass die Genauigkeit der Trainingsdaten für höhere Epochen weitersteigt und gleichzeitig der Verlustwert kontinuierlich sinkt. Allerdings fällt es auf, dass die die Klassifikationsgenauigkeit für die Testdaten ca. ab 100 Epochen stagniert und die Kurve komplett abflacht.
Vergleichen der Endergebinsse mit 100 Epochen bestätigt auch den Kurvenverlauf der Abb. 7 . Während die Trainingsgenauigkeit um mehr als 0.1 steigt, sieht man, dass die Genauigkeit für Test- und Validierungsdaten über die weiteren 300 Epochendurchläufe recht konstant geblieben sind. Für die Bestimmung

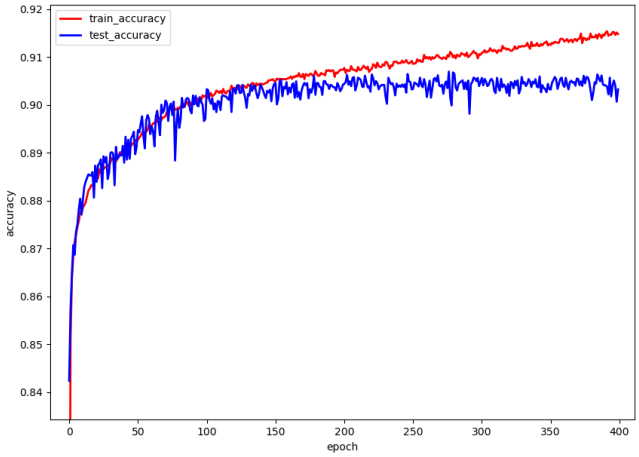


Abbildung 7: Klassifikationsgenauigkeit des MLP für Trainings- und Testdaten über 400 Epochen. Man sieht eine durchgehend steigende die Trainingsgenauigkeit, aber die Testgenauigkeit fängt an zu stagnieren und daher ergibt sich 100 ein guter Richtwert für die Epochenzahl

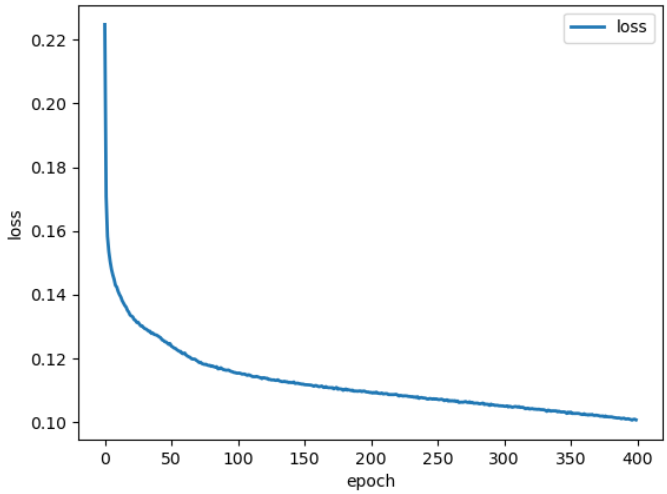


Abbildung 8: Verlustwert der Klassifikation mit MLP

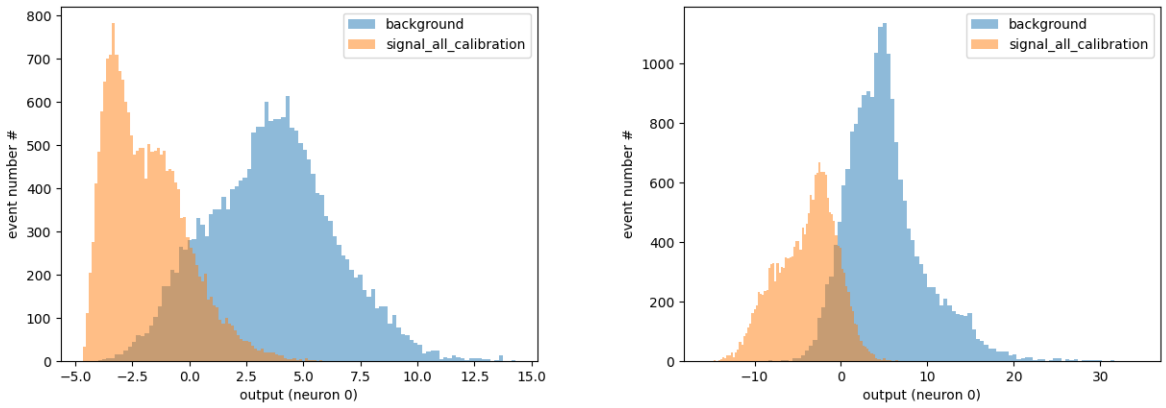
der Epochenzahl ist die Trainingsgenauigkeit etwas weinger aussagekräftig als die die Genauigkeit für Test- und Validierungsdaten, da das Netzwerk für die Optimierung an die Trainingsdaten anpassen, wodurch die Genauigkeit potentiell immer weiter steigen kann, wenn man größere Epochenzahl wählt. Dementsprechend ist der Verlustwert, welcher auch aus den Trainingsdaten berechnet wird, auch nur beschränkt bedeutsam, was die tatsächliche Leistung eines Netzwerks für reale Daten angeht. Daher wird hier auf die Test- und Validierungsgenauigkeit orientiert und die optimale Epochenzahl wird auf 100-150 gelegt.

Batchgröße: Für die Batchgröße wird zunächst 10 gewählt, das heißt, die Gewichte sowie die anderen internen Netzwerkparameter werden immer nach 10 durchgelaufenen Ereignissen aktualiersiert bzw. anhand der berechneten Gradienten angepasst. Dann wird die Batchgröße auf 100 und zum Schluss auf 1000 erhöht und die Eregbnisse verglichen, die in der Tabelle 5 eingetragen sind.

Batchgröße	loss	train acc	test acc	validierung acc
10	0,10659	0,8980	0,8970	0,8982
50	0,1138	0,9062	0,9035	0,9002
100	0,1154711	0,9015	0,8970	0,8977
200	0,1217	0,8993	0,8992	0,8949
1000	0,14148	0,8786	0,8748	0,8451

Tabelle 5: MLP: Variation des Hyperparameters Batchgröße

Die Ausgabeverteilungen des Ausgabeneurons (Neuron 0) für Batchgröße 10 und 1000 ist in Abb. 9 zu sehen.



(a) Batchgröße 1000

(b) Batchgröße 10

Abbildung 9: MLP: Signal- und Ausgabeverteilung von Ausgabeneuron 0 für verschiedene Batchgrößen

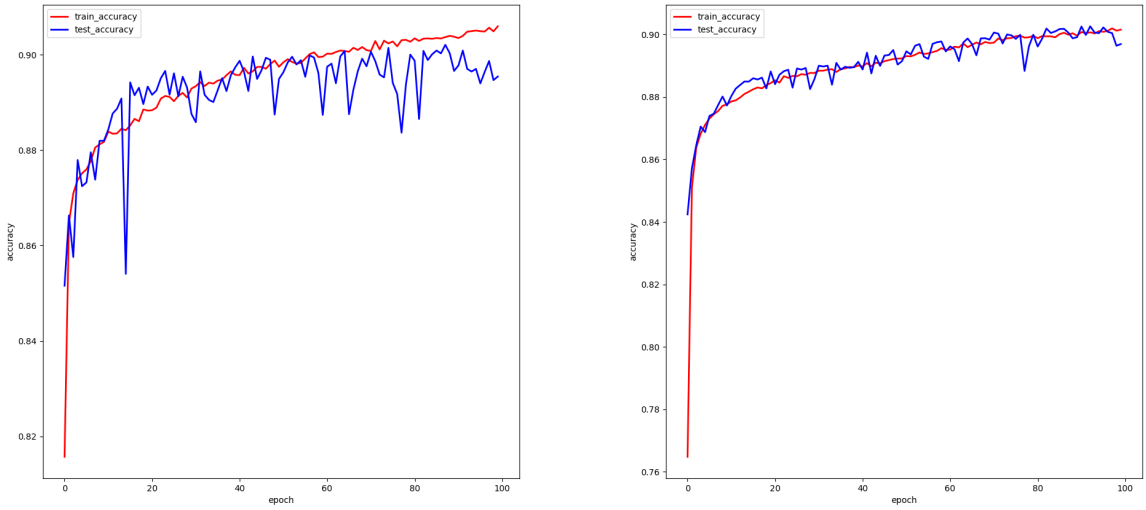
Mit der Batchgröße von 1000 sieht man eine deutlich schlechtere Genauigkeit und der Verlustwert ist ebenfalls größer. Auch die Ausgabeverteilung zeigt, dass die Überlappung der Verteilungen für Batchgröße 1000 deutlich größer ist als mit Batchgröße 100 (Siehe Abb. 6). Zwischen der Batchgröße 10 50 und 100 scheint es sowohl bei der Ausgabeverteilung als auch bei der Klassifikationsgenauigkeit kein großer Unterschied zu geben, allerdings kann eine zu kleine Batchgröße möglicherweise zu einer Überanpassung (overfitting) des Systems führen, weil die Gewichtsparameter eventuell zu oft anpasst werden. Deshalb wird die Batchgröße 100 als Optimalwert genommen.

leraning rate: Mit einer größeren *learning rate* kann zwar eine schnellere Optimierung erreicht werden, auf der anderen Seite kann es aber sein, dass der Anpassungsschritt zu groß wird und deshalb das optimale Minimum übersehen und nicht gefunden werden kann. Als *learning rate* wird zwischen 0.005 und 1 variiert und die ergebende Genauigkeiten verglichen, die in der Tabelle 6 dargestellt sind:

learning rate	loss	train acc	test acc	validierung acc
1	0,11228	0,9060	0,8954	0,8946
0,1	0,1146	0,9055	0,9014	0,8987
0,05	0,1154711	0,9015	0,8970	0,8977
0,01	0.161168	0,8607	0,8598	0,8605

Tabelle 6: MLP: Klassifikationsgenauigkeiten bei der Variation des Hyperparameters learning rate

Wie erwartet erhält man für kleine *learning rate* schlechtere Ergebnisse, da die Batchgröße und Epochenzahl unverändert bleiben. Das bedeutet, dass die Anzahl der Anpassungsversuche sich nicht verändert, während die Veränderung pro Anpassungsversuch kleiner wird. Gleichzeitig läuft man aber mit einer zu großen*learning rate* wie 1 dem Gefahr, dass das Netzwerk nicht das optimale Minimum findet, weil der Anpassungsschritt der Parameter zu groß ist. Dementsprechend sieht man in der Abb. 10, dass die Genauigkeit bei *learning rate* = 1 viel stärker schwankt als beim 0,05. Von 0,05 zu 0,1 sieht man wiederum eine nur eine leichte Verbesserung der Genauigkeiten, dafür dass der Wert stark erhöht wurde und somit das Risiko zur fehlhaften Optimierung gestiegen ist. So wird der Wert 0,05 als ein optimaler Wert für *learning rate* angenommen.



(a) learning rate 1

(b) learning rate 0,05

Abbildung 10: MLP: Klassifikationsgenauigkeiten der Trainings- und Testdaten für verschiedene learning rate

Aktivierungsfunktion: Es werden die Optimierungen jeweils für ReLU(Rectified Linear Unit)-, Sigmoid- und tanh- Funktion betrachtet und verglichen. Für den loss-Wert und Klassifikationsgenauigkeit ergaben folgendes in der Tabelle 7:

Die Verlust- und Genauigkeitswerte sowie die Ausgabeverteilungen zeigen, dass die ReLU-Funktion das

Aktivierungsfunktion	loss	train acc	test acc	validierung acc	Trainingsdauer
ReLU	0,1154711	0,9015	0,8970	0,8977	200s
Sigmoid	0,143188	0,8779	0,8777	0,8779	206s
Tanh	0,12557	0,8914	0,8900	0,8904	218s

Tabelle 7: MLP: Klassifikationsgenauigkeiten bei der Variation der Aktivierungsfunktion

beste Ergebnis liefert.
Außerdem wurde hier zusätzlich noch die Trainingsdauer gemessen. Ein großer Unterschied ist bei der Dauer nicht zu sehen, trotzdem wird mit der ReLU-Funktion die kürzeste Trainingszeit gebraucht. Insgesamt erzielt also die ReLU-Funktion die beste Effizienz.

Anzahl versteckter Ebenen: Für eine versteckte Ebene mit 500 versteckte Einheiten erhält man folgendes Ergebnis in der Tabelle 8:

loss	train acc	test acc	validierung acc	Trainingsdauer	verwendetes RAM
0,1245	0,8934	0,8935	0,8859	144	1,22 MB

Tabelle 8: Klassifikationsgenauigkeiten nach dem Trainieren des MLP mit einer versteckten Ebene mit 500 Einheiten

Nun wird eine weitere versteckte Ebene mit 300 Einheiten hinzugefügt (Tabelle 9):

loss	train acc	test acc	validierung acc	Trainingsdauer	verwendetes RAM
0,1155	0,9015	0,8970	0,8977	200s	1,26 MB

Tabelle 9: Klassifikationsgenauigkeiten nach dem Trainieren des MLP mit zwei versteckten Ebenen mit jeweils 500 und 300 Einheiten

Man sieht, dass eine weitere versteckte Ebene in dem MLP für die Optimierung nicht signifikant beiträgt. Allerdings wird bei Zunahme einer weiteren Ebene eine deutlich längere Trainingsdauer sowie größere RAM Verwendung erwartet und bei der Trainingsdauer ist auf jeden Fall ein signifikanter Unterschied zu beobachten. Da ein MLP aber allgemein nicht viel Speicher einnimmt und deshalb auch auf CPU tendenziell kleine Durchlaufzeit hat, sollen zwei versteckte Ebenen genutzt werden, um die maximale Genauigkeit zu erreichen.

Anzahl Trainingsdaten: In der Regel erhält ein Netzwerk mehr Informationen je mehr Trainingsdaten das Netzwerk bekommt. Allerdings kann es zu einem Übertraining (*overtraining*) des Netzwerks führen, wenn zu viele unreine Daten zum Trainieren verwendet werden. Für die Anzahl der Trainingsdaten sollten ebenfalls das Verhältnis zu der gesamten vorhandenen Daten berücksichtigt werden. Es werden für 8 verschiedene Quellen jeweils 30000 Kalibrationsdaten genommen und zufällig gemischt. Dann ergibt sich ein Kalibrationsdatensatz von 240000 Ereignissen. Nach einem Schnitt auf die Daten sind es dann nur noch etwas weniger als 220000 Ereignisse. Davon werden 70000 Ereignisse genommen und weitere 70000 Ereignisse aus einem Untergrunddatensatz, der aus ca. 110000 Untergrundereignisse besteht, sodass insgesamt 140000 Daten zum Training verwendet werden.

Nun wird überprüft, wie sich das Netzwerk für niedrigere Anzahl an Trainingsdaten verhält, die Genauigkeitsergebnisse sind in der Tabelle 10 zu finden.

Anzahl Trainingsdaten	loss	train acc	test acc	validierung acc
140000	0,1154711	0,9015	0.8970	0,8977
80000	0,1226	0,8987	0,8936	0,8914
20000	0,13775	0,8877	0,8849	0,8738

Tabelle 10: MLP: Klassifikationsgenauigkeiten bei der Variation der Anzahl der Trainingsdaten

Man erkennt zwar nicht viel aber dennoch klar eine Genauigkeitsminderung in allen Datensätzen sowie einen größeren Verlustwert. Ein Anzeichen von Übertraining durch Vergrößerung der Trainingsdaten ist dabei nicht zu sehen.

6.2 Optimierung des Convolutional Neural Network (CNN)

Das CNN wurde auf einem *GPU* trainiert, da das *CPU* irgendwann zu langsam für die Durchführung der massiven Mengen an Berechnungen eines CNNs wurde. Das Quellcode zum Training des *Convolutional Neural Network*s ist auf *Github Repository ANN*[1] verfügbar. Für das CNN sind die Hyperparameter und die Optimalwerte in der folgenden Tabelle 11 zu sehen:

Hyperparameter	als Standard verwendeter Wert/Funktion
Epochenanzahl	100
Batchgröße	100
learning rate	0,01
Aktivierungsfunktion	tanh-Funktion
Anzahl convolutional/pooling layer	3 Ebenen
Anzahl Trainingsdaten	32000
Kernelgröße (conv layer)	5x5
Padding (conv layer)	2
Art des pooling layers	Max-pooling
Filtergröße (pooling layer)	2x2
Stride (poolung layer)	2
Anzahl vollständig verbundener Ebenen	3

Tabelle 11: Auf der linken Seite der Tabelle sind die relevanten Hyperparameterarten des CNN, auf der rechten Seite sind die dazugehörige zum Trainieren verwendete Hyperparameter angeordnet.

Die Trainingsergebnisse mit den obigen Hyperparameter in Tabelle 11 sind in Tabelle 12 und Abb. 11 dargestellt:

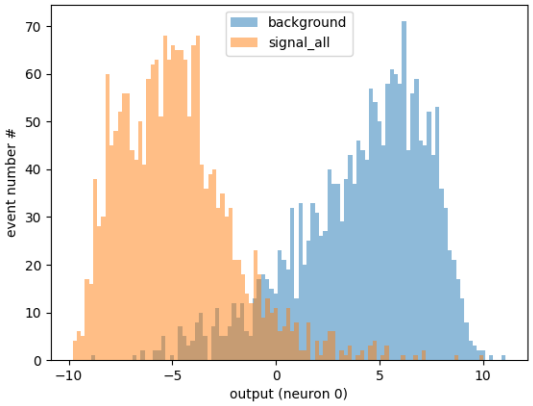
loss	train acc	test acc	validierung acc
0,119	0,956	0,920	0,9183

Tabelle 12: Die Klassifikationsgenauigkeiten des CNN mit den Standard Hyperparameterwerte

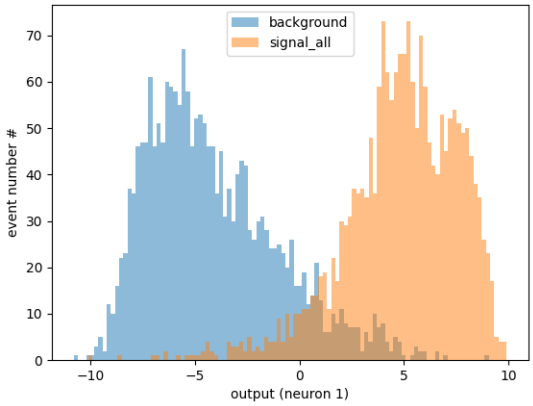
Epochenanzahl: Für die Bestimmung der optimalen Epochenanzahl kann wie beim MLP wieder der Genauigkeitsverlauf der Klassifikation über Epochenanzahl betrachtet werden.

Das CNN wurde dann für 1000 Epochen trainiert und der Genauigkeitsverlauf in der Abb. 12(b) zeigt sehr gut den Fall einer Überanpassung der Parameter, denn die Kurven der Testgenauigkeit fallen ab ca. 300

[1] https://github.com/dongjinsuh/ANN/blob/main/train_cnn.py



(a) Ausgabeneuron 0

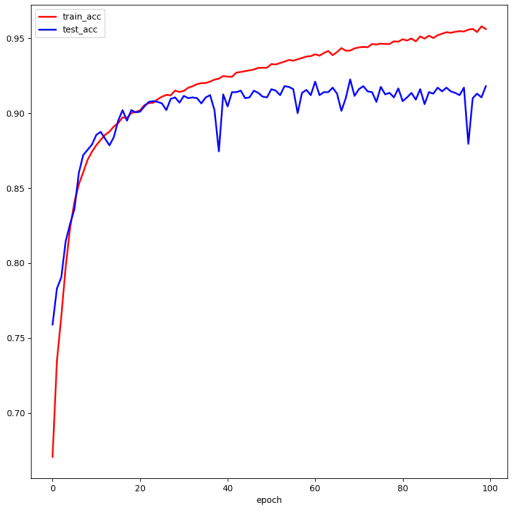


(b) Ausgabeneuron 1

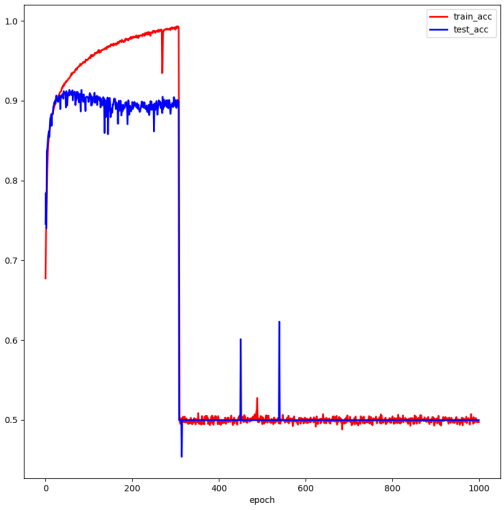
Abbildung 11: Signal- und Untergrundverteilungen der Ausgabeneuronen des CNN mit den optimalen Hyperparameter

Anzahl Epochen	loss	train acc	test acc	validierung acc
100	0,119	0,956	0,920	0,9183
200	0,05	0,983	0,913	0,9133

Tabelle 13: CNN: Variation der Anzahl der Epochenzahl



(a) Geanuigkeitsverlauf CNNs über 100 Epochen



(b) Geanuigkeitsverlauf über 1000 Epochen

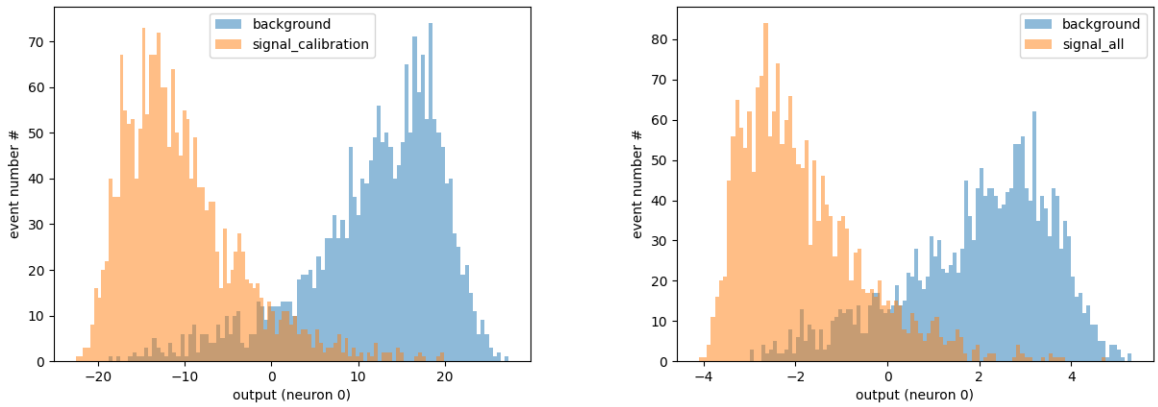
Abbildung 12: Trainings- und Testgenauigkeitsverlauf des CNNs mit unterschiedlicher Epochenzahl

Epochen drastisch ab auf eine Genauigkeit von 0,5.
Die Test- und Validierungsgenauigkeit scheinen für 100 Epochen am höchsten zu sein.

Batchgröße: Das CNN wird jeweils mit einer Batchgröße von 10, 100 und 1000 trainiert. Für die verschiedenen Batchgrößen ergeben sich folgende Klassifikationsgenuigkeiten in der Tabelle 14:

Batchgröße	loss	train acc	test acc	validierung acc
10	0,001	1,00	0,913	0,9128
100	0,119	0,956	0,920	0,9183
1000	0,299	0,879	0,887	0,8838

Tabelle 14: CNN: Klassifikationsgenauigkeiten bei der Variation der Batchgröße



(a) Ausgabeverteilung mit Batchgröße 10

(b) Ausgabeverteilung mit Batchgröße 1000

Abbildung 13: CNN: Signal- und Untergrundverteilung von Ausgabeneuron 0 für verschiedene Batchgrößen

Batchgröße scheint nicht sehr viel Einfluss auf die Klassifikation zu haben. Die Ausgabeverteilungen zwischen Batchgröße 10 und 1000 in Abb. 13 zeigen keinen signifikanten Unterschied, was die Überlappung angeht. Dennoch ist zu erkennen, dass die Batchgröße 100 die höchste Genauigkeit erweist, auch der Überschneidunganteil der Ausgabeverteilung (siehe Abb. 11 (a)) ist etwas geringer. Mit der Batchgröße 10 wird zwar der Verlustwert sowie die Trainingsgenauigkeit auf Maximum erreicht, allerdings zeigt es keine zusätzliche Verbesserung bei der Klassifikation der Test- und Validierungsdaten, außerdem zeigt ihre Ausgabeverteilung (Abb. 13 (a)) eine schlechte Trennung der beiden Verteilungen gegenüber der anderen.

learning rate: Das Ergebnis für Variieren der *learning rate* ist in der folgenden Tabelle Tabelle 15 zu sehen:

learning rate	loss	train acc	test acc	validierung acc
1	1,779	0,499	0,5	0,5
0,1	0,699	0,499	0,500	0,500
0,01	0,119	0,956	0,920	0,9183
0,005	0,186	0,930	0,892	0,8993

Tabelle 15: CNN: Klassifikationsgenauigkeiten bei der Variation der learning rate

Man erkennt, dass CNN für größere *learning rate* über 0,1 nicht gut trainiert werden kann. Man erkennt gut, dass die Genauigkeit während 100 Epochendurchläufe durchgehend den Bereich um 0,5 (50%) nicht verlässt.

Aktivierungsfunktion: Betrachtet werden jeweils die Tanh-, Sigmoid- und ReLU-Funktion als Aktivierungsfunktion und untersuchen die Effizienz des Netzwerks.

Aktivierungsfunktion	loss	train acc	test acc	validierung acc
Sigmoid (Batch 100)	1,144	0,504	0,499	0,500
Sigmoid (Batch 10)	0,208	0,919	0,910	0,9083
ReLU (Batch 10)	0,699	0,499	0,500	0,500
ReLU (batch 100)	0,130	0,949	0,901	0,8952
Tanh	0,119	0,956	0,920	0,9183

Tabelle 16: CNN: Klassifikationsgenauigkeiten bei der Variation der Aktivierungsfunktion

Einige Aktivierungsfunktionen scheinen stark von anderen Hyperparameter abhängig zu sein. Beispielsweise sieht man für Sigmoid-Funktion, dass das Training mit Batchgröße 100 nicht sinnvoll funktioniert, während man mit der Batchgröße 10 gute Ergebnisse bekommt. Für ReLU- oder Tanh-Funktion ist der Unterschied der Ergebnisse bei Variation von Batchgröße wiederum nicht groß und das CNN lässt sich immer gut trainieren. Es ist zu sehen, dass mit Tanh-funktion die beste Test- und Validierungsgenauigkeit erreicht wurden, wobei hier ebenfalls der Unterschied nicht all zu sehr groß ist. (Siehe Tabelle 16)

Anzahl Trainingsdaten: Für das CNN wurden die maximale Anzahl an Trainingsdaten verwendet, die der Speicherplatz erlaubt hat. Es werden 30000 Trainingsdaten mit jeweils 15000 Kalibrations- und 15000 Untergrundereignissen verwendet.

Convolutional/Pooling Layer: Da man für den Aufbau von Convolutional und Pooling Layer unendlich viele Möglichkeiten hat, und man auch nicht genau die möglichen vorhandenen Korrelationen abschätzen kann, ist es schwierig einen Anhaltspunkt für die Optimierung zu finden. Letztendlich wurden die folgenden Hyperparameter durch zahlreiche Ausprobieren und Vergleiche festgelegt (Siehe Tabelle 11). Deshalb sei es zu erwähnen, dass vor allem für diese Parametergrößen noch einiges an Potential für weitere Optimierungen bestehen kann.

7 Validierung der neuronalen Netzwerke

Ein sehr wichtiges Teil beim Aufbau eines Netzwerks ist die Validierung. Wenn ein Netzwerk trainiert wurde, muss zum Schluss geprüft werden, ob das Netzwerk auch tatsächlich die Aufgabe erfüllt, die von einem erwartet wird. Denn eine hohe Effizienz und Klassifikationsgenauigkeit eines Netzwerks muss noch nicht lange heißen, dass das Netzwerk auch für ganze andere und unbekannte Daten dieselben Ergebnisse liefert. Allerdings hat ein neuronales Netzwerk sehr komplexe interne Struktur und die Existenz von *hidden layers* (versteckte Ebenen) macht es einem sehr schwer hinterher zu folgen und nachzuvollziehen, wie das Netzwerk innerhalb der Ebenen abläuft und die Daten klassifiziert. Aus diesem Grund sollte ein Netzwerk validiert werden, um sicher zu gehen, dass durchgehend eine effiziente und korrekte Klassifikation garantiert wird.

In folgenden werden einige mögliche Validierungsprozesse von *Multilayer Perceptron* und *Convolutional Neural Network* vorgestellt und durchgeführt. Außerdem wird das Ergebnis anschließend mit der Likelihood-Methode verglichen. Alle Quellcodes zur Validierung der neuronalen Netzwerke sind auf dem *Github Repository ANN*[1] verfügbar.

7.1 Trainieren und Validieren der Netzwerke in Abhängigkeit der Energie

7.1.1 Validierung des Multilayer Perceptron (MLP)

Inputvariablen / geometrische Eigenschaften: Je nach dem, wie viele geometrische Eigensschaften man zur Verfügung hat und auch berücksichtigen möchte, ändert sich auch die Anzahl der Inputvariablen des ***M**ulti**L**ayer **P**erceptrons* (MLP). Denn der Wert einer geometrischen Eigenschaft entspricht einer Eingabevariable des Netzwerks. In der Regel kann ein MLP bei mehr Eingabevariablen möglicherweise eine bessere Klassifikation erreichen, da man mit größere Eingaben auch mehr Informationen über die geometrischen Eigenschaften eines Ereignisses bekommt, um diese bei der Klassifikation zu berücksichtigen. Die Frage, die man sich aber hier stellen muss, ist, wie aussagekräftig die einzelnen Eigenschaften sind, wenn es darum geht, Photonsignale und Untergrundsignale zu trennen. Was man sich dazu anschaut, ist zunächst der Unterschied einer geometrischen Eigenschaft zwischen Photonen und Untergrund. Dazu wird für jede Eigenschaft ein Histogramm jeweils von Kalibrationsdaten und Untergrunddaten und dies übereinander gelegt. Insgesamt werden 12 verschiedene Eigenschaften angeschaut und beispielsweise sind die Histrogramme von Silber der einzelnen geometrischen Eigenschaften in Anhang Abb. 31 bis 34 zu sehen.

Je kleiner die Überlappung zwischen den beiden Verteilungen, desto einfacher und besser kann das MLP die Ereignisse korrekt klassifizieren. Bei einer starken Überlappung der Signal und Untergrund würde die Wahrscheinlichkeit zur Falschklassifikation steigen, denn eine Überlappung der Verteilungen bedeutet, dass die entsprechende geometrische Eigenschaft zwischen den Photon- und Untergrundereignissen sich ähneln und somit für das MLP schwerer zu unterscheiden ist. Allerdings muss es nicht immer sinnvoll sein, eine geometrische Eigenschaft wegen einer starken Überlappung der Verteilungen auszuschließen. Denn eventuell kann ein kleines nicht überschneidendes Teil bei einem Ereignis der entscheidende Faktor sein, der ein photonartiges Signal von Untergrundereignissen trennt. Man sieht, dass die beiden Verteilungen je nach Eigenschaft zwar unterschiedlich stark überschneiden, aber dennoch ist bei allen Eigenschaften zu sehen, dass immer mindestens ein kleines Teil sich unterscheidet. Aus dem Grund wird keine geometrische Eigenschaft wegen

[1] https://github.com/dongjinsuh/ANN

ihrer Verteilungsähnlichkeit zwischen Signal und Untergrund ausgeschlossen.

Ein weiterer Aspekt, den man sich bei der Wahl der geometrischen Eigenschaften anschauen kann, ist die Verteilungsabweichung zwischen den einzelnen Kalibrationstargets bzw. Energien. Je nach Kalibrationstarget besitzen auch ihre Röntgenphotonen andere Energien, welche für manche Eigenschaften Auswirkungen haben kann. Um sich ihre Abhängigkeit von Energie anzuschauen, bildet man die Verteilung der signalartigen Ereignissen von allen Targets aufeinander ab, wodurch man die Abweichung der Verteilungen je nach Energie anschaulich darstellen kann. Diese sind in Abb. 14 und 15 und in Anhang Abb. 35 und 36 dargestellt.

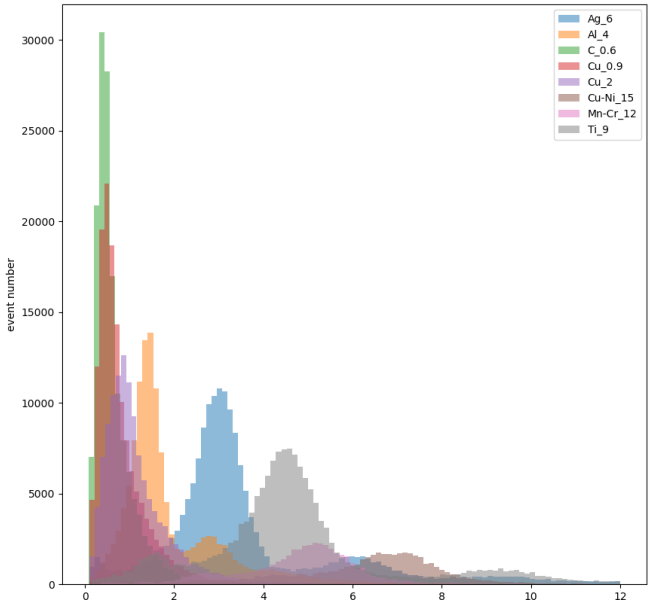


Abbildung 14: Die Verteilungen der Eigenschaft *Ladungsenergie (energyFromCharge)* von allen acht Kalibrationstargets sind gemeinsam dargestellt.

Man erkennt vor allem bei den Eigenschaften *Ladungsenergie* (*energyFromCharge*) und *Hits* eine signifikante Abweichung zwischen den einzelnen Verteilungen. Dieses Ergebnis ist allerdings zu erwarten, denn die Eigenschaft *Ladungsenergie* gibt die Energie der Gesamtladung und *Hits* die Anzahl der am Chip getroffenen Elektronen für ein Ereignis an. Das heißt nichts anderes als, dass die beiden Eigenschaften stark von der Energie des detektierten Photons abhängen muss. Nun kann es dazu führen, dass das Netzwerk sich einfach die Energien der Ereignisse merkt und später die Ereignisse nur anhand ihrer Energie klassifiziert, während die anderen Eigenschaften nicht mehr berücksichtigt werden. Deshalb werden die Eigenschaften *Ladungsenergie* und *Hits* nicht verwendet und die Anzahl der Eingaben des MLPs verkleinert sich auf 10. Die Histogramme für alle anderen geometrischen Eigenschaften sind in Anhang Abb. 35 und 36 zu finden. Anhand dieser Histogramme sieht man ganz gut, dass alle anderen Eigenschaften nicht viel bzw. kaum von dem Targetmaterial, also von der Energie abhängen.

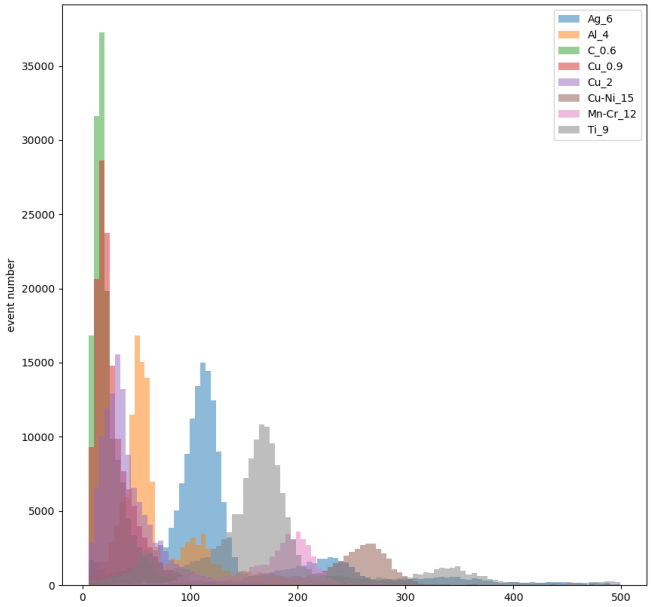


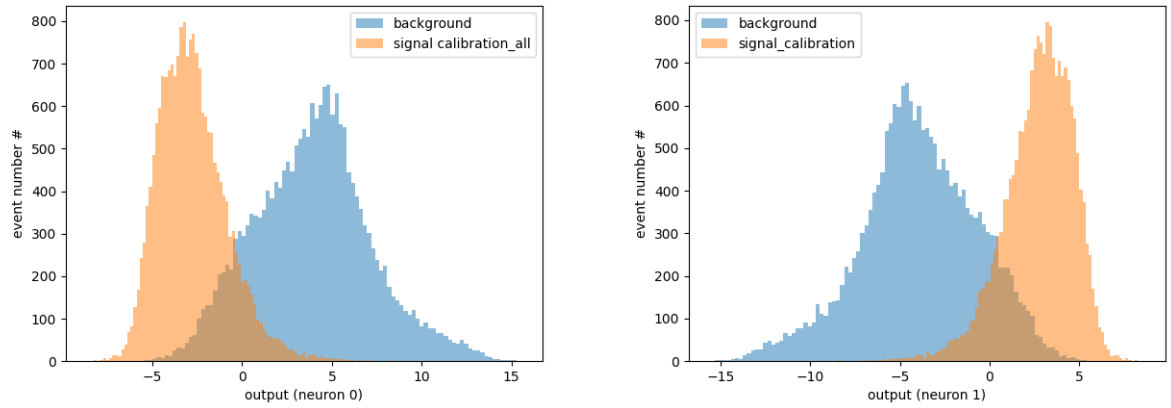
Abbildung 15: Die Verteilungen der Eigenschaft *Hits* von allen acht Kalibrationstargets sind gemeinsam dargestellt.

Allgemeine Validierung: Nun kommt man zu der tatsächlichen Validierung des MLPs, nachdem das Netzwerk mit den optimalen HyperparameterTabelle 2 trainiert wurde. Zunächst kann man ein Netzwerk mit allen vorhandenen Kalibrationsdaten trainieren. Dazu werden für jedes Kalibrationstarget 30000 Ereignisse genommen und diese zufällig gemischt. Da insgesamt Datensätze aus 8 verschiedenen Targets vorhanden sind, hat man einen Datensatz mit 240000 Ereignissen aus allen Energiebereichen. Am Ende sind es etwas weniger als 220000, nachdem die Daten durch den Schnitt gefiltert werden. Für das Training von MLP werden 70000 Ereignisse aus diesem Datensatz verwendet, dann weitere 10000 als Testdaten, und für die Validierung werden 10000 neue Daten genommen. Dann werden jeweils aus dem Untergrunddatendatz genauso viele Ereignisse wie die Kalibrationsdaten genommen. Für die Validierung hat man also insgesamt 20000 Ereignisse mit jeweils 10000 aus Kalibrationsdaten und 10000 Untergrunddaten. Die Klassifikationsgenauigkeiten für die erste Validierung mit allen gemischten Daten sind in der Tabelle 17 zu sehen.

loss	train acc	test acc	validierung acc
0,11398	0,9059	0,9006	0,898
			(Kalibration: 0,9182 / Untergrund: 0,8778)

Tabelle 17: Die Klassifikationsgenauigkeiten des MLPs für die Validierungsdaten, bestehend aus 20000 Ereignissen aus allen Kalibrationsdaten und Untergrunddaten

Um die Signaleffizienz eines Netzwerks genau zu betrachten, erstellt man die sogenannte ROC-Kurve (***R**eceiver **O**perating **C**haracteristic-curve*), welche allgemein eine Methode zur Bewertung und Optimierung von Analysestrategien ist. Die ROC-Kurve stellt visuell die Abhängigkeit der Effizienz gegen Untergrundunterdrückung dar. Aus der Ausgabeverteilung kann dann die ROC-Kurve erstellt werden, hierfür wurde die



(a) Neuron 0

(b) Neuron 1

Abbildung 16: Ausgabeverteilungen der Ausgabeneuronen des Multilayer Perceptron für die Validierungsdaten, bestehend aus 20000 Ereignissen aus allen Kalibrationsdaten und Untergrunddaten

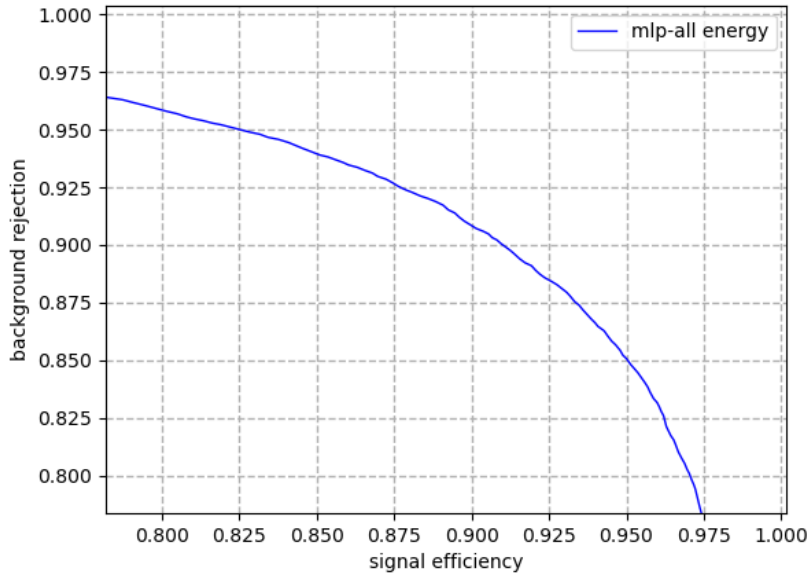


Abbildung 17: ROC-Kurve des Neuron 0 von der Validierung des MLP mit Validierungsdaten bestehend aus 20000 Ereignissen aus allen Kalibrationsdaten und Untergrunddaten

Ausgabeverteilung des Neuron 0 (Abb. 16 (a)) genommen. Die ROC-Kurve für die Validierung ist in Abb. 17 zu sehen.
Der optimalste Punkt wird bei einer Signaleffizienz von knapp unter 91% und ebenfalls bei 91% Untergrundunterdrückung erreicht, von diesem Punkt aus verhält sich die Signaleffizienz und Untergrundunterdrückung umgekehrt proportional zueinander.

Validierung im Bezug auf die Energieabhängigkeit: Nun lässt man das MLP nur mit Daten im höheren Energiebereich trainieren und schaut, wie gut das Netzwerk Ereignisse aus niedrigeren Energiebereichen klassifiziert. Dazu werden die Validierungsdaten ebenfalls in hochenergetische und niederenergetische Daten getrennt und separat validiert.
Insgesamt sind Kalibrationsdaten aus acht verschiedene Targets vorhanden. Die Energie der angeregten Linien in jedem Target ist aus der Tabelle 18 zu entnehmen:

Target	Filter	HV	Linie	Energie/keV
Cu	Ni	15	CuK_α	8,04
Mn	Cr	12	MnK_α	5,89
Ti	Ti	9	TiK_α	4,51
Ag	Ag	6	AgL_α	2,98
Al	Al	4	AlK_α	1,49
Cu	EPIC	2	CuL_α	0,930
Cu	EPIC	0,9	OK_α	0,525
C	EPIC	0,6	CK_α	0,277

Tabelle 18: Tabelle aller Ziel-Filter-Kombinationen und der entsprechenden Fluoreszenzlinien sowie ihre Energien[22][23]

Um Daten aus hohen Energiebereichen zu generieren, wurden nur aus den Kalibrationsdaten von Silver (Ag), Kupfer-Nickel (Cu-Ni), Moscovium-Chrom(Mn-Cr) und Titan (Ti) die Ereignissen entnommen. Für den niederenergetischen Datensatz wurden die Kalibrationsdaten Aluminium (Al), Kohlenstoff (C)und Kupfer (Cu) mit jeweils 0,9 kV und 2 kV Hochspannung verwendet.
Dann wird nur mit den Ereignissen aus den niedrigen Energiebereichen trainiert und zum Schluss wird wieder mit Datensatz aus allen Kalibrationsdaten trainiert. Es werden jeweils die hochenergetische Ereignisse und niederenergetische Ereignisse validiert und beobachten das Verhalten des MLPs für die einzelnen Fälle. Die Validierungsergebnisse sind in der Tabelle 19 sowie in Abb. 18 und 20 und **??** zu sehen.

Kalibrationstarget Trainingsdaten	loss	train acc	test acc	valid acc (alle)	valid acc (niederenerg.)	valid acc (hochenerg.)	valid acc (untergrund)
Alle	0,114	0,906	0,9006	0,8918	0,9160	0,9746	0,8778
Al, C, Cu (09), Cu(2)	0,127	0,895	0,8925	0,849	0,9548	0,7634	0,8530
Ag, Cu-Ni, Mn-Cr, Ti	0,0629	0,9555	0,9531	0,8609	0,3985	0,9686	0,9552

Tabelle 19: MLP: Die Klassifikationsgenauigkeiten der Validierung jeweils für Daten aus vier Kalibrationstargets der oberen Energiebereichen und vier Targets der unteren Energie

Für das Training mit den vier Kalibrationsdaten aus oberen Energiebereichen liefert das MLP eine absolut hohe Genauigkeit für hochenergetische Validierungsdaten gegenüber der niederenergetische Validierung. Umgekehrt sieht man auch dasselbe Verhalten für das Training mit Ereignissen der niederenergetischen Kalibrationstargets. Anhand der ROC-Kurven in Abb. 19 und 20 sieht man auch ganz gut, dass die Effizienz für die Validierungsdaten viel besser ist, wenn sie im selben Energiebereich wie die Trainingsdaten liegen.

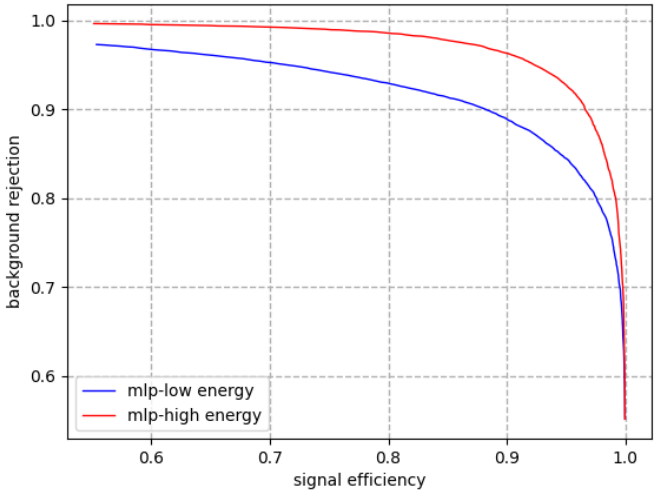


Abbildung 18: MLP: ROC-Kurve der Validierung mit Training aus allen Energiebereichen unter Verwendung aller Kalibrationsdaten

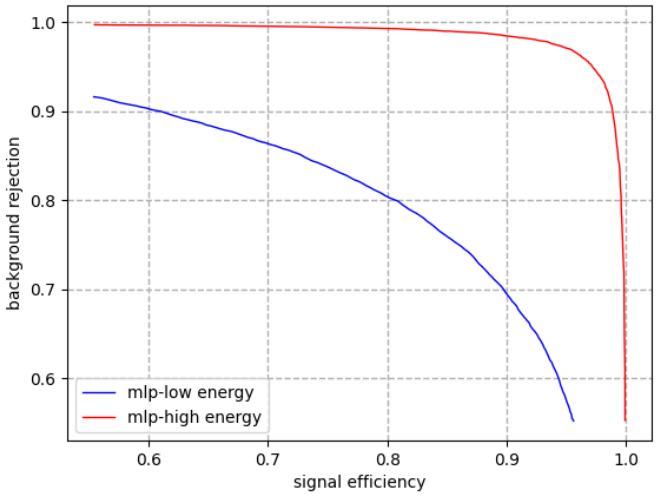


Abbildung 19: MLP: ROC-Kurve der Validierung mit hochenergetischen Kalibrationsdaten (Ag, Cu-Ni, Mn-Cr, Ti)

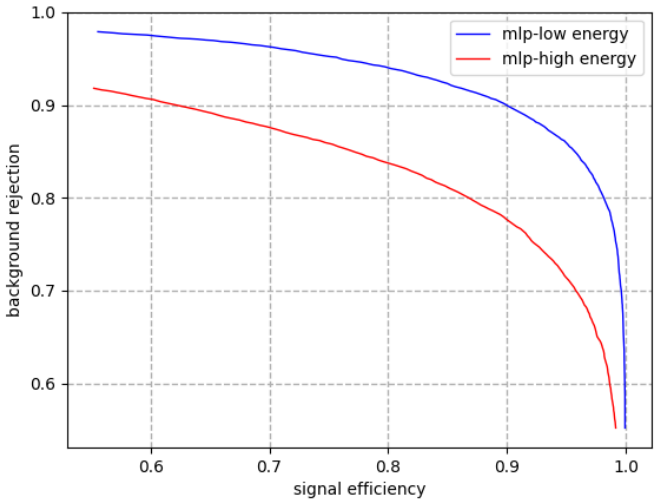


Abbildung 20: MLP: ROC-Kurve der Validierung mit nirderenergetischen Kalibrationsdaten (Al, C, Cu (09), Cu(2))

Signifikant auffallend ist auch die schlechte Validierungsergebnisse für die Daten der niederenergetischen Kalibrationstargets. Anhand der allen drei ROC-Kurven (Abb. 18 bis 20) sowie der Klassifikationsgenauigkeit (Tabelle 19) kann man erkennen, dass die Effizienz des MLPs für den niederen Energiebereich deutlich abfällt als bei den Ereignissen im höheren Energiebereich. Zusammen über alle drei Trainingsdaten erreicht das MLP für höhere Energie deutlich bessere Effizienz und man sieht auch beim Training mit allen Daten, dass die Effizienz für hochenergetische Daten höher ist. Vor allem, wenn das MLP nur mit den Hochenergiedaten trainiert wird, wird dieser Effekt viel größer. Die Validierungsgenauigkeit von Niederenergiedaten fällt auf knapp 40% und auch der Abstand der beiden ROC-Kurven ist am größten bei allen drei Trainingsfällen.

Zusätzlich soll auf die reine Energieabhängigkeit des MLPs bzw. die Abhängigkeit zwischen Energie und Netzwerkeffizienz untersucht werden. Wenn allerdings für die Validierungsdaten der aus allen Targets zufällig gemischten Datensatz genommen wird, ergibt sich ein folgendes Problem. Dazu betrachtet man wieder die Abb. 14 an, wo das Histogramm der Eigenschaft *energyFromCharge* von jedem Kalibrationstarget übereinander gelegt wurden. Die Energie eines Kalibrationstarget folgt einer Wahrscheinlichkeitsverteilung und man erkennt den energetischen Unterschied von allen acht Targets durch die Abweichung der jeden einzelnen Verteilungen. Wenn man die Ereignisse aller Kalibrationstargets gemeinsam als ein Histogramm darstellt, ergibt sich dann die obige Abb. 21. Dazu wurde der Energiebereich von 0 bis 8 keV in Abschnitten von 0,25 keV Intervall aufgeteilt und für jeden Abschnitt wurde eine obere Begrenzung gesetzt, dass in jedem Energieabschnitt maximal 30000 Daten durchgelassen werden kann, das heißt, es werden so gut wie kein Ereignis ausgefiltert.

Man erhält aufgrund verschiedender Röntgenphotonenergie zwischen den einzelnen Kalibrationstargets eine Verteilung aus mehreren nebeneinander angereihten Verteilungen der Quellen, wodurch mehrere Maxima und Minima entstehen. Wenn man aber aus dieser Verteilungsdaten das Netzwerk validiert, bekommt man je nach Energiebereiche unterschiedliche Statistik. Man sieht an dem Histogramm in Abb. 21, dass man in manchen Bereichen eine sehr niedrige Statistik hat, während für einige Energien sehr hohe Statistik vorliegt. Das bedeutet, dass man keine klare Aussage treffen kann, wenn später die Effizienz gegen die Energie

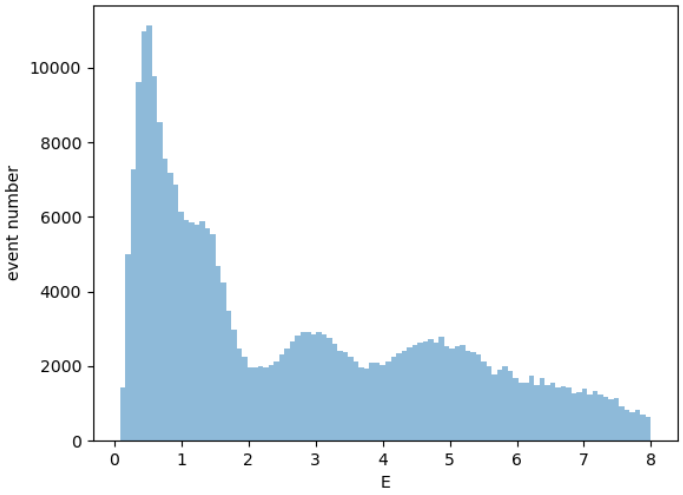


Abbildung 21: Verteilung der Validierungsdaten gegen die Energie, in jedem Energieabschnit von 0,25 keV wird eine Begrenzung von 30000 Ereignissen gesetzt.

aufgetragen wird, da bei einer hohen Statistik eine genaue Bestimmung der Effizienz erwartet wird, während dessen bei einer zu niedrigen Anzahl an Validierungsdaten, die Wahrscheinlichkeit groß ist, dass die Effizienz für die Energiebereiche mit niedrigen Statistik verfälscht dargestellt werden kann.
Um dieses Problem zu vermeiden, werden die Validierungsdaten so gewählt, dass die Anzahl der Ereignisse in allen Energiebereichen gleichmäßig verteilt ist und man somit überall einen ziemlich ähnlichen statistischen Einfluss auf die Effizienz des Netzwerks erwarten kann. Dazu wird für jeden Energiebereich mit einem 0,25 keV Intervall die maximale Datenanzahl von ursprünglich 30000 Ereignissen soweit reduziert bis man über den gesamten relevanten Energiebereich eine konstante Ereignisanzahl hat. In der Abb. 22 ist nun das Histogramm der gefilterten Validierungsdaten mit einer oberen Begrenzung von 2000 Ereignissen zu sehen und man erkennt, dass die Ereignisanzahl in allen Energiebereichen gleichmäßig verteilt ist.

Mit diesen gefilterten Daten wird das MLP validiert und für jeden Energieintervall wird die Genauigkeit berechnet und als Energie-Genauigkeit-Diagramm (Abb. 23) dargestellt:

Man erkennt hier vor allem in sehr niedrigen Energiebereichen eine deutliche schlechtere Klassifikationsgenauigkeit (Accuracy). Ab 3 keV bleibt die Genauigkeit dann durchgehend konstant über 90%. Bereits oben hat man versucht, das Verhalten des MLP zu validieren, indem man die Daten in zwei große Energiebereiche aufgeteilt hat. Es lassen sich wieder ein ähnliches Verhalten feststellen, dass die Ereignisse mit kleineren Energie häufiger falsch klassifizieren lässt. Man erkennt ab 3 keV einen drastischen Abfall der Genauigkeit. Vorhin wurden alle Ereignisse unter ca. 2-3 keV als niederenergetisch gruppiert und in diesem Energie-Genauigkeit-Diagramm lässt sich dieses Phänomen bestätigen. Im niedrigeren Energiebereich lassen sich dann vor allem unter 1 keV eine schlechte Ergebnisse zeigen, wo die Genauigkeit durchgehend unter 90% fällt. Ab 0,5 keV sinkt die Genauigkeit sogar auf 76% und zum Schluss unter 70%. Dazu schaut man sich folgendes an. Es wird beispielsweise das Histogramm der Eigenschaft *kurtosisLongitudinal* in der Abb. 36(d) angeschaut, wo die Verteilungen aller Kalibrationstargets zu sehen sind. Man sieht im wesentlichen eine recht breite Überlappung der Verteilungen, allerdings zeigen C und Cu (0,9kV) eine etwas deutliche Abweichung gegenüber der anderen Quellen. Die angeregten Linien der Targets C und Cu (0,9kV) haben Energie von 0,525 keV und 0,277 keV.

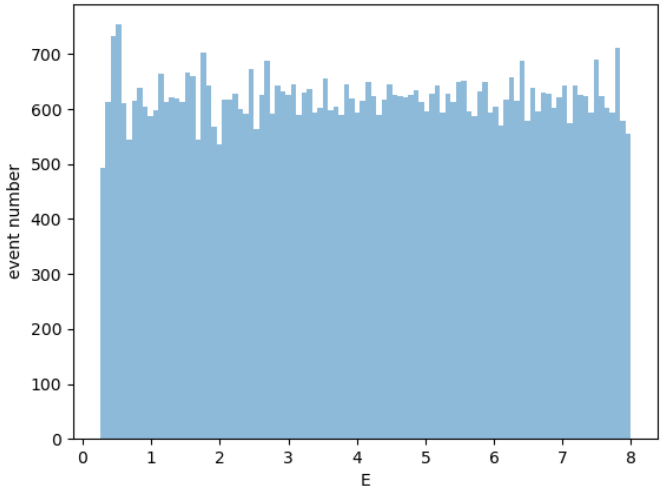


Abbildung 22: Verteilung der Validierungsdaten gegen die Energie, in jedem Energieabschnitt von 0,25 keV werden maximal 2000 Ereignisse aufgenommen.

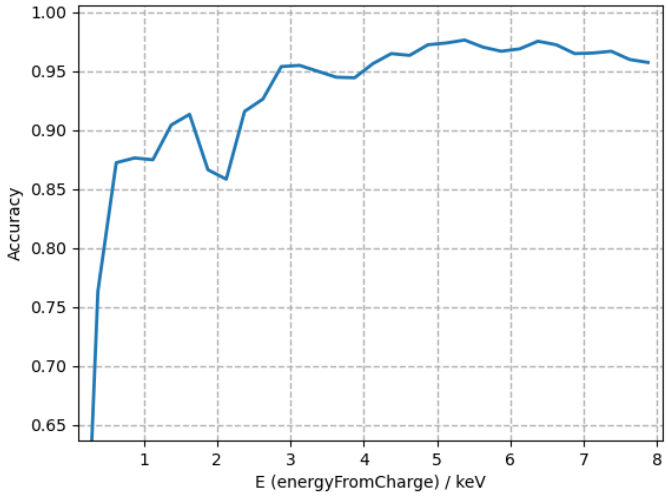


Abbildung 23: Darstellung der Klassifikationsgenauigkeit des MLPs gegen die Energie aus den Validierungsdaten, in den die Energieverteilung gleichmäßig hoch gefiltert wurden.

Da die Eigenschaft *kurtosisLongtudinal* als eine Eingabevariable im MLP berücksichtigt wird, kann diese Abweichung nicht wenige Auswirkung auf die Klassifikation haben, weshalb man eine deutlich schlechtere Klassifikationsergebnisse in den Energiebereich von C und Cu (0,9kV) erwarten kann.

Insgesamt lässt sich schlussfolgern, dass das MLP die Ereignisse nicht ganz unabhängig von der Energie klassifizieren kann, auch wenn die stark von der Energie abhängigen Eigenschaften verworfen wurden. Zum einen ist es wahrscheinlich, dass auch andere geometrische Eigenschaften nicht komplett von der Energie unabhängig sind, was man auch an den Histogrammen der manchen Eigenschaften leicht erkennen kann. Außerdem ist eine zusätzliche Vermutung, dass je nach Kalibrationstarget, abgesehen von der unterschiedliche Energie, andere materialabhängige Eigenschaften für die erzeugten Röntgenstrahlungen existieren, die dann Einfluss auf die geometrische Größen der detektierten Pixeldaten haben können. Allerdings wird in dieser Arbeit nicht darauf eingegangen. Außerdem lässt sich konstant zeigen, dass die Klassifikation von niederenergetischen Ereignisse wesentlich schlechter ausfällt als von hochenergetischen Ereignissen.

7.1.2 Validierung des Convolutional Neural Network (CNN)

Im voraus sei gesagt, dass die Validierung für das CNN etwas eingeschränkter ist als das MLP, weil man viel mehr Speicher und höhere Rechenleistung des CPU/GPU braucht. Denn ein CNN besitzt üblicherweise mehr Ebenen und auch innerhalb der einzelnen versteckten Ebenen sind viel mehr Neuronen, was heißt, die Anzahl der Parameter bzw. Neuronen eines CNNs, die jedes mal verändert und berücksichtigt werden muss, ist um einige Größenordnungen größer und erfordert dementsprechend eine höhere Rechenleistung. Außerdem besteht ein Ereignis beim CNN aus einem Pixeldaten aus 256x256 Elektronladungen, also muss es 65536 Eingaben für ein Ereignis geben, während MLP als Eingabe nur die 10 geometrische Eigenschaften nimmt. Diese führt dazu, dass man einen sehr großen Datenspeicher braucht, damit ein CNN diese Daten aufnimmt und verarbeiten kann. Leider sind die einem zur Verfügung stehende Ausstattungen beschränkt, weshalb verschiedene und umfangreiche Validierungen mit vielen Daten etwas schwerer umzusetzen sind.

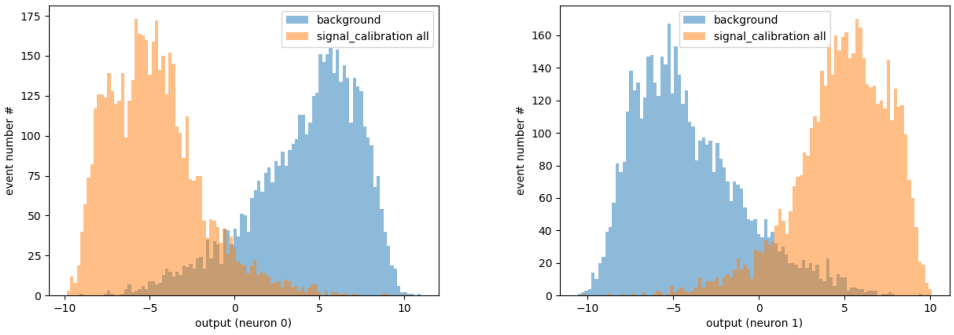
Allgemeine Validierung: Trotz allem kann man zunächst das CNN wie beim MLP mit allen Kalibrationsdaten verschiedener Energien trainieren und das Validierungsergebnis betrachten. Es werden möglichst viele Daten verwendet und dem CNN gegeben, wie der Speicher es zulässt. Insgesamt können 10000 Validierungsdaten mit jeweils 5000 Kalibrations- und 5000 Untergrundereignisse verwendet werden. Die verwendeten Hyperparameter zum Trainieren des CNNs sind in Tabelle 11 zu finden.

Die Ergenisse der Klassfikationsgenauigkeiten sind in Tabelle 20, die Verteilungen der zwei Ausgabeneuronen sowie die entsprechende ROC-Kurve dazu sind in Abb. 24 und 25 zu sehen.

loss	train acc	test acc	validierung acc
0,119	0,957	0,920	0,9138
			(kalibration: 0,9414 / Untergrund: 0,8862)

Tabelle 20: Die Klassifikationsgenauigkeiten des CNN für die Validierungsdaten, bestehend aus 5000 Ereignissen aus allen Kalibrationstarget und 5000 Untergrundereignissen

Der optimalste Punkt für die Signaleffizienz und Untergrundunterdrückung liegt knapp über 93% für die allgemeine Validierung des CNN.



(a) Neuron 0 (b) Neuron 1

Abbildung 24: Ausgabeverteilungen der Ausgabeneuronen des CNN für die Validierungsdaten, bestehend aus 5000 Ereignissen aus allen Kalibrationstarget und 5000 Untergrundereignissen

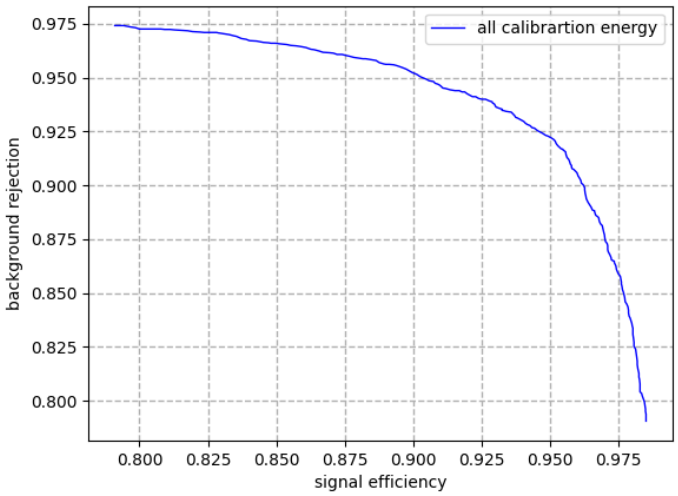


Abbildung 25: CNN: ROC-Kurve des Neuron 0 für die Validierungsdaten aus 5000 Ereignissen aus allen Kalibrationstarget und 5000 Untergrundereignissen

Validierung im Bezug auf die Energieabhängigkeit: Wie beim MLP lässt man ebenfalls das CNN nur mit Daten im höheren bzw. niedrigeren Energiebereich trainieren und schaut, wie gut das Netzwerk Ereignisse aus sowohl niedrigeren als auch höheren Energiebereichen klassifiziert. Dann soll das Netzwerk wieder mit Daten trainiert werden, die die Ereignisse von allen Kalibrationen enthalten. Es werden jeweils mit Validierungsdaten aus allen Energiebereichen, dann in niedrigen Energiebereich und in höheren Energiebereichen validiert. Ebenfalls validiert werden die Untergrundereignisse.

Der untere Energiebereich geht von 0 keV bis ca. 2 keV, dazu wurden Ereignisse aus Kalibrationsdaten Aluminium (Al), Kohlenstoff (C) und Kupfer (Cu)(0,9 kV) und (Cu)(2 kV) genommen, der Hochenergiebereich besteht aus Kalibrationsdaten von Silver (Ag) mit Verteilungsmaximum bei 2,98 keV, Kupfer-Nickel (Cu-Ni) bei 8,04 keV, Moscovium(Mn-Cr) mit 5,89 keV und Titan (Ti) mit 4,51 keV, die Energie der einzelnen Ereignisse reichen also von ca. 3 keV bis 8 keV (siehe Tabelle 18). Ebenfalls anhand der Energieverteilungen der Kalibrationsdaten Abb. 14 kann man einschätzen, in welchem Energiebereich die jeweiligen Kalibrationsdaten liegen. Allerdings sind die beiden niederen und oberen Energiebereiche nicht exakt bei 2 keV begrenzt und getrennt, da alle Ereignisse aus einem Kalibrationstarget nicht genau die selbe Energie besitzt, sondern einer gewissen Verteilung folgt und die Verteilungen zum Teil sich überschneiden. Also kann durchaus ein Ereignis von Aluminium (Al) eine höhere Energie haben als die eines Silbers (Ag), wobei solche einzelne Vermischung bei einer Datensatzumfang von mehreren Tausenden Ereignissen wenig Einfluss auf die Validierungsergebnisse haben werden.

Die Validierungsergebnisse sind in **??** zu finden, die Untergrundunterdrückungen und Signaleffizienz sind in Abb. 26 bis 28 dargestellt.

Kalibrationstarget Trainingsdaten	loss	train acc	test acc	valid acc (alle)	valid acc (niederenerg.)	valid acc (hochenerg.)	valid acc (untergrund)
Alle	0,120	0,957	0,921	0,9132	0,9606	0,9712	0,8822
Al, C, Cu (09), Cu(2)	0,091	0,967	0,914	0,9133	0,9468	0,9134	0,8928
Ag, Cu-Ni, Mn-Cr, Ti	0,092	0,967	0,947	0,9464	0,7662	0,9316	0,9552

Tabelle 21: CNN: Die Klassifikationsgenauigkeiten der Validierung jeweils für die Daten aus vier Kalibrationstargets der oberen Energiebereiche und aus vier Targets der unteren Energiebereiche

Man sieht, dass der Unterschied zwischen Hochenergie- und Niederenergiedaten für alle drei Trainingsfälle relativ gering ist. Ansonsten fällt auf, während für den Trainingsfall mit niederenergetischen Daten der Unterschied eher beim Signaleffizienz zu sehen ist, unterscheidet sich für die anderen Fällen die Untergrundunterdrückung stärker. Eine größere Abweichung ist dennoch weder bei Untergrundunterdrückung als auch bei Signaleffizienz für alle Fälle nicht zu erkennen. Auch wenn man beim CNN die Validierung zur Überprüfung der Energieabhängigkeit nur begrenzt grob in hoch- und niederenergetischen Bereichen durchführen konnte, kann dennoch definitiv festgestellt werden, dass die Energieabhängigkeit des CNN bei der Klassifikation gegenüber MLP deutlich geringer ist.

7.2 Vergleich der Signaleffizienz mit der Likelihood-Methode

In diesem Abschnitt möchte man sowohl die MLP-, als auch die CNN-Methode mit der sogenannten Likelihood-Methode vergleichen, welche vergleichsweise eine einfache Methode ist. Auf Basis von Kalibrationsdaten einer Röntgenröhre kann man für verschiedene Energien die geometrischen Eigenschaften dieser Röntgenphotonen in dem Detektor charakterisieren. Indem man jede Verteilung als eine Wahrscheinlichkeitsdichte interpretiert,

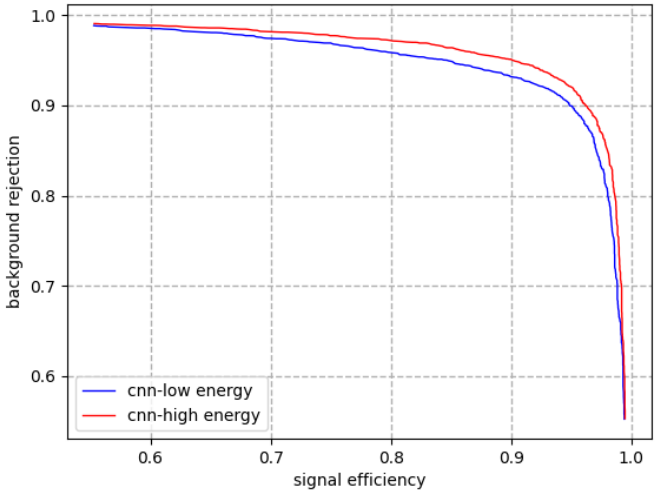


Abbildung 26: ROC-Kurve der Validierung mit Training aus allen Energiebereichen unter Verwendung aller Kalibrationsdaten für CNN

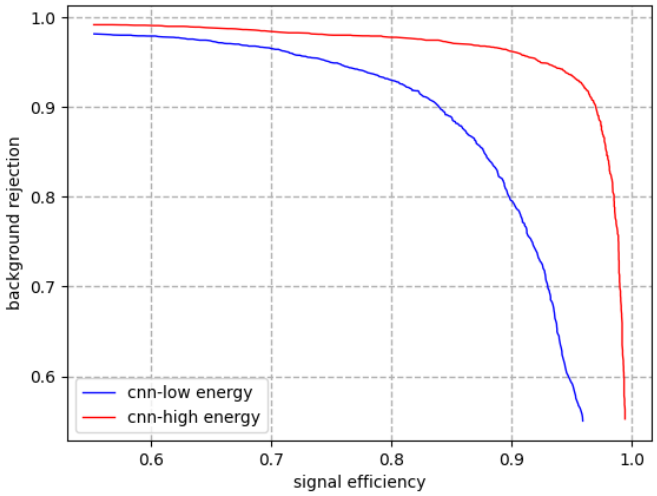


Abbildung 27: CNN: ROC-Kurve der Validierung mit hochenergetischen Kalibrationsdaten (Ag, Cu-Ni, Mn-Cr, Ti)

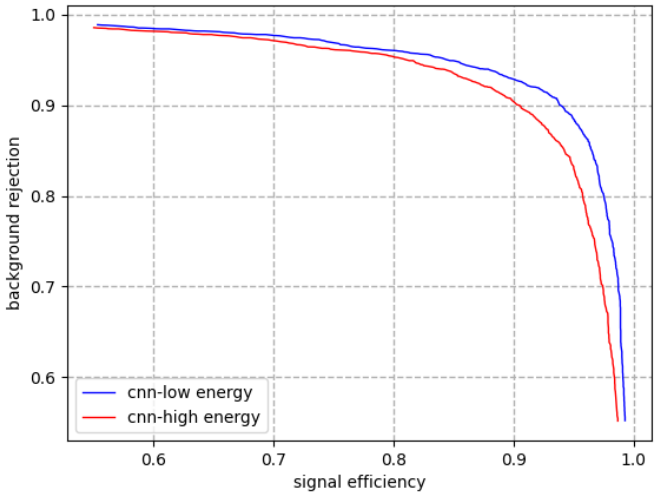


Abbildung 28: CNN: ROC-Kurve der Validierung mit niederenergetischen Kalibrationsdaten (Al, C, Cu (09), Cu(2))

kann eine Likelihood Verteilung der Photonen und des Untergrundes berechnet werden, wobei Likelihood definiert ist als das Produkt der Wahrscheinlichkeitsverteilungen. Anhand dieser eindimensionalen Verteilung definieren wir einfach einen Schnitt auf den Likelihood Wert (auf die X Achse des 'Histogramms'), sodass man eine gewisse Signaleffizienz erreicht, oder anders gesagt, dass man einen gewissen Teil aller Photonen behält und gewillt ist, den Rest zu verwerfen. Die Einfachheit der Methode lässt aber zu, dass man einiges an Potential verschwenden, das man nutzen kann, um die zwei Arten Ereignissen besser zu trennen.

Für die beiden Methoden der neuronalen Netzwerke werden also jeweils mit der Likelihood-Methode verglichen und geschaut, in wieweit sich die Anwendung von neuronalen Netzwerken gegenüber Likelihood-Methode im Hinblick auf die Trennung der Ereignissen ausmacht.

In der Abb. 29 sind die ROC-Kurven der Likelihood-Verteilungen und MLP-Ausgabeverteilungen für alle Kalibrationstargets gemeinsam dargestellt. Die Programme zur Rekonstruktion der verwendeten Likelihood-Werte sind in dem Repository *TimepixAnalysis*[1] zu finden. Zur besseren Veranschaulichung sind die Kurven außerdem getrennt für jedes Target in Anhang Abb. 37 und 38 zu finden.

Für die CNN sind die gemeinsame Darstellung der ROC-Kurven in Abb. 30, sowie die einzelnen Kurven ebenfalls in Anhang Abb. 39 und 40 zu sehen.

Es lässt sich anhand der Abb. 29 und 30: eindeutig sagen, dass sowohl das MLP als auch CNN eine deutlich bessere Signaleffizienz in Abhängigkeit von Untergrundunterdrückung erreichen als die Likelihood-Methode. Es fällt auf, dass die Likelihood-Kurven in den Abb. 29 und 30 nicht indentisch sind, was darin begründet wird, dass beim Vergleich mit MLP-Ergebnissen die Likelihood-Werte zusätzlich anhand der anderen geometrischen Eigenschaften gefiltert wurden, während beim CNN die rohen Likelihood-Daten ohne Schnitt auf die geometrischen Verteilungen verwendet wurden. Für die Schnittwerte wurden die selben Werte verwendet, wie die Trainingsdaten der Netzwerke geschnitten wurde. Auch die Likelihood-Methode scheint eine Effizienzsteigerung durch den Schnitt der Daten zu erreichen. Dennoch ist klar zu beobachten, dass

[1] https://github.com/Vindaar/TimepixAnalysis

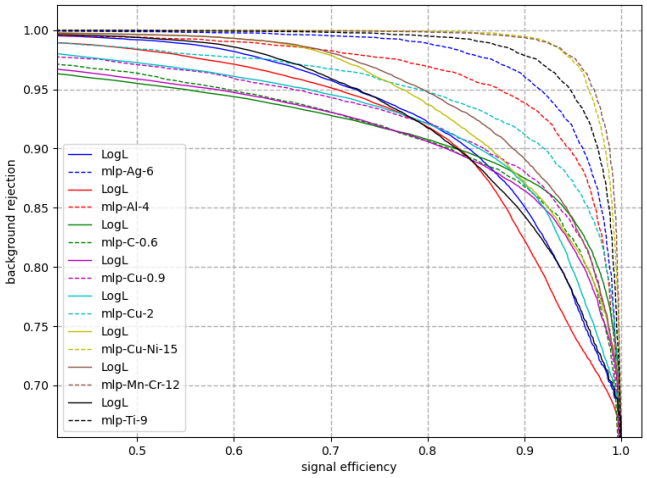


Abbildung 29: ROC-Kurve der Likelihood- und MLP-Ergebnisse von allen Kalibrationstargets

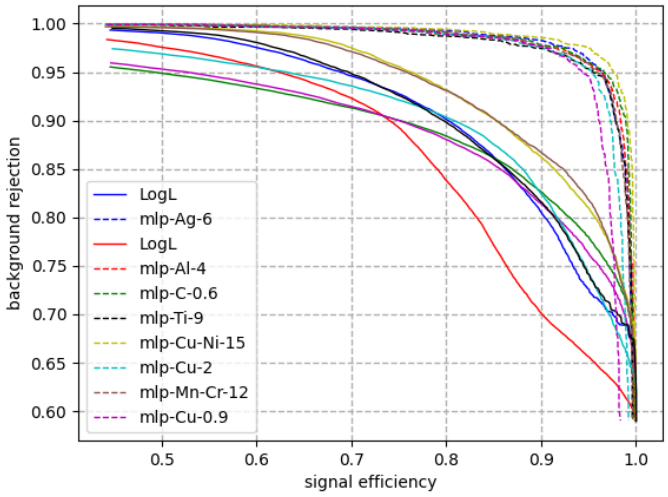


Abbildung 30: ROC-Kurve der Likelihood- und CNN-Ergebnisse von allen Kalibrationstargets

die beiden Netzwerke gegenüber der sowohl geschnittenen als auch rohen Likelihood-Werte tendenziell eine signifikante Erhöhung der Signaleffizienz erzielen.

7.3 Gegenüberstellung der Effizienz des Multilayer Perceptron und Convolutional Neural Network

Zum Abschluss sollen alle Ergebnisse der beiden Netzwerke gegenübergestellt und verglichen werden im Hinblick auf die Effizienz der Signal- und Untergrund-Klassifikation.
Dazu werden zuerst die Validierungsergebinsse für alle Kalibrationsdaten angeschaut. Anhand der Abb. 17 und 25 erkennt man eine bessere Signaleffizienz für das CNN. Das CNN erreicht eine maximale Signaleffizienz und Untergrundunterdrückung von über 93% (siehe Abb. 25), während das MLP das Optimum bei ca. 90,5% Signaleffizienz bzw. Untergrundunterdrückung liegt. Bei einer Untergrunduntergrückung von 93% liegt nur eine Signaleffizienz von etwa 86-87% vor (siehe Abb. 17).
Außerdem können noch die Validierungsergebnisse in Abhängigkeit der verschiedenen Energie verglichen werden. Die Abb. 18 bis 20 für MLP sowie Abb. 26 bis 28 zeigen, dass das CNN eine ähnliche Signaleffizienz für niederenergetische und hochenergetische Ereignisse erreicht, während beim MLP ein deutlicher Effizienzunterschied zwischen beiden Energiebereichen zu erkennen ist. Anhand der Abb. 30 von CNN lässt sich auch schön sehen, dass die Effizienz zwischen den einzelnen Kalibrationstarget sich nicht stark voneinander abweichen. Das MLP zeigt dagegen ein deutlich schlechteres Ergebnis. Es ist ein deutlichere Abweichung der Signaleffizienz zwischen den einzelnen Target erkennen. Da jedes Target in unterschiedlichen Energiebereichen liegen, lässt sich daraus schlussfolgern, dass die Energieabhängigkeit beim CNN viel geringer ist als beim MLP.
Insgesamt lässt sich zusammenfassen, dass CNN verglichen mit dem MLP in allen Bereichen der Validierung eine höhere Leistung liefert. Zwar zeigen die Signaleffizienzen gegenüber der Untergrundunterdrückung sowie die Klassifikationsgenauigkeiten keinen sehr großen Unterschied zwischen den beiden Netzwerken, trotzdem sind kontinuierlich bessere Ergebnisse für das CNN zu sehen. Auch im Bezug auf die Energieabhängigkeit sieht man, dass CNN weniger abhängig von der Energie die Ereignisse klassifizieren kann, bzw. über allen Energiebereichen durchgehend konstante Ergebnisse liefert, während das MLP eine deutliche energieabhängige Leistungsschwankung zeigt.

Die Überlegenheit eines CNNs gegenüber MLP sollte allerdings nicht sehr überraschend sein. Denn ein CNN erhält die gesamten rohen Pixeldaten aus dem Detektor als Eingabe, das bedeutet, es können viel mehr Aspekte und Eigenschaften zur Unterscheidung der Signal- und Untergrunddaten berücksichtigt werden als die einem bekannt ist, während das MLP nur anhand der verwendeten geometrischen Eigenschaften, welche im voraus separat bestimmt werden müssen, die Klassifikation durchführt. Das bedeutet auch wiederum, dass ein CNN viel mehr Neuronen besitzt. In dieser Arbeit hatte das MLP zum Beispiel 10 Eingabeneuronen, 800 Neuronen in den versteckten Ebenen und zum Schluss 2 Ausgabeneuronen, während das CNN allein schon in der Eingabeebene 256x256 Pixeldaten, also insgesamt 65536 Eingabeneuronen haben muss. Somit wird auch erwartet, dass CNN gegenüber eines MLP mehr Potential für eine genaue und effiziente Klassifikation besitzt und diese Vermutung lässt sich auch in diesem Vergleich der beiden Netzwerke gut wiederspiegeln. Auf der anderen Seite ist es aber auch zu erwähnen, dass man mit einem CNN häufig auf mögliche Speicher- und Leistungsprobleme des Rechners stoßen kann. Denn aufgrund der um einige Größenordnungen höhere Anzahl an Neuronen innerhalb eines CNN verursacht das Problem, dass der nötige Speicherplatz sowie die Laufzeit des Netzwerks ganz schnell über die Grenze hinaus steigen kann, das auch zum Teil innerhalb dieser Arbeit als ein Problem ergeben hat. Ein weiterer möglicher Nachteil eines CNN gegenüber MLP ist das Finden von optimalen Hyperparameter sowie die Validierung des Netzwerks. Die sehr große Anzahl an Hyperparameter sowie Neuronen und Gewichte erhöht die Komplexität eines Netzwerks auf das Maximum,

wodurch man oft nicht genau wissen und zurück folgen kann, was in einem Netzwerk passiert. Dadurch kann ein Validierungsverfahren beim CNN etwas schwieriger und möglicherweise auch fehleranfälliger als ein MLP sein.

8 Zusammenfassung und Ausblick

In dieser Arbeit wurden als eine mögliche Methode zur Signal- und Untergrundklassifikation für das BabyIAXO-Experiment die Neuronale Netzwerke betrachtet. Dazu wurden zwei verschiedene Arten von Netzwerken, jeweils das *Multilayer Perceptron (MLP)* und das *Convolutional Neural Network (CNN)* aufgebaut und trainiert. Das Ziel der Arbeit war dann letztendlich die Validierung der beiden Netzwerke. Es wurden sinnvolle Validierungsmethoden zur Überprüfung überlegt, ob das Netzwerk sich nachvollziehbar verhält und die erwartete Aufgaben richtig erfüllt.

Im ersten Teil der Arbeit ging es darum, die beiden betrachteten Netzwerke aufzubauen und zu optimieren. Für die Optimierung wurden die Netzwerke mit Trainingsdaten trainiert. Zuvor mussten die Daten zunächst passend generiert werden und Schnittwerte wurden an die Daten angelegt, um mögliche fehlerhafte Ereignisse zu gut wie möglich herauszufiltern. Der nächste Schritt zur Optimierung der Netzwerke war die Bestimmung der Hyperparameter. Dafür wurden die einzelne Hyperparameter in verschiedener Kombination variiert, um zu sehen, wann das Netzwerk das beste Ergebnis liefert. Allerdings sind die Wahlmöglichkeiten der Hyperparameter sehr groß und die Korrelation zwischen den Parameter ist auch sehr schwer einzuschätzen, weshalb die bestimmten Hyperparameter nur eine gewisse Annäherung zur Optimierung darstellt.

Im weiteren Teil wurden dann die beiden Netzwerke mit den ausgewählten Parametern und Daten trainiert. Nach Optimierung der Netzwerke folgte das Validierungsteil. Für die Validierung wurden an den erfolgreich trainierten Netzwerken die Validierungsdaten gegeben, um die Klassifikationsgenauigkeit der optimierten Netzwerke zu untersuchen. Der Validierungsprozess ließ sich bei beiden Netzwerken etwas unterschieden, da man beim CNN aufgrund von Speicherproblemen etwas eingeschränkte Validierungsmöglichkeiten hatte.
Für das MLP wurde zuerst der Einfluss der einzelnen geometrischen Eigenschaften auf das Netzwerk betrachtet und einige Eigenschaften aussortiert. Dann wurde das MLP zunächst mit neuen Daten aus allen Kalibrationstarget validiert. Dabei erreichte MLP bei einer 90% Untergrundunterdrückung eine Signaleffizienz von ca. 91%. Für die selbe Validierung für das CNN erreichte man bei gleicher Untergrundunterdrückung eine Signaleffizienz von über 96%. Des weiteren wurde die Abhängigkeit der Netzwerke auf die Energie bzw. Kalibrationstarget untersucht. Dabei wurde sowohl für MLP als auch CNN zunächst die Kalibrationsdaten in obere und untere Energiegruppen aufgeteilt und das Verhalten der Netzwerke auf die entsprechende Trainings- und Validierungsdaten betrachtet. Auch hier zeigte sich für CNN deutlich höhere Effizienz und Genauigkeit gegenüber das MLP. Man konnte eine deutlich geringere Energieabhängigkeit des CNN bei der Klassifikation feststellen. Für das MLP wurden noch zusätzlich die Validierungsdaten in sehr kleinen Energieabschnitten geteilt, um eine genauere Abhängigkeit der Klassifikationsgenauigkeit des Netzwerks zur Energie zu erfassen. Man konnte dabei feststellen, dass abweichende Verteilung der geometrischen Eigenschaften bei unterschiedlicher Energie eine negative Auswirkung auf die Genauigkeit gezeigt hat. Auch war tendenziell zu sehen, dass die Genauigkeit sowie die Effizienz eines Netzwerks sowohl beim CNN als auch MLP bei Ereignissen mit niedrigeren Energie (beim MLP vor allem unter 1 keV) signifikant verschlechtert hat.
Dann wurde die Signaleffizienz des MLP und CNN mit den Werten aus der Likelihood-Methode für die einzelnen Target angeschaut. Man konnte bei beiden Netzwerken über allen Energiebereichen lang eine signifikant höhere Effizienzen erreichen.

Abschließend wurde dann das CNN und MLP im Hinblick auf die Signaleffizienz sowie ihre Abhängigkeit von der Energie miteinander verglichen. Wie erwartet zeigte das CNN in allen Bereichen eine deutliche höhere Leistung gegenüber MLP. Auch im Hinblick auf die Energieabhängigkeit zeigte CNN bessere Ergebnisse.

CNN erreichte für alle acht Kalibrationstarget eine ähnlich hohe Signaleffizienz, während das MLP deutlich je nach Energie abweichende Effizienz erreichte. Trotz dessen stellten sich auch die Nachteile eines CNN heraus, unter anderem die Komplexität innerhalb des Netzwerks sowie vergleichsweise eine große Menge an zu verarbeitenden Daten, welche es einem schwerer macht, das Netzwerk optimal zu trainieren und auch sinnvoll zu validieren.

Zusammenfassend kann man sagen, dass beide neuronale Netzwerke MLP und CNN sehr viel Potential zur Verbesserung der Signal- und Untergrundklassifikation gegenüber der alten Methode wie Likelihood-Methode gezeigt haben. Das CNN hat dann nochmal vergleichsweise zur MLP eine deutlich höhere Leistung erbringen können. Es war außerdem eines der Ziele dieser Arbeit die Netzwerke sinnvoll zu validieren und mögliche zur Validierungsschritte zu überlegen. Insgesamt lässt sich sagen, dass die alle hier vorgeführten Validierungen sinnvolle und nachvollziehbare Ergebnisse geliefert haben und auch die Effizienz der beiden Netzwerke erwartungsgemäß bestimmen konnten. Trotzdem sollte erwähnt werden, dass im Hinblick auf die Optimierung der Netzwerke sowie die Validierung noch viel Potential vorhanden sind. Möglicherweise können vor allem beim CNN durch feinere Bestimmung der Hyperparameter noch mehr optimiert werden, um höhere Effizienzen des Netzwerks zu erreichen. Außerdem sollte bei größeren vorhandenen Speicher eine erweiterte Validierung des CNN möglich sein, wodurch das Netzwerk genauer im Details auf die Abhängigkeit von Energie sowie von anderen Aspekten untersucht werden können.

Literatur

[1] Baker, C. A. ; Doyle, D. D. ; Geltenbort, P. ; Green, K. ; Grinten, M. G. D. d. ; Harris, P. G. ; Iaydjiev, P. ; Ivanov, S. N. ; May, D. J. R. ; Pendlebury, J. M. ; Richardson, J. D. ; Shiers, D. ; Smith, K. F.: Improved Experimental Limit on the Electric Dipole Moment of the Neutron. In: *Physical Review Letters* 97 (2006), Sep, Nr. 13. http://dx.doi.org/10.1103/physrevlett.97.131801. – DOI 10.1103/physrevlett.97.131801. – ISSN 1079–7114

[2] Peccei, R. D. ; Quinn, H. R.: CP Conservation in the Presence of Pseudoparticles. In: *Phys. Rev. Lett. 38* (1977), 1440-1443. http://dx.doi.org/10.1103/PhysRevLett.38.1440. – DOI 10.1103/PhysRevLett.38.1440

[3] Dent, James B. ; Dutta, Bhaskar ; Newstead, Jayden L. ; Thompson, Adrian: Inverse Primakoff Scattering as a Probe of Solar Axions at Liquid Xenon Direct Detection Experiments. In: *Physical Review Letters* 125 (2020), Sep, Nr. 13. http://dx.doi.org/10.1103/physrevlett.125.131805. – DOI 10.1103/physrevlett.125.131805. – ISSN 1079–7114

[4] ResearchGate, Scientific F.: *Weighing the solar axion.* https://www.researchgate.net/figure/The-solar-axion-flux-expected-on-Earth-and-its-components-due-to-the-axion-electron_fig1_331334319. – Aufgerufen am 07.03.2022

[5] al., T. A.: Axion emission from red giants and white dwarfs. In: *Astroparticle Physics. Vol. 2* (1994), 175. http://dx.doi.org/10.1016/0927-6505(94)90040-X. – DOI 10.1016/0927–6505(94)90040–X

[6] Laget, J. M.: The Primakoff effect on a proton target. In: *Phys.Rev.C72:022202* (2005). arXiv:hep-ph/0502233

[7] M. Rosu, u.a.: A Solar Axion Search Using a Decommissioned LHC Test Magnet. (2000), 3. https://cds.cern.ch/record/5791

[8] S. Andriamonje, T. Dafni G. Fanourakis E. Ferrer Ribas H. Fischer J. Franz T. Geralis A. Giganon Y. Giomataris F.H. Heinsius K. Königsmann T. Papaevangelou K. Z. S. Aune A. S. Aune: A Micromegas detector for the CAST experiment. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment Vol. 518, Issues 1-2* https://doi.org/10.1016/j.nima.2003.10.074

[9] Collar, J.I ; Giomataris, Y: Possible low-background applications of MICROMEGAS detector technology. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 471 (2001), Sep, Nr. 1–2, 254–259. http://dx.doi.org/10.1016/s0168-9002(01)00986-x. – DOI 10.1016/s0168–9002(01)00986–x. – ISSN 0168–9002

[10] P. Abbon, u.a.: The Micromegas Detector of the CAST Experiment. In: *In: New J. Phys. Vol. 9* (2007), 170. https://iopscience.iop.org/article/10.1088/1367-2630/9/6/170

[11] X. Llopart, u.a.: Timepix, a 65 k programmable pixel readout chip for arrival time, energy and/or photon counting measurements. In: *Nucl. Instrum. Methods Phys. Res. Sect. A 581 (1)* (2007), 485-494. http://dx.doi.org/10.1016/j.nima.2007.08.079

[12] M. Chefdeville, u.a.: An electron-multiplying 'Micromegas' grid made in silicon wafer post-processing technology. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 556.2* (2006), 490–494. http://dx.doi.org/http://dx.doi.org/10.1016/j.nima.2005.11.065. – DOI http://dx.doi.org/10.1016/j.nima.2005.11.065

[13] Chefdeville, M. A.: Development of Micromegas-like gaseous detectors using a pixel readout chip as collecting anode. (2009). http://dare.uva.nl/document/123168

[14] T. Dafni, J. G.: Digging into Axion Physics with (Baby)IAXO. (2021). https://doi.org/10.48550/arXiv.2112.02286

[15] Zell, Andreas: *Simulation Neuronaler Netze [Simulation of Neural Networks]*. Addison-Wesley, 1994. – 73 S. – ISBN 3–89319–554–8

[16] Sherstinsky, Alex: Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. In: *Physica D: Nonlinear Phenomena* 404 (2020), Mar, 3-8. http://dx.doi.org/10.1016/j.physd.2019.132306. – DOI 10.1016/j.physd.2019.132306. – ISSN 0167–2789

[17] Nair, Geoffrey E. Vinod; Hinton H. Vinod; Hinton: *Rectified Linear Units Improve Restricted Boltzmann Machines*. Omnipress, 2010. – 807–814 S. – ISBN 9781605589077

[18] Rojas, Raul: *Theorie der Neuronalen Netze*. Springer, 1996. – 151 ff. S. – ISBN 3540563539

[19] Zhang, Aston ; Lipton, Zachary C. ; Li, Mu ; Smola, Alexander J.: Dive into Deep Learning. In: *arXiv preprint arXiv:2106.11342* (2021)

[20] *HDF5 documentation*. https://portal.hdfgroup.org/display/HDF5/HDF5. – Aufgerufen am 09.03.2022

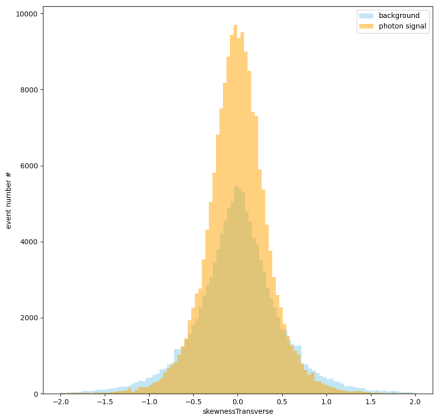
[21] LeCun, u.a.: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324. http://dx.doi.org/10.1109/5.726791. – DOI 10.1109/5.726791

[22] . https://f000.backblazeb2.com/file/vindaarNotes/status/index.html#org88226e1. – Aufgerufen am 10.03.2022

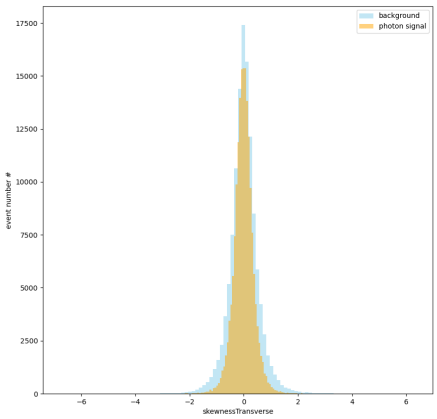
[23] *Github repository TimepixAnalysis/Analysis/ingrid*. https://github.com/Vindaar/TimepixAnalysis/blob/master/Analysis/ingrid/cdl_spectrum_creation.nim#L118-L230. – Aufgerufen am 10.03.2022

9 Anhang

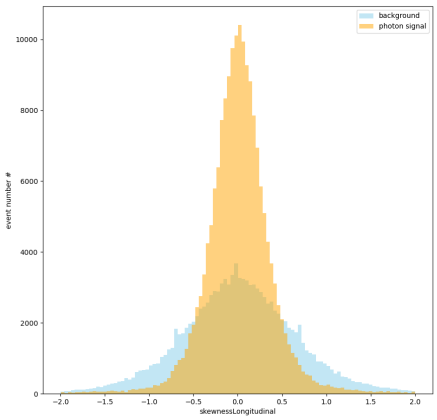
9.1 Signal- und Untergrundverteilung der geometrischen Eigenschaften für Silber



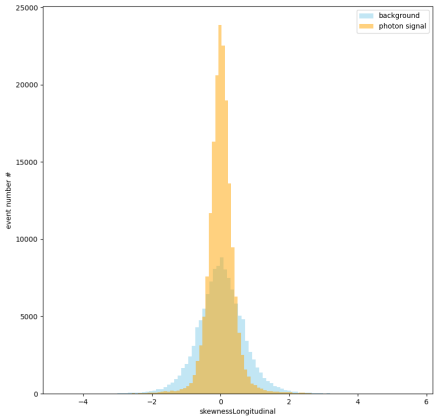
(a) skewness Transverse mit Schnitt



(b) skewness Transverse ohne Schnitt

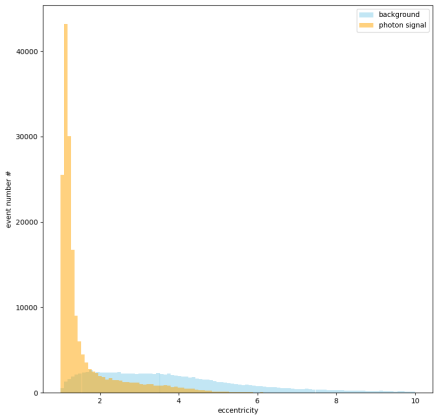


(c) skewness Longitudinal mit Schnitt

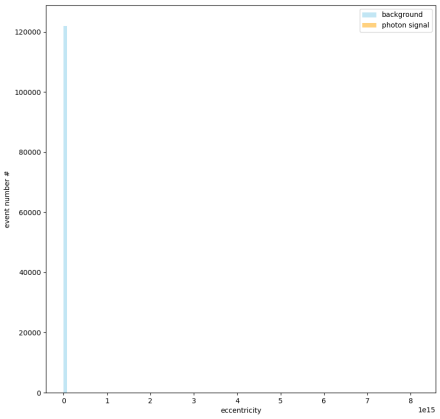


(d) skewness Longitudinal ohne Schnitt

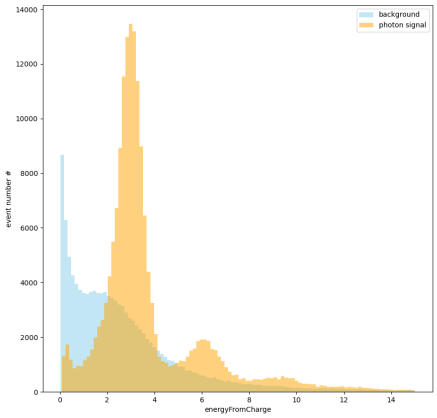
Abbildung 31: Signal- und Untergrundverteilung der geometrischen Eigenschaften mit und ohne Schnittanwendung auf die Kalibrationsdaten von Silber



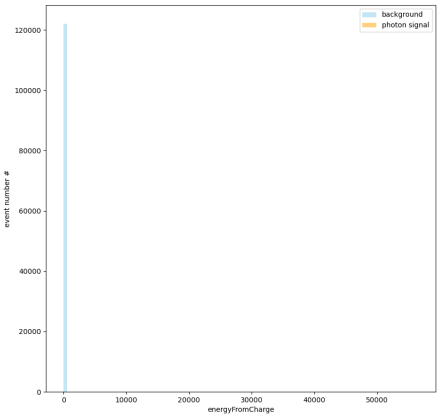
(a) Exzentrizität mit Schnitt



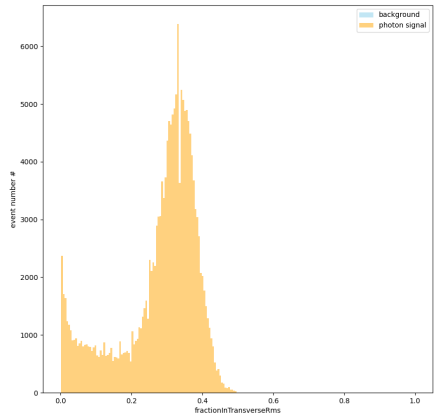
(b) Exzentrizität ohne Schnitt



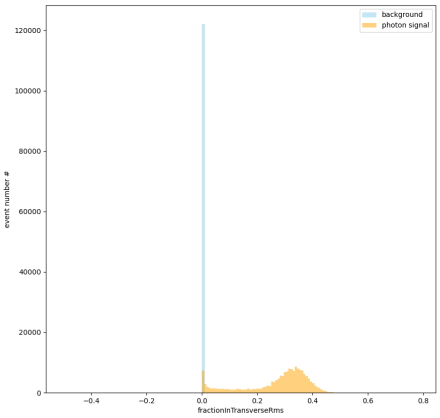
(c) Ladungsenergie mit Schnitt



(d) Ladungsenergie ohne Schnitt

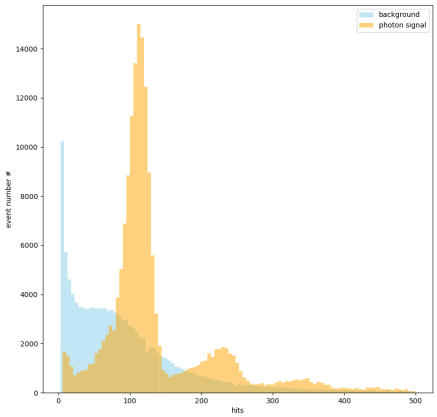


(e) Fraction In Transverse RMS mit Schnitt

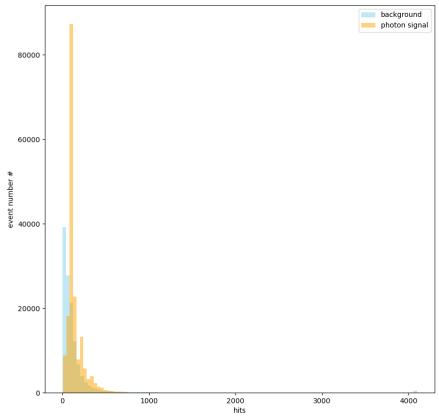


(f) Fraction In Transverse RMS ohne Schnitt

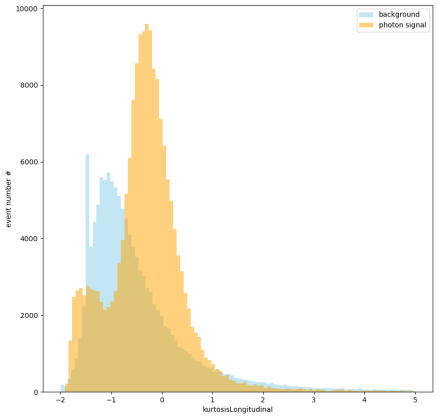
Abbildung 32: Signal- und Untergrundverteilung der geometrischen Eigenschaften mit und ohne Schnittanwendung auf die Kalibrationsdaten von Silber



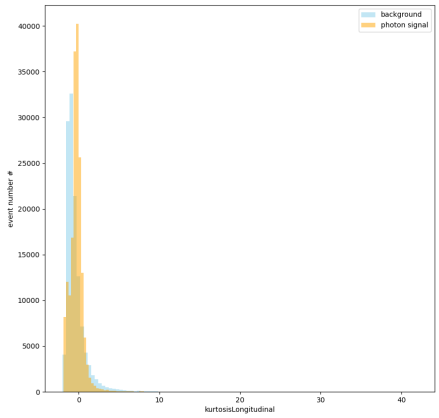
(a) Hits mit Schnitt



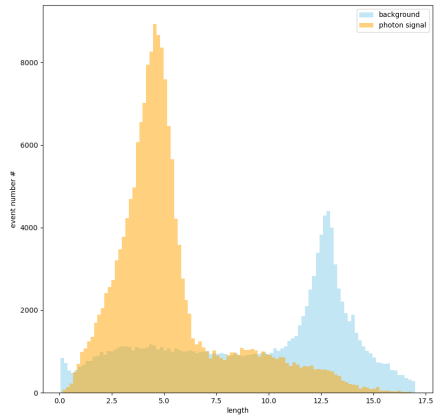
(b) Hits ohne Schnitt



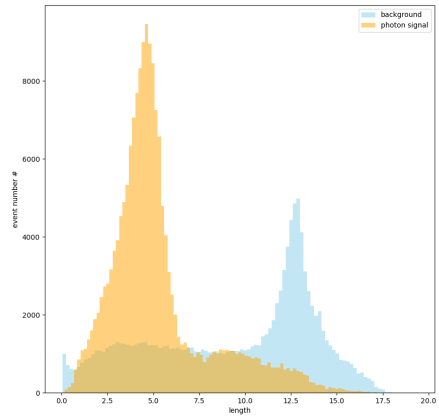
(c) kurtosis Longitudinal mit Schnitt



(d) kurtosis Longitudinal ohne Schnitt

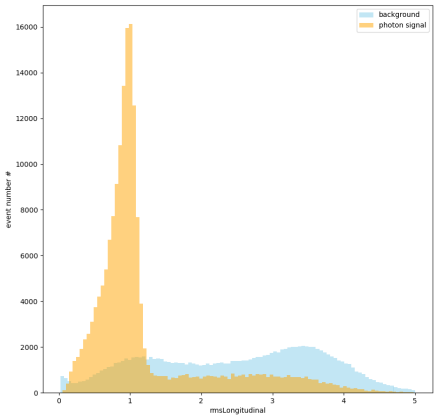


(e) Length mit Schnitt

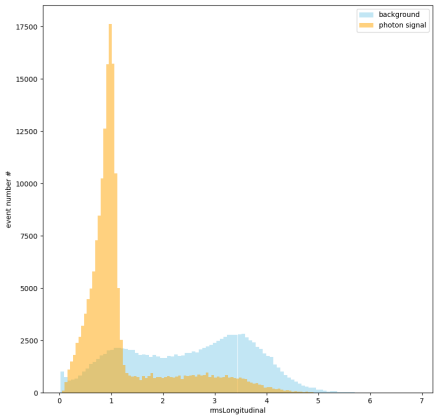


(f) Length ohne Schnitt

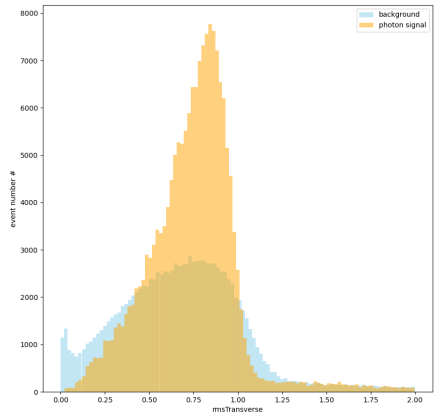
Abbildung 33: Signal- und Untergrundverteilung der geometrischen Eigenschaften mit und ohne Schnittanwendung auf die Kalibrationsdaten von Silber



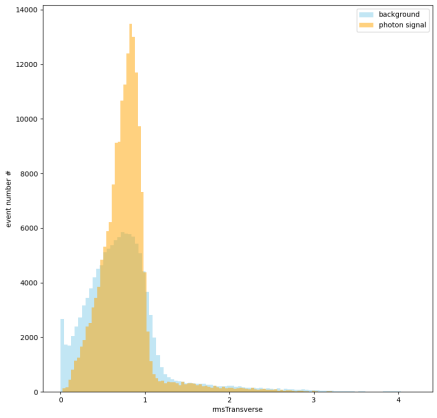
(a) RMS Longitudinal mit Schnitt



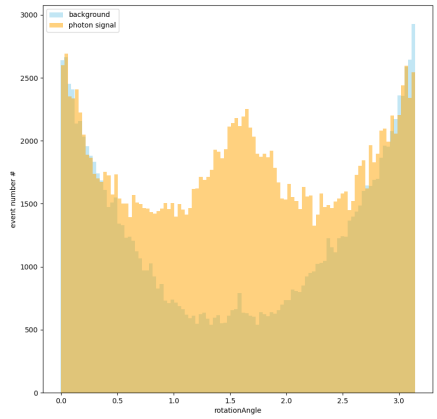
(b) RMS Longitudinal ohne Schnitt



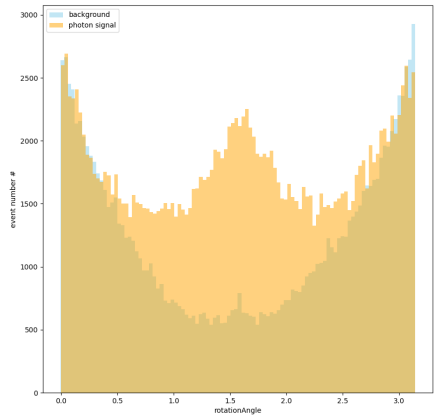
(c) RMS Transverse mit Schnitt



(d) RMS Transverse ohne Schnitt



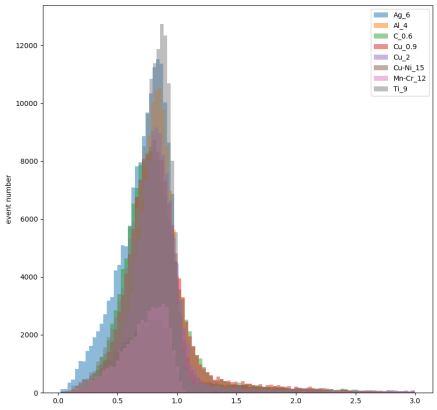
(e) Rotation Angle mit Schnitt



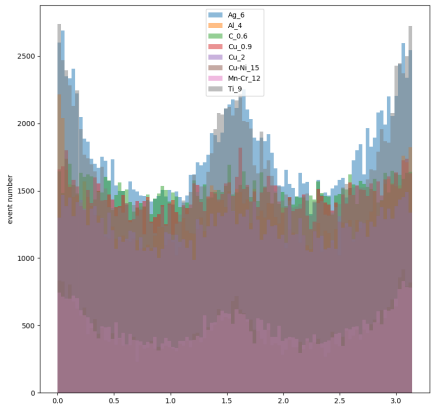
(f) Rotation Angle ohne Schnitt

Abbildung 34: Signal- und Untergrundverteilung der geometrischen Eigenschaften mit und ohne Schnittanwendung auf die Kalibrationsdaten von Silber

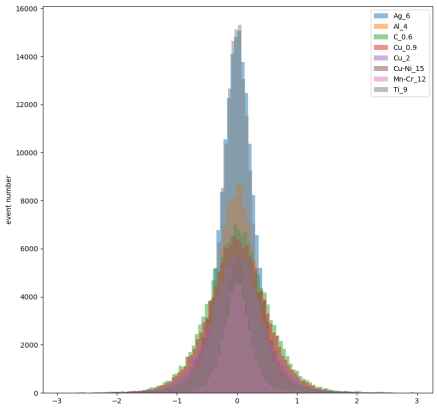
9.2 Verteilung der geometrischen Eigenschaften über alle Kalibrationstarget



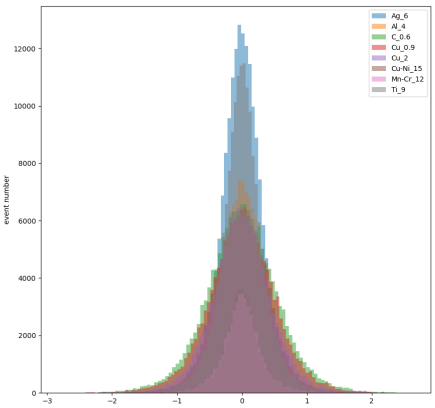
(a) RMS Transverse



(b) Rotation Angle

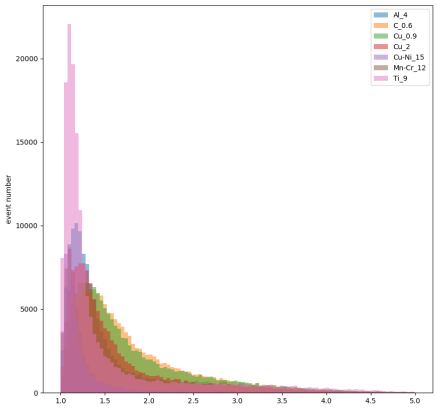


(c) skewness Longitudinal

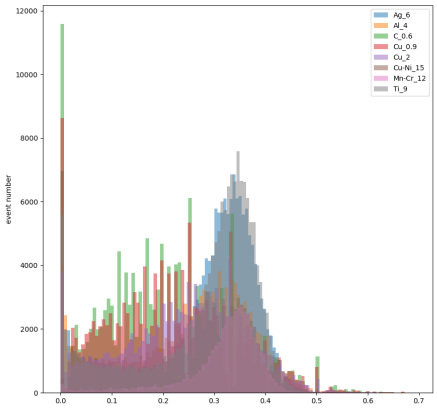


(d) skewness Transverse

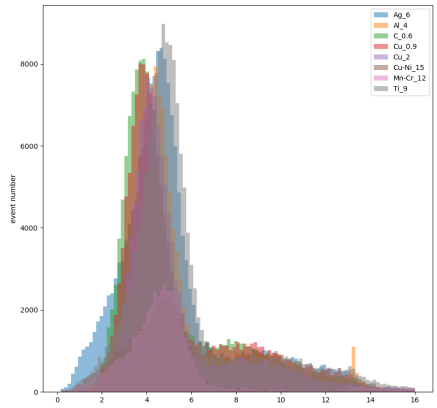
Abbildung 35: Die Verteilungen aller acht verwendeten Kalibrationsdaten werden für jede Eigenschaft übereinander dargestellt und verglichen.



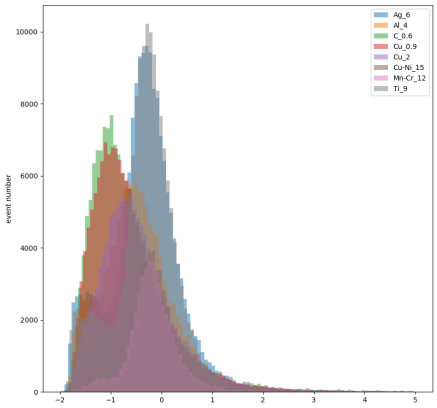
(a) Exzentrizität



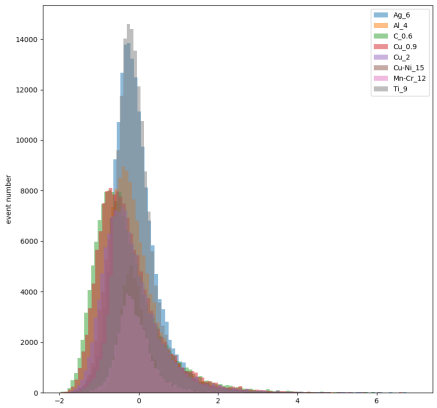
(b) Fraction In Transverse RMS



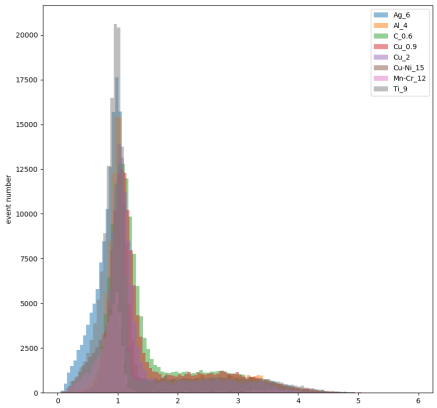
(c) Length



(d) kurtosis Longitudinal



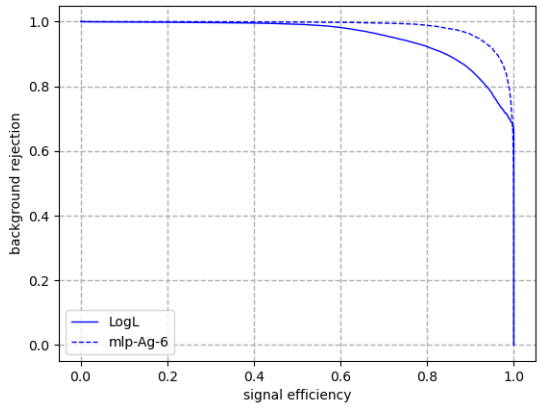
(e) kurtosis Transverse



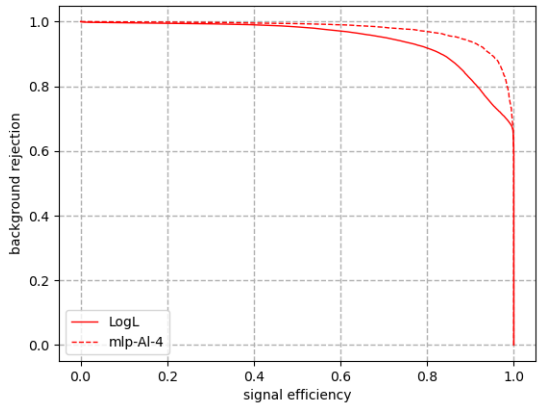
(f) RMS Longitudinal

Abbildung 36: Die Verteilungen aller acht verwendeten Kalibrationsdaten werden für jede Eigenschaft übereinander dargestellt und verglichen.

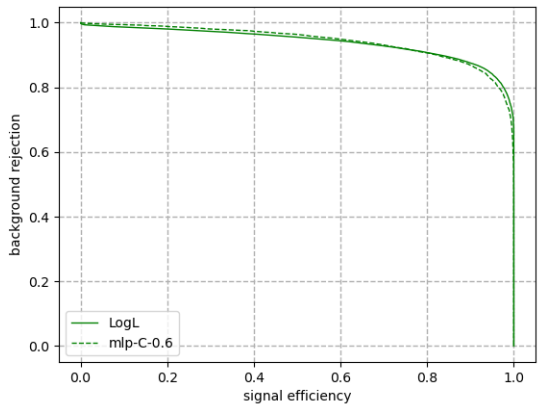
9.3 Vergleich ROC-Kurven der Likelihood-Verteilung mit MLP- und CNN-Verteilung der Ausgabeneuronen



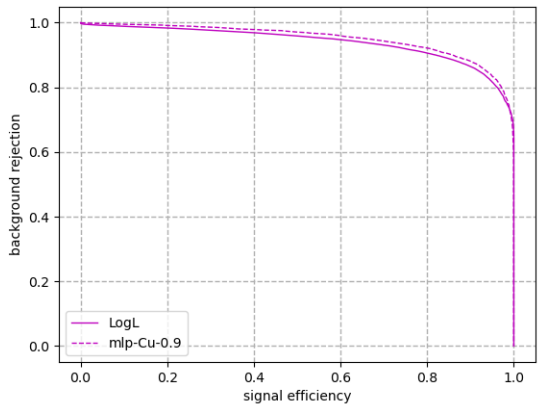
(a) Silber (Ag)



(b) Aluminium (Al)

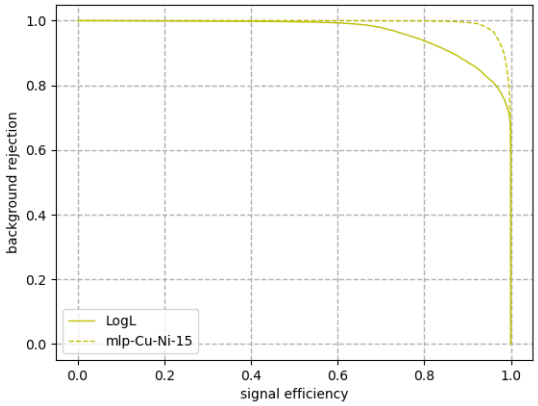


(c) Kohlenstoff (C)

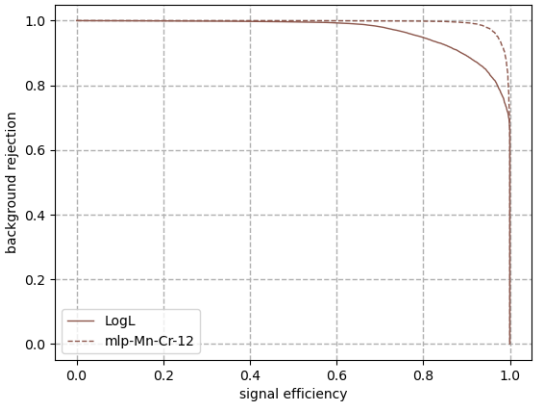


(d) Kupfer (Cu) (0,9kV)

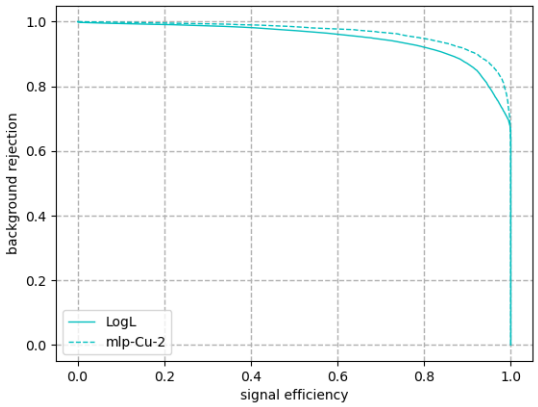
Abbildung 37: ROC-Kurven der Likelihood- und MLP-Ausgabeverteilung



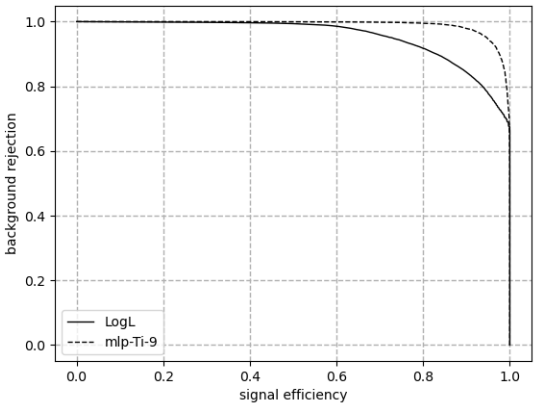
(a) Kupfer-Nickel (Cu-Ni)



(b) Moscovium-Chrom (Mn-Cr)

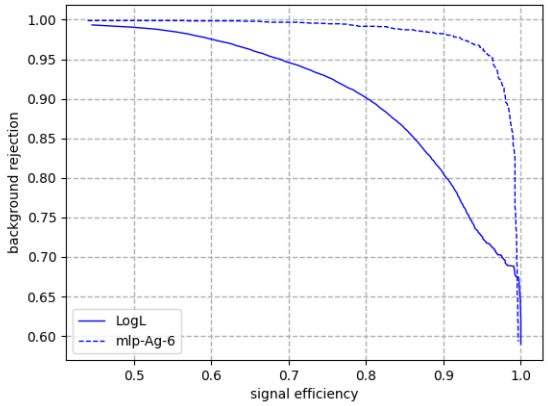


(c) Kupfer (Cu) (2kV)

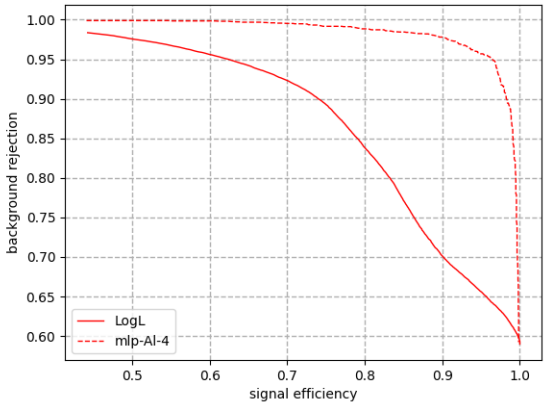


(d) Titanium (Ti)

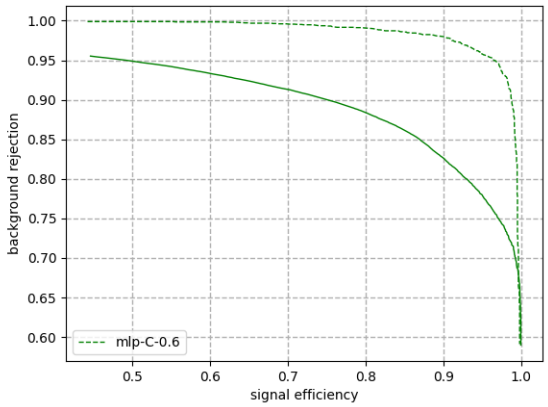
Abbildung 38: ROC-Kurven der Likelihood- und MLP-Ausgabeverteilung



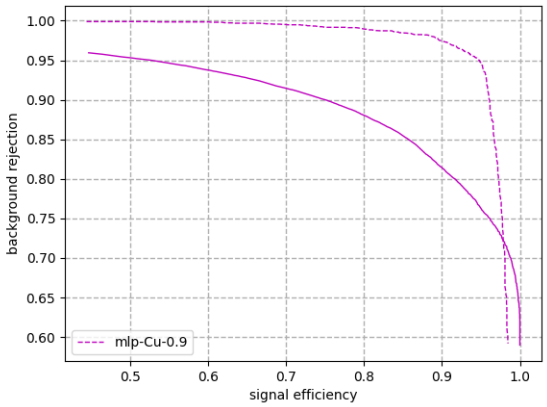
(a) Silber (Ag)



(b) Aluminium (Al)

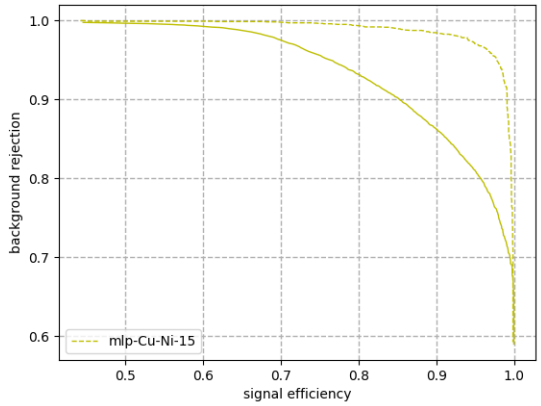


(c) Kohlenstoff (C)

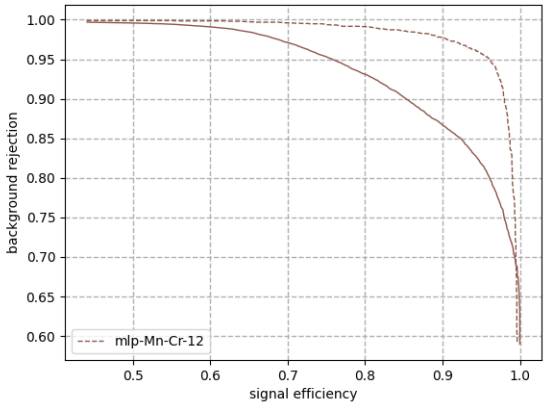


(d) Kupfer (Cu) (0,9kV)

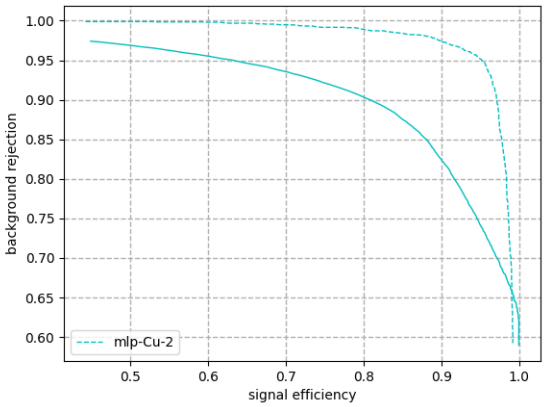
Abbildung 39: ROC-Kurven der Likelihood- und CNN-Ausgabeverteilung



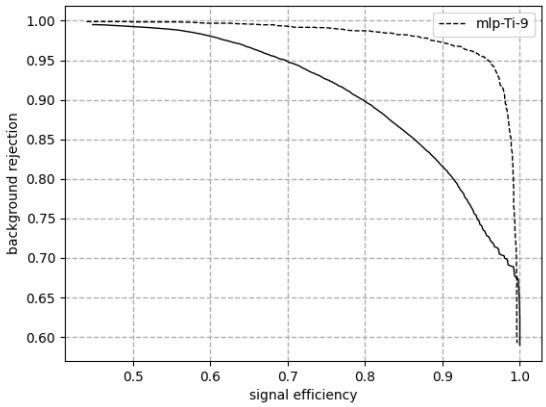
(a) Kupfer-Nickel (Cu-Ni)



(b) Moscovium-Chrom (Mn-Cr



(c) Kupfer (Cu) (2kV)



(d) Titanium (Ti)

Abbildung 40: ROC-Kurven der Likelihood- und CNN-Ausgabeverteilung