

**Verwendung von
Neuronalen Netzen zur
Untergrundminimierung der Reaktion
 $\gamma p \rightarrow p\eta'$**

JANIK KREIT

Bachelorarbeit in Physik
angefertigt im Helmholtz-Institut für Strahlen- und
Kernphysik

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

Oktober 2022

Inhaltsverzeichnis

| | |
|--|-----------|
| 1 Einleitung | 3 |
| 2 Das Experiment | 4 |
| 2.1 Elektron-Stretcher-Anlage | 4 |
| 2.2 Crystal Barrel und Mini-TAPS | 5 |
| 2.3 γp -Reaktion | 6 |
| 3 Theorie der Neuronalen Netze | 7 |
| 3.1 Aufbau von künstlichen Neuronalen Netzen | 7 |
| 3.1.1 Ein einzelnes Neuron | 7 |
| 3.1.2 Layer | 7 |
| 3.1.3 Aktivierungsfunktionen | 9 |
| 3.1.4 Das Trainieren | 9 |
| 3.2 Analyse des Neuronalen Netzes | 10 |
| 3.2.1 Änderung der Genauigkeit durch Ausschließen von Parametern | 10 |
| 3.2.2 Fuzzy Curves | 10 |
| 3.2.3 Weight of Evidence | 11 |
| 3.3 Verwendete Software | 11 |
| 3.3.1 ROOT | 12 |
| 3.3.2 Python | 12 |
| 4 Bilderkennung mithilfe Neuronaler Netze | 14 |
| 4.1 Datenset | 14 |
| 4.2 Vorhersage mit Python | 14 |
| 4.3 Vergleich zu Root | 17 |
| 5 Analyse von $\gamma p \rightarrow p\eta'$ | 18 |
| 5.1 Reduktion des Untergrundes mithilfe Cuts | 18 |
| 5.2 Erste Entwicklung des Neuronalen Netzes | 20 |
| 5.2.1 Eingabeparameter | 20 |
| 5.2.2 Zweiteilchenzerfall | 21 |
| 5.2.3 Feinjustierung | 23 |
| 5.3 Finale Entwicklung des Neuronalen Netzes | 25 |
| 5.3.1 Beachtung aller Teilchen | 25 |
| 5.3.2 Wahl von Layer und Neuronen | 27 |
| 5.3.3 Ergebnis | 28 |
| 5.4 Analyse des Neuronalen Netzes | 29 |
| 5.4.1 Ausschließen von Parametern | 29 |
| 5.4.2 Fuzzy Curves | 30 |
| 5.4.3 Weight of Evidence | 31 |
| 5.4.4 Physikalische Erklärung | 32 |
| 5.4.5 Vergleich aller Methoden | 33 |
| 5.5 Anwendung auf reale Daten | 33 |
| 5.5.1 Vergleich zur herkömmlichen Analyse | 35 |
| 6 Zusammenfassung und Ausblick | 37 |

1 Einleitung

Für eine Analyse der Reaktion $\gamma p \rightarrow p\eta'$ ist es notwendig das Spektrum dieser Reaktion von der anderer Teilchen zu unterscheiden. Beim Beschuss von energiereichen Photonen γ auf Protonen p können viele verschiedene Mesonen entstehen, wodurch das Herausfiltern des η' Mesons eine eigene Untersuchung bedarf. Die erzeugten Mesonen zerfallen nach kurzer Zeit in weitere Teilchen — unter anderem Photonen, die vom Detektor gemessen werden können. Je nach Reaktion unterscheidet sich die Anzahl der entstandenen Photonen, wodurch diese Ereignisse voneinander trennbar sind. Allerdings werden manche Photonen nicht registriert oder werden fälschlicherweise als solche erkannt, sodass das Extrahieren der η' Reaktion aufwendiger wird.

Neuronale Netze können in ihrer Funktion Zusammenhänge in Datensätzen erkennen, die einer herkömmlichen Analyse voraus sind. Die Ausgabe dieser Modelle kann zur Klassifizierung von Daten oder zur Bestimmung expliziter Werte genutzt werden. Durch Trainieren der Netzwerke werden sie konkret an ein komplexes Problem angepasst.

Ziel dieser Arbeit ist es, ein Neuronales Netz zu entwickeln, dass anhand von Informationen einer einzelnen Reaktion entscheiden kann, ob ein η' Meson entstanden ist oder nicht. Ein perfekt trainiertes Netzwerk könnte somit alle Teilchen extrahieren und die Untergrundminimierung wäre erfolgreich.

Im ersten Teil der Untersuchung wird der Aufbau vom CB-ELSA/TAPS Experiment in Bonn beschrieben und die betrachtete Reaktion genauer erläutert. Danach werden Neuronale Netze im Allgemeinen vorgestellt und ihre Funktionsweise erklärt. Zusätzlich sind Analysemethoden für die Netzwerke relevant — ebenso wie die Software, in der sie erstellt werden. Nachdem die Theorie beschrieben wurde, folgt eine erste praktische Anwendung dieser Systeme, indem handgeschriebene Zahlen von 0 bis 9 erkannt werden. Der Hauptteil widmet sich schließlich der Analyse des η' Mesons, in der Simulationen verwendet werden, um das Neuronale Netz zu trainieren und daraufhin auf echte, gemessene Daten anzuwenden.

Diese Arbeit soll zeitgleich als Beschreibung des Algorithmus zur spezifischen Anwendung von Neuronalen Netzen zur Untergrundminimierung dienen, der vollständig unter <https://github.com/janikkreit/ParticleAI> zu finden ist. Damit wird ermöglicht eine weiterführende Analyse diverser Reaktionen durchzuführen und die Vorteile von maschinellen Lernen zu nutzen, die im Weiteren verdeutlicht werden.

2 Das Experiment

2.1 Elektron-Stretcher-Anlage

Die Elektron-Stretcher-Anlage (ELSA) befindet sich unter dem Physikalischen Institut der Universität Bonn. Der Elektronen-Beschleuniger (Abbildung 2.1) liefert Elektronen mit einer maximalen Energie von 3,5 GeV, die wahlweise longitudinal polarisiert werden können [12].

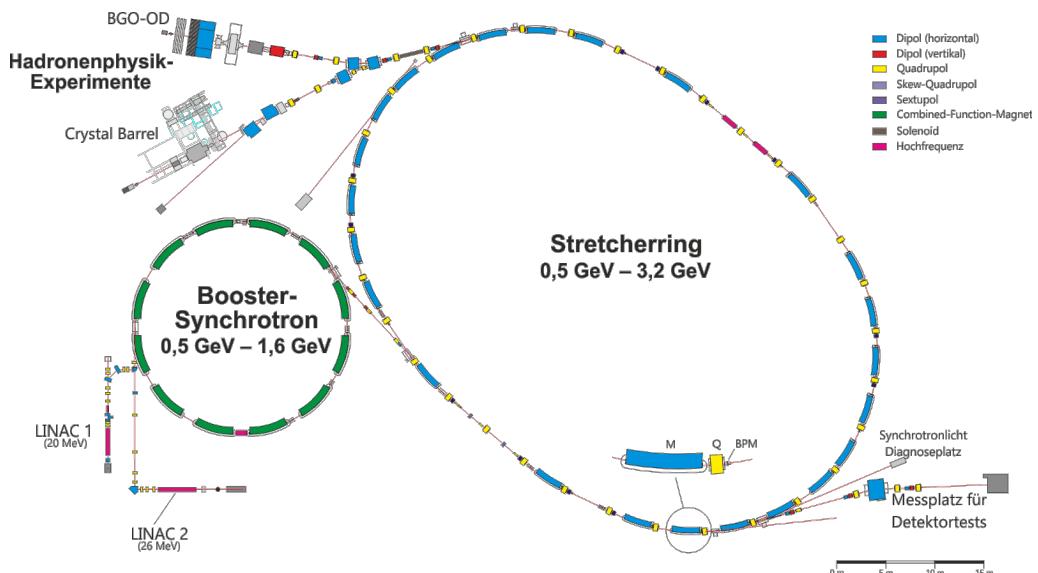


Abbildung 2.1: Elektronen-Stretcher-Anlage (ELSA) [12]

Nach der Produktion der Elektronen über eine thermische Quelle erfolgt die Beschleunigung in drei Stufen: LINAC¹, Booster-Synchrotron und Stretcherring. Die Elektronen werden für das Experiment aus der dritten Stufe extrahiert und gelangen in das Gonimeter, in dem durch Bremsstrahlungsprozesse Photonen erzeugt werden. Abhängig vom Bremsstrahlungs-Target und von wahlweise longitudinal polarisierten Elektronen können transversal oder zirkular polarisierte Photonen produziert werden [1, 7].

Die durch den Prozess gestreuten Elektronen werden in dem Tagging-System (Abb. 2.2 unten links) in einem Magnetfeld abgelenkt und detektiert. Durch den gemessenen Ablenkungsradius der Elektronen kann auf deren Energie E_e geschlossen werden. Da die Strahlenenergie E_0 bekannt ist, kann die Photonenenergie mittels $E_\gamma = E_0 - E_e$ rekonstruiert werden.

¹Abkürzung für den englischen Ausdruck Linear Accelerator.

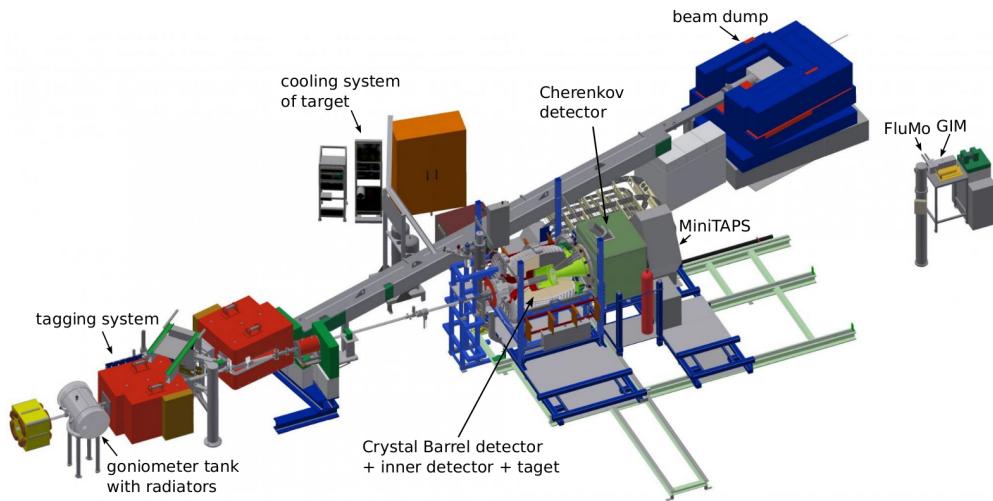


Abbildung 2.2: Aufbau des CB-ELSA/TAPS Experiments. [1]

2.2 Crystal Barrel und Mini-TAPS

Hinter dem Tagging-System befindet sich der Crystal Barrel Detektor (siehe Abb. 2.3) mit dem polarisiertem Target. Da der Strahl ebenso wie das Target polarisiert werden kann, ermöglicht das die Messung von Doppel-Polarisations-Observablen [1].

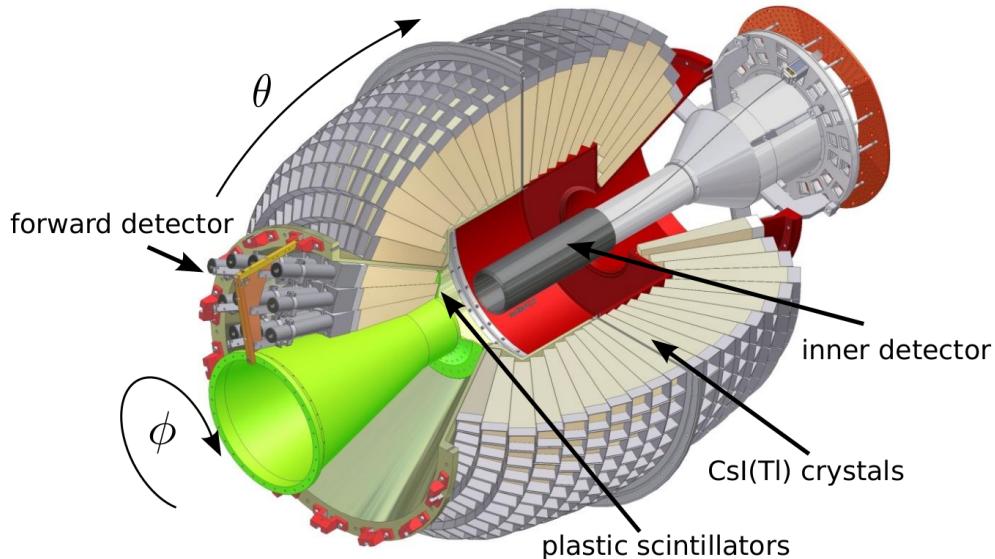


Abbildung 2.3: Crystal Barrel Detektor [1]

Der Crystal Barrel besteht aus insgesamt 24 Ringen, die fassförmig angeordnet sind. Die ersten drei Ringe bestehen aus jeweils 30 CsI(Tl)-Kristallen und bilden den Vorwärts-Detektor. Die Ringe 4 bis 23 besitzen jeweils 60 Kristalle, die einen Azimutwinkel $\Delta\phi = 6^\circ$ und einen Polarwinkel $\Delta\theta = 6^\circ$ abdecken. Der letzte Ring besteht wie die ersten drei Ringe aus 30 Kristallen und ist somit $\Delta\phi = 12^\circ$ und $\Delta\theta = 6^\circ$ breit. Damit wird der komplette Azimutbereich abgedeckt und der Polarwinkelbereich von 12° bis 156° [1]. Die in der Reaktion entstehenden Photonen erzeugen einen elektromagnetischen Schauer, welcher sich über mehrere Kristalle ausbreitet. Durch die Analyse der Energieverteilung in den Detektoren kann eine Winkelgenauigkeit von mehr als 6° erreicht werden [21].

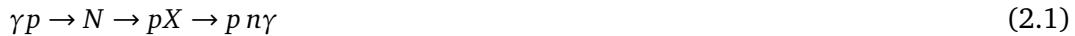
2 Das Experiment

Hinter dem Crystal Barrel in Strahlrichtung befindet sich der Cherenkov Detektor, der beim Durchtreten von hochenergetischen, geladenen Teilchen ein Signal abgibt. Die Geschwindigkeit für die Erzeugung von Cherenkov-Strahlung muss größer als die Lichtgeschwindigkeit im enthaltenden Medium sein. Das trifft auf Elektronen zu, die beim Experiment entstehen, und deren Detektion im Mini-TAPS daher gezielt gefiltert werden können [7].

Der Mini-TAPS Detektor (vgl. Abb. 2.2) befindet sich hinter dem Cherenkov Detektor und ist aus 216 hexagonalen BAF_2 -Kristallen aufgebaut. Er deckt den Polarwinkelbereich von 1° bis 12° ab [1].

2.3 γp -Reaktion

In dem Experiment wird flüssiger Wasserstoff² als Target verwendet [1], was zu der Teilchenreaktion (2.1) [4] führt. Theoretisch sind auch Reaktionen möglich, die neben dem Proton weitere geladene Teilchen enthalten. Diese können allerdings durch Voranalysen gefiltert werden oder sind statistisch vernachlässigbar [9].



Hierbei beschreibt N eine Protonresonanz, die durch das energiereiche Photon γ angeregt wird und X bezeichnet ein oder mehr Mesonen, die auf Grund der starken Wechselwirkung erzeugt werden. Die generierten Hadronen zerfallen daraufhin in n Photonen, die im CB und Mini-TAPS gemessen werden.

Die spätere Analyse nutzt die Ereignisselektion von J. KRAUSE [9] als Grundlage. Es soll nur kurz zusammengefasst werden, welche Reaktionen für diese Analyse relevant sind. Für weitere Informationen wird auf seine Arbeit verwiesen.

Bei einem Energiebereich der Strahlenenergie von 1400 MeV bis 2800 MeV werden die Reaktionen $\gamma p \rightarrow p\eta'$, $p\eta$, $p\pi^0$, $p2\pi^0$, $p\pi^0\eta$ und $p\omega$ betrachtet. In dieser Arbeit soll das η' untersucht werden, dessen wahrscheinlichsten Zerfälle in Tabelle 2.1 gezeigt sind. Es wurde der Kanal gewählt, der zu 2,3 % in zwei Photonen zerfällt, da dieser den einfachsten Endzustand mit drei Teilchen gewährt [9]. Teilchen wie z.B. $2\pi^0$ oder ω können in mehr Photonen (vier und drei) zerfallen und sollten in der Analyse theoretisch nicht auftreten. Jedoch ist es möglich, dass im Detektor zwei Photonen als ein Photon registriert werden, oder dass Teilchen in Richtungen fliegen, in denen sich keine Detektoren befinden. Dadurch gehen Photonen verloren und die erwähnten Mesonen tauchen in der Untersuchung auf.

| Zerfall | Anteil |
|--|-------------------------|
| $\pi^+\pi^-\eta$ | 42,6 % |
| $\rho^0\gamma(\rightarrow\pi^+\pi^-\gamma)$ | 28,9 % (28,9 %) |
| $\pi^0\pi^0\eta(\rightarrow 6\gamma)$ | 22,8 % (8,8 %) |
| $\omega\gamma(\rightarrow\pi^+\pi^-\pi^0\gamma/\pi^0\gamma\gamma)$ | 2,52 % (2,2 % / 0,21 %) |
| $\gamma\gamma$ | 2,3 % |

Tabelle 2.1: Die fünf wahrscheinlichsten Zerfälle vom η' Meson. Die wahrscheinlichsten weiteren Zerfälle sind in den Klammern angegeben [5].

²In weiteren Experimenten am Crystal Barrel wird Butanol verwendet [1].

3 Theorie der Neuronalen Netze

3.1 Aufbau von künstlichen Neuronalen Netzen

Die künstlichen Neuronalen Netze (NN) adaptieren in Funktionsweise und Aufbau ihr biologisches Vorbild. Die Nervenzellen — auch Neuronen genannt — bilden die Recheneinheiten im Gehirn und sind für die Datenverarbeitung verantwortlich. Die Synapsen hingegen bezeichnen die Verbindungen zwischen den Neuronen [8]. Dennoch unterscheiden sich die biologischen Neuronalen Netze von den künstlichen erheblich in der Anzahl der Neuronen. Unser Nervensystem erreicht dabei eine Anzahl von 100 Milliarden Nervenzellen [8], während die heutigen Computer nur Neuronale Netze mit ca. 1000 Neuronen³ zulassen.

3.1.1 Ein einzelnes Neuron

Um ein gesamtes künstliches Neuronales Netz⁴ zu verstehen, wird zunächst ein einzelnes Neuron betrachtet. In Abbildung 3.1 sind n Eingabeparameter (x_1, x_2, \dots, x_n) zu sehen, die jeweils mit einer Gewichtung ($w_{1j}, w_{2j}, \dots, w_{nj}$) multipliziert werden. Daher gibt es ebenfalls n Gewichtungen. Der Index j bezeichnet hierbei das Neuron. Alle Eingabewerte werden mit den Gewichtungen aufsummiert und an eine Aktivierungsfunktion φ weitergegeben. Wie diese Funktionen explizit aussehen können, wird in Kapitel 3.1.3 gezeigt. Zusätzlich wird ein Schwellenwert θ_j und die Aktivierungsfunktion übergeben. Aus diesem System an Rechenschritten folgt dann ein Ausgabeparameter/Aktivierung o_j .

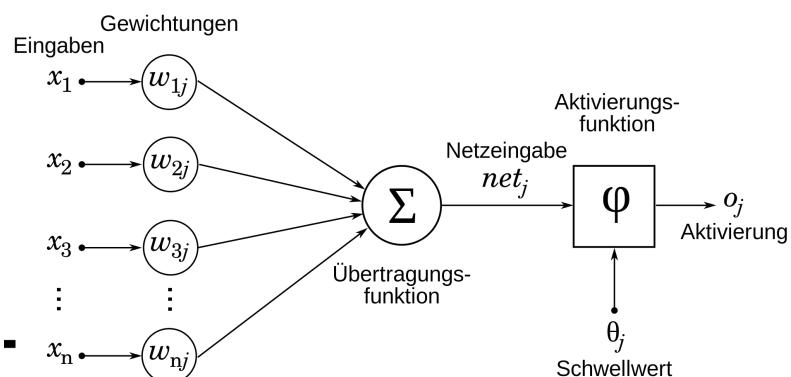


Abbildung 3.1: Schematische Darstellung für ein künstliches Neuron mit n Eingabeparametern x , Gewichtungen w , einem Schwellenwert θ und Ausgabewert o . Grafik von [10].

3.1.2 Layer

Die gleichen Eingaben x können an mehrere Neuronen übergeben werden, um eine Vielzahl von Ausgaben o zu erhalten. Die Gewichtungen w und der Schwellenwert θ unter-

³Technisch lassen sich auch NN mit mehr Neuronen realisieren. Der Zeitaufwand für das Trainieren ist hierbei aber der limitierende Faktor.

⁴Von nun an wird auf den Hinweis *künstlich* verzichtet.

3 Theorie der Neuronalen Netze

scheiden sich in jedem Neuron. Wird eine Anzahl von m Neuronen mit $1 \leq j \leq m$ und n Eingaben mit $1 \leq i \leq n$ betrachtet, so ergeben sich $n \cdot m$ Gewichtungen w_{ij} , m Schwellenwerte θ_j und m Ausgaben o_j . Der Zusammenschluss dieser Neuronen wird dann ein *Layer*⁵ genannt.

Mathematisch wird der Ausgabewert o_j über den Zusammenhang (3.1) [13] berechnet.

$$o_j = \varphi(w_{1j}x_1 + w_{2j}x_2 + \dots + w_{nj}x_n + \theta_j) \quad (3.1)$$

Um alle Ausgaben zu erhalten, fassen wir die m Gleichungen (3.1) zusammen und erhalten Gleichung (3.2).

$$\mathbf{o} = \varphi(\mathbf{W}\mathbf{x} + \boldsymbol{\theta}) \quad (3.2)$$

\mathbf{o} und $\boldsymbol{\theta}$ sind m -dimensionale Vektoren und \mathbf{x} hat n Dimensionen. φ ist weiterhin die Aktivierungsfunktion — allgemein gilt $\varphi : \mathbb{R}^m \rightarrow \mathbb{R}^m$. Zusätzlich ist \mathbf{W} eine $m \times n$ Matrix mit den Komponenten

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix}. \quad (3.3)$$

Diese Beschreibung ermöglicht eine einfache Erweiterung auf ein NN mit mehreren Layern. Die Verknüpfung wird erreicht, wenn eine Ausgabe o_j als eine Eingabe x_i an einem neuen Neuron verwendet wird. Als Beispiel ist in Abbildung 3.2 ein NN mit drei Eingabeparametern, zwei versteckten⁶ Layer mit jeweils fünf Neuronen und einem Ausgabeparameter dargestellt. Es ist unverkennbar, dass selbst für ein kleines Netzwerk eine erhebliche Anzahl an Verbindungen zwischen den Neuronen entsteht. Dadurch wird der enorme Rechenaufwand deutlich, der anfällt, wenn beispielsweise ein NN mit 100 Neuronen in zwei versteckten Layern, trainiert werden muss.

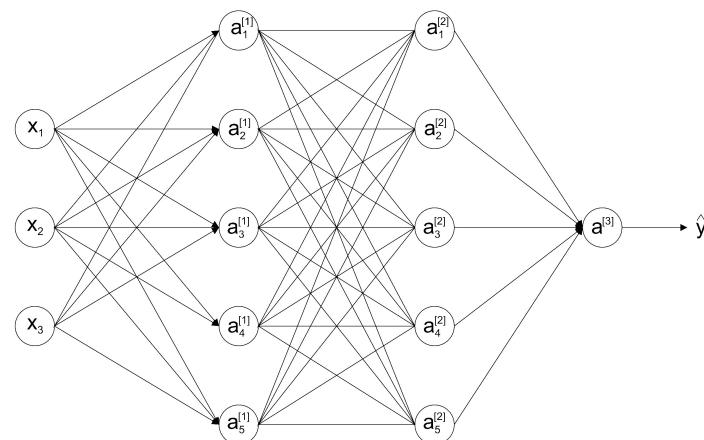


Abbildung 3.2: Neuronales Netz mit drei Eingabeparameter, zwei versteckten Layer mit jeweils fünf Neuronen und einem Ausgabeparameter. Grafik von [6].

⁵vom Englischen, deutsch *Schichten*

⁶Der Zusatz *versteckt* bezeichnet die Layer zwischen dem Eingabe- und Ausgabelayer.

3.1.3 Aktivierungsfunktionen

Die Aktivierungsfunktionen sind maßgeblich dafür verantwortlich, wie Eingabeparameter an einem Neuron selektiert und verarbeitet werden. Die Funktionen weisen im Allgemeinen ein nicht-lineares Verhalten auf [20]. Als Veranschaulichung werden zwei Beispiele diskutiert.

Die **Sigmoid**-Funktion (3.4) [20] weist einen Wertebereich $\mathbb{W} = (0, 1)$ auf, wobei die Funktion für Parameter $x \rightarrow -\infty$ gegen 0 und für $x \rightarrow \infty$ gegen 1 konvergiert.

$$\varphi_{\text{Sigmoid}}(x) = \frac{1}{1 + \exp(-x)} \quad (3.4)$$

Die **Rectified Linear Unit (ReLU)** Aktivierungsfunktion (3.5) [20] hat hingegen einen linearen Verlauf für $x > 0$ und ist Null für $x \leq 0$.

$$\varphi_{\text{ReLU}}(x) = \max(x, 0) \quad (3.5)$$

Zur Veranschaulichung sind die beiden Funktionen in Abbildung 3.3 dargestellt.

Der grundlegende Sinn der Funktionen besteht darin, dass wenn der übergebende Wert (siehe Argument in Gleichung (3.2)) negativ ist, die Funktionen die Ausgabe unterdrücken, indem diese sehr nahe oder gar gleich Null sind. Für positive Werte ist die Ausgabe größer als Null und trägt somit zur weiteren Berechnung im NN bei.

Der Vorteil der ReLU-Funktion ist dabei, dass die Ausgabe nicht wie bei Sigmoid konvergiert, sondern proportional zum übergebenden Wert steigt. Dadurch tragen solche Eingaben stärker zur Entscheidung bei und das NN prägt sich stärker aus. Außerdem werden Werte ≤ 0 vollkommen unterdrückt.

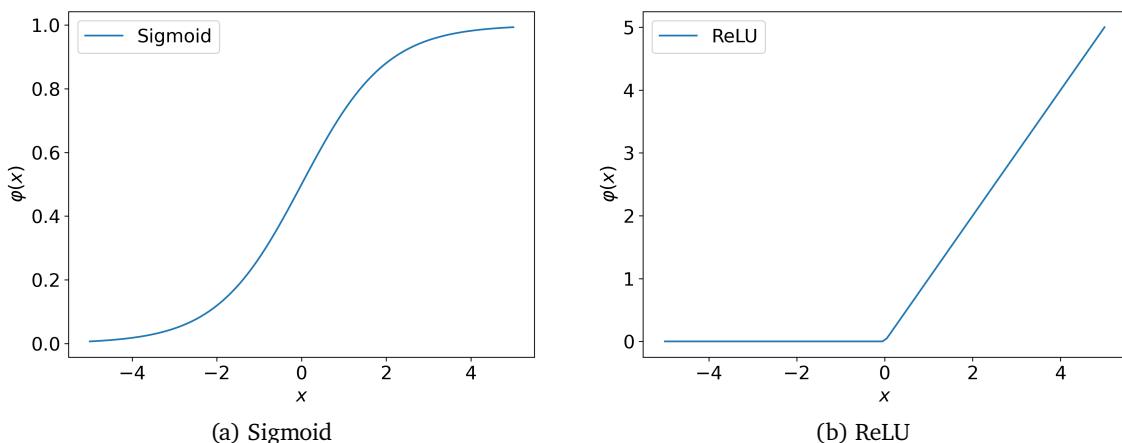


Abbildung 3.3: Sigmoid- und ReLU-Aktivierungsfunktion für das Neuronale Netz

3.1.4 Das Trainieren

Das NN soll für bestimmte Datensätze Vorhersagen treffen, auf die es zuvor trainiert wurde. Das können beispielsweise Bilder von Objekten sein, in denen das NN auf Grund der Werte der einzelnen Pixel erkennen kann, ob es sich bei dem Objekt um einen Stuhl oder einen Tisch handelt. Wichtig ist aber, dass das NN nur korrekte Vorhersagen treffen kann, wenn es auch vorher explizit mit diesem Objekt trainiert wurde. Wird dem NN z.B. ein Bild von

3 Theorie der Neuronalen Netze

einem Pferd übergeben, könnte es sich zu 99 % sicher sein, dass es ein Tisch ist. Dadurch wird auch deutlich, dass das Netzwerk keineswegs wirklich intelligent ist, sondern lediglich die Aufgaben gut erfüllt, auf die es trainiert wurde.

Das Konzept eines Trainings-Algorithmus umfasst mehrere Schritte. Das untrainierte NN berechnet für jeden Datensatz eine Ausgabe, die anfangs noch komplett willkürlich ist und vergleicht diese mit dem Sollwert. Dadurch lässt sich bestimmen, wie sich Gewichtungen und Schwellenwerte des NN verändern müssen, damit der Fehler kleiner und das Netzwerk optimiert wird. Diese Prozedur erfolgt in mehreren Schritten — den sog. *Epochen* [14]. Die genaue Beschreibung dieser Algorithmen würde diese Arbeit jedoch überschreiten. Allerdings können die Optimierungs- und Fehlerfunktionen bei dieser Prozedur angepasst werden, was in Kapitel 5.2.3 durchgeführt wird und die Effizienz des NN deutlich verbessert.

3.2 Analyse des Neuronalen Netzes

Für die spätere Analyse des NN, ist es interessant, welche Eingabeparameter maßgeblich für eine Entscheidung sind. Das ist einerseits nützlich, wenn durch Weglassen einzelner Eingaben die Laufzeit erheblich verbessert werden kann. Andererseits können die Daten an sich besser verstanden werden, das eine fortgehende Analyse ermöglicht.

Es werden drei Methoden vorgestellt, die im Kapitel 5.4 zum Einsatz kommen.

3.2.1 Änderung der Genauigkeit durch Ausschließen von Parametern

Um herauszufinden, wie wichtig ein bestimmter Eingabe-Parameter ist, wird das NN ohne diesen Parameter trainiert und die Genauigkeiten bei der Vorhersage verglichen. Weicht dieser Wert stark von der Effizienz ab, die das NN erzielt, wenn es mit allen Parametern trainiert wird, ist dieser Eingabe-Parameter besonders wichtig.

3.2.2 Fuzzy Curves

Die zweite Methode basiert auf einer Arbeit von Y. LIN und G.A. CUNNINGHAM (1995) [2] und bewertet die Parameter, ohne dass das NN vorher trainiert werden muss. Es sei ein Netzwerk mit Eingabeparametern (x_1, x_2, \dots, x_N) und einem Ausgabeparameter y , das mit K Datensätzen trainiert wird. Es wird x_{ik} als den i -ten Parameter im Datensatz k bezeichnet, wobei $k \leq K$ ist. Die *Fuzzy Membership Function* ϕ_{ik} [2] für ein frei wählbares x_i lautet

$$\phi_{ik}(x_i) = \exp\left(-\left(\frac{x_{ik} - x_i}{b}\right)^2\right). \quad (3.6)$$

Der Parameter b wird in der Arbeit [2] nicht beschrieben und wird daher in der Analyse 5.4 gewählt. Damit berechnen sich die *Fuzzy Curves* (3.7) [2]

$$C_i(x_i) = \frac{\sum_{k=1}^K \phi_{ik}(x_i) y_k}{\sum_{k=1}^K \phi_{ik}(x_i)}, \quad (3.7)$$

welche die Bedeutsamkeit eines Parameters für das NN charakterisiert. Wird die Funktion C_i in Bezug zu x_i dargestellt und weist einen konstanten Verlauf auf, ist der Eingabe-Parameter x_i nicht ausschlaggebend für die Ausgabe y des NN. Die Bedeutsamkeit der Parameter kann daraufhin über das Schwanken der Kurve eingestuft werden [2].

3.2.3 Weight of Evidence

Die Methode *Weight of Evidence* von M. ROBNIK-ŠIKONJA und I. KONONENKO (2008) [3] beschreibt die Bedeutsamkeit der Parameter für ein einzelnes Datenobjekt. Anders als bei den Methoden in Abschnitt 3.2.1 und 3.2.2 kann hier nicht nur der Datensatz als Gesamtes analysiert werden, sondern ist in der Lage, spezifische Entscheidungen des NN zu untersuchen.

Dafür wird ein Modell mit allen Parameter trainiert und zusätzliche NN erstellt, bei dem ein Parameter fehlt. Um die Wichtigkeit einer Eingabe zu bestimmen, werden die Chancen verglichen, dass sich das vollständige Modell und das Netzwerk mit diesem fehlenden Parameter für eine Ausgabe entscheidet. Unterscheiden sich die Chancen stark voneinander ist die Eingabe bedeutend.

In Gleichung (3.8) [3] werden die Chancen (englisch *odds*) über den Logarithmus zur Basis 2 verrechnet, um das Gewicht WE_i für den i -ten Parameter zu erhalten.

$$WE_i(y|\mathbf{x}) = \log_2(\text{odds}(y|\mathbf{x})) - \log_2(\text{odds}(y|\mathbf{x}_{\setminus i})) \quad (3.8)$$

y steht für die Ausgabe und \mathbf{x} für die Eingabe-Parameter. In Gleichung (3.9) [3] werden die Chancen für eine Ausgabe bestimmt.

$$\text{odds}(y|\mathbf{x}) = \frac{p(y|\mathbf{x})}{1 - p(y|\mathbf{x})} \quad (3.9)$$

Die Wahrscheinlichkeit $p(y|\mathbf{x})$ beschreibt, wie sicher sich das NN ist, dass die Ausgabe y zu den Eingabe-Parametern \mathbf{x} gehört. Analog beschreibt $p(y|\mathbf{x}_{\setminus i})$ ebenso die Sicherheit, wobei $\mathbf{x}_{\setminus i}$ die Eingabe-Parameter ohne den i -ten Parameter bezeichnet.

Schließlich gibt $WE_i(y|\mathbf{x})$ einen Wert für einen bestimmten Eintrag \mathbf{x} mit Ausgabe y aus, das die Bedeutsamkeit eines Eingabeparameters mit Index i beschreibt. Ist $WE_i(y|\mathbf{x})$ positiv, spricht der i -te Parameter dafür, dass sich das NN *für* die Ausgabe y entscheidet. Ein negativer Wert spricht daher *gegen* eine Ausgabe [3].

3.3 Verwendete Software

Für die Erstellung eines NN wurden zwei Systeme untersucht:

1. *TMultiLayerPerceptron* von *ROOT* in *C* [19]
2. *Sequential Model* von *Tensorflow-Keras* in *Python* [16]

Die Klasse in *ROOT* hat den großen Vorteil, dass die Datenübergabe für die spätere Analyse (siehe Kapitel 5) sehr einfach wäre, da alle Systeme bereits auf *ROOT* basieren. *Python* wurde hier als zweite Option gewählt, um die Ergebnisse in *C* bewerten und vergleichen zu können.

Es soll nun kurz der Programmcode für das Definieren und Trainieren des NN der beiden Systeme beschrieben werden.

3.3.1 ROOT

Im Algorithmus 3.1 wird gezeigt, wie ein NN mit zwei versteckten Layer in *ROOT* definiert wird.

Der erste Layer hat 50 und der zweite 20 Neuronen. Es werden zwei Eingabeparameter übergeben (*input1* & *input2*) und eine Ausgabe (*output*). Das @-Zeichen steht für das Normalisieren der Daten und wird für ein erfolgreiches NN empfohlen, da so sehr große oder sehr kleine Werte vermieden werden. Die Eingaben stammen von dem im zweiten Argument angegebenen *ROOT Tree*⁷. Die letzten Argumente geben an, welche Daten für das Trainieren und welche für das Testen des NN genutzt werden sollen. In dem Fall werden die Hälfte der Daten zum Trainieren genutzt. Diese Trennung dient für die Aussagekräftigkeit des Modells, da die Genauigkeit später über die Test-Daten bestimmt wird, mit denen das NN nicht trainiert wurde. Um sortierte Werte zu vermeiden, werden den Daten automatisch gemischt.

Nach der Definition des NN in Zeile 1 kann es mit dem Befehl *mlp->Train* (Zeile 2) trainiert werden, wobei die Anzahl der Epochen im ersten Argument angegeben wird. Die weiteren Angaben dienen der Ausgabe des Fortschritts beim Trainieren, welche alle zehn Epochen aktualisiert wird.

Im Zeile 3 kann über den *mlp->Evaluate* Befehl, eine Vorhersage *y_pred* aus den Testdaten *x_test* bestimmt werden. Das Argument 0 gibt den Index der Ausgabe an.

```

1 TMultilayerperceptron *mlp = new TMultilayerperceptron("@input1,@input2
   :50:20:output", tree, "Entry$%2", "(Entry$+1)%2");
2 mlp->Train(100, "text, graph, update=10");
3 y_pred = mlp->Evaluate(0, x_test);

```

Algorithmus 3.1: Das Modell mit 50 Neuronen im ersten Layer und 20 Neuronen im zweiten Layer wird in Zeile 1 deklariert. In Zeile 2 wird es mit 100 Epochen trainiert und in Zeile 3 eine Vorhersage erzeugt.

3.3.2 Python

In *Python* wird das Modell in Algorithmus 3.2 deklariert und trainiert. Dabei wird im Block ab Zeile 4 ein NN mit zwei Layer (50 & 20 Neuronen) erstellt, das zwei Eingabeparameter (*input_dim* = 2) und einen Ausgabeparameter (Zeile 8) hat. Zusätzlich kann hier die Aktivierungsfunktion (siehe Kapitel 3.1.3) für jedes Neuron angeben werden, wobei hier die ReLU-Funktion übergeben wird. Soll die Ausgabe zwischen 0 und 1 liegen, bietet es sich an, die Sigmoid-Funktion als Aktivierung zu verwenden. Durch Runden des Ausgabewertes kann ein NN zwischen zwei Ereignissen wie z.B. Signal oder Untergrund unterscheiden. In Zeile 12 wird das Modell in 100 Epochen trainiert. *x_train* ist hierbei eine Liste für die Trainingsdaten, mit zwei Spalten für die zwei Eingabeparameter. Die Ausgabe steht in *y_train*, das nur eine Spalte hat.

Schließlich wird die Vorhersage *y_pred* in Zeile 15 mit den Testdaten *x_test* berechnet.

```

1 from keras.models import Sequential
2 from keras.layers import Dense
3
4 # Neuronales Netz definieren
5 model = Sequential()
6 model.add(Dense(50, activation="relu", input_dim=2)
7 model.add(Dense(20, activation="relu"))
8 model.add(Dense(1))

```

⁷Ein *Tree* bildet in *ROOT* das Datensystem, in dem einzelne Ereignisse einer Reaktion gespeichert werden.

```

9 model.compile(loss="mse", optimizer="adam")
10
11 # Trainieren
12 model.fit(x_train, y_train, epochs=100)
13
14 # Vorhersagen
15 y_pred = model.predict(x_test)

```

Algorithmus 3.2: Das Modell mit 50 Neuronen im ersten Layer und 20 Neuronen im zweiten Layer wird in Zeile 5 bis 9 deklariert. In Zeile 12 wird es mit 100 Epochen trainiert und in Zeile 15 eine Vorhersage berechnet.

Anders als bei *ROOT* müssen die Rohdaten hier manuell auf Trainings- und Testdaten geteilt und normiert werden. Dies geschieht über Algorithmus 3.3 mithilfe des Pakets *sklearn* [15]. In Zeile 5 werden die Daten zu 50% geteilt und durch das Argument *shuffle = True* auch gemischt. Danach wird mit *StandardScaler* (Zeile 8) die Normalisierung der Werte vorgenommen.

```

1 from sklearn.model_selection import train_test_split
2 from sklearn.preprocessing import StandardScaler
3
4 # Aufteilen der Daten
5 x_train, x_test, y_train, y_test = train_test_split(x_data, y_data,
6     test_size=0.5, shuffle=True)
7
8 # Normalisieren der Daten
9 scaler = StandardScaler()
10 scaler.fit(x_data)
11 x_train_norm = scaler.transform(x_train)
12 x_test_norm = scaler.transform(x_test)

```

Algorithmus 3.3: Aufteilen der Rohdaten in Test- und Trainings-Pakete in Zeile 5. Normalisierung der Daten in Zeile 8 bis 11.

4 Bilderkennung mithilfe Neuronaler Netze

Als erstes Anwendungsbeispiel wird ein NN auf die Erkennung der Zahlen 0 bis 9 trainiert. Es soll veranschaulicht werden, wie gut das Netzwerk zwischen Bildern unterscheiden kann und welche Schwierigkeiten dabei auftreten.

4.1 Datenset

Die Daten für das Trainieren stammen aus Quelle [17]. Hierbei handelt es sich um handgeschriebene Ziffern von 0 bis 9, die in Graustufen mit 8×8 Pixeln aufgelöst werden. Damit ergeben sich für das NN 64 Eingabeparameter pro Bild. Mit farbcodierten Bildern, wären es dreimal so viele Werte, das die Erkennung sofort aufwendiger gestaltet. In Abbildung 4.1 sind zehn von insgesamt 1797 Ziffern dargestellt.

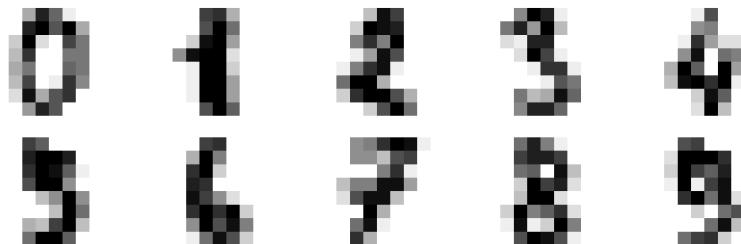


Abbildung 4.1: Das Datenset mit handgeschriebenen Beispielziffern von 0 bis 9.

4.2 Vorhersage mit Python

Es wird nun ein NN in *Python* erstellt (siehe Kapitel 3.3.2), das aus zwei versteckten Layer mit verschiedener Anzahl von Neuronen und einem Ausgabe-Layer mit zehn Neuronen besteht. Es werden 70 % aller Daten zum Trainieren und 30 % zum Testen verwendet.

Im Algorithmus 4.1 wird das Modell im ersten Block definiert und im zweiten trainiert. Der Zusatz *EarlyStopping* in Zeile 9 verhindert ein Überfitten des NN und bricht das Trainieren vor dem Erreichen der 100 Epochen ab, wenn zehn Epochen lang keine Verbesserung von 10^{-6} des *val_loss*-Wertes erreicht wird. Dieser charakterisiert den Fehler bzgl. des Sollwerts vom Netzwerk und nimmt beim Trainieren im besten Falle ab⁸. Des Weiteren wird dieser Fehler nur aus Daten berechnet, mit denen nicht trainiert wird. Diese werden in jeder Epoche zufällig aus den Trainingsdaten gewählt, wobei *validation_split* in Zeile 10 den Anteil angibt [16].

Die Form der übergebenden Daten *x_train* und *y_train* müssen der Anzahl der Neuronen im Netzwerk entsprechen. Da zehn Ausgabeneuronen definiert wurden, haben die Ausgabedaten pro Bild zehn Werte. Dabei werden alle Werte auf 0 gesetzt und der Index, der der

⁸Kann das NN keine Korrelation zwischen Eingabe und Ausgabe-Parameter herstellen, wird auch der Loss-Wert nicht besser.

Ziffer entspricht, wird mit 1 markiert. Wird ein NN dieser Art verwendet, sollte die *Loss* Funktion angepasst werden, wie in Zeile 6 zu sehen ist.

Im letzten Block werden mithilfe der Testdaten x_{test} die Ziffern vorhergesagt. Als Vorhersage kann mit $model.predict(x_{test})$ eine Liste mit zehn Werten für jedes Bild generiert werden. Der Index mit dem größten Wert entspricht der vorhergesagten Ziffer. Es bietet sich vorher allerdings an, die sog. *Softmax* Funktion (Zeile 13) anzuwenden, die die Wahrscheinlichkeit einer Vorhersage bestimmt. Mathematisch sieht die Softmax-Funktion [11], wie in Gleichung (4.1) gezeigt, aus.

$$\varphi_{\text{Softmax}}(y_i) = \frac{e^{y_i}}{\sum_k e^{y_k}} \quad (4.1)$$

Dabei beschreibt y den Ausgabewert und i dessen Index. Bei der Summe über alle Indizes k wird über den kompletten Ausgabe-Layer iteriert und in Zeile 14 eine Liste für jedes Bild generiert, die angibt, wie wahrscheinlich eine bestimmte Ziffer ist. In Abbildung 4.2 ist das Wirken der *Softmax* Funktion auf einen Ausgabe-Layer mit fünf Neuronen dargestellt.

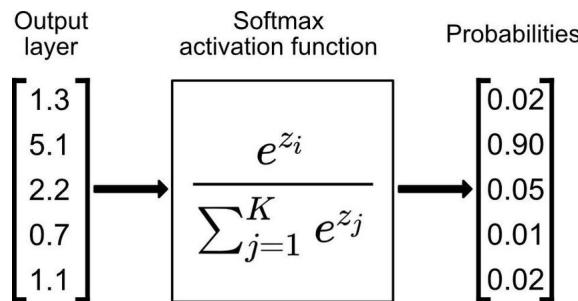


Abbildung 4.2: Die Softmax-Funktion auf einen Ausgabe-Layer mit fünf Neuronen angewendet. Es werden die Wahrscheinlichkeiten ausgegeben. Grafik aus [11].

```

1 # Neuronales Netz definieren
2 model = Sequential()
3 model.add(Dense(100, activation="relu", input_dim=64)
4 model.add(Dense(50, activation="relu"))
5 model.add(Dense(10))
6 model.compile(optimizer="adam", loss=tf.keras.losses.
    SparseCategoricalCrossentropy(from_logits=True))
7
8 # Trainieren
9 es = EarlyStopping(monitor="val_loss", mode="min", min_delta=1e-6,
    patience=10)
10 model.fit(x_train, y_train, epochs=100, validation_split = 0.1, shuffle
    =True, callbacks=[es])
11
12 # Vorhersagen
13 prob_model = Sequential([model, Softmax()])
14 predicted = prob_model.predict(x_test)

```

Algorithmus 4.1: NN zur Ziffererkennung von 0 bis 9. Es besteht aus zwei Layer mit 100 und 50 Neuronen und aus Ausgabe-Layer mit zehn Neuronen. Trainiert wird das Modell in 100 Epochen. Für die Vorhersage wird außerdem die *Softmax*-Funktion verwendet.

4 Bilderkennung mithilfe Neuronaler Netze

Mit der Vorhersage vom NN lässt sich daraufhin die Qualität des Modells analysieren. Zunächst werden beispielhaft ein paar Ziffern der Testdaten mit der dazugehörigen, wahrscheinlichsten Zahl betrachtet. In Abbildung 4.3 sind fünf Ziffern abgebildet, die das NN korrekt vorhergesagt hat. Diese Zahlen sind anders als in Abbildung 4.4 eindeutig zu erkennen. Dort sind Ziffern gezeigt, die falsch berechnet wurden. Die Zahlen sind teilweise uneindeutig. Das Ziffer ganz links im Bild kann z.B. auch als eine 9 anstatt einer 4, was korrekt wäre, erkannt werden. Die übrigen Zahlen sind jedoch klar zu erkennen.

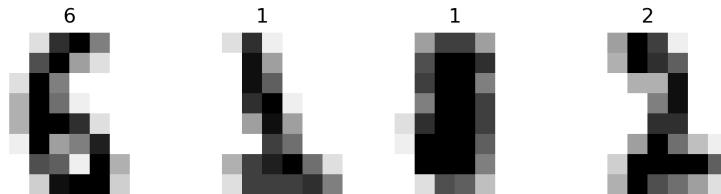


Abbildung 4.3: Vom NN richtig vorhergesagte Ziffern. Oben die Vorhersage und unten das übergebende Bild.

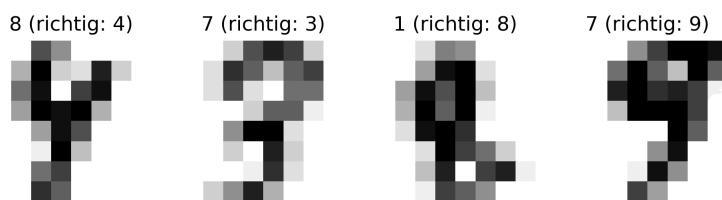


Abbildung 4.4: Vom NN falsch vorhergesagte Ziffern. Oben die Vorhersage mit richtiger Zahl und unten das übergebende Bild. Die Ziffer ganz links ist uneindeutig.

Das NN leistet also gute Arbeit bei einfachen Bildern, scheitert aber bei etwas komplexeren Aufgaben, die für einen Menschen deutlich sind. Besteünde das Datenset aus mehr Bildern, mit denen das Netzwerk trainieren könnte, würden solche Bilder womöglich besser erkannt werden.

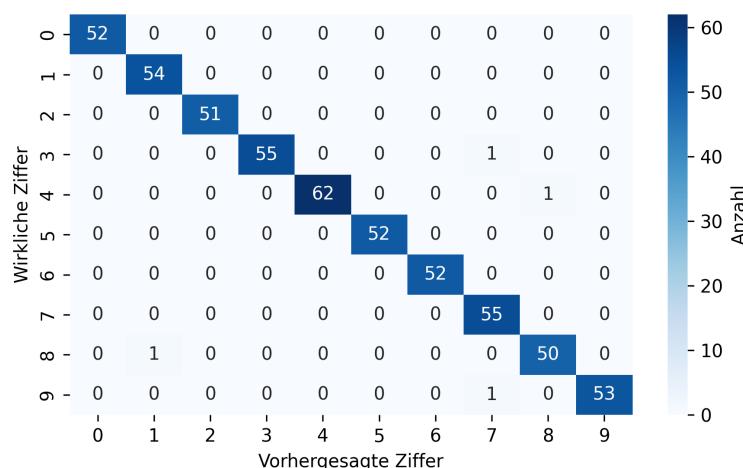


Abbildung 4.5: Konfusionsmatrix mit der Anzahl aller wirklichen gegen die vorhergesagten Ziffern dargestellt. Die dominante Linie auf der Diagonalen spricht für die hohe Genauigkeit des NN. Diese liegt bei 99,3 % für 540 Testdaten.

Diese Erkenntnisse beschränken sich jedoch nur auf insgesamt acht von fast 2 000 Bildern

und sind nicht aussagekräftig für das Datenset. Es werden nun die Testdaten insgesamt betrachtet und dafür eine sogenannte Konfusionsmatrix benutzt, die in einer Dimension die tatsächliche, korrekte Ziffer und in der anderen Dimension die vom NN vorhergesagte Ziffer anzeigt. Die Einträge sind histogrammgemäß aufsummiert. Würden alle Daten korrekt vorhergesagt werden, wäre in der Matrix eine ausgezeichnete Diagonale erkennbar. Dieser Fall ist in Abbildung 4.5 zu erkennen, in der alle Test-Ziffern aufgezählt werden. Es sind nur vier Einträge entfernt von der Diagonalen zu finden, das die hohe Genauigkeit des Modells beweist. Es werden 99,3 % aller Bilder korrekt erkannt.

4.3 Vergleich zu Root

Um die selben Daten mit dem NN in *Root* zu verwenden, muss hier zunächst ein *Root Tree* erzeugt werden. Die genauen Prozeduren sind für die/den Root-Anwender:in trivial und werden daher nicht beschrieben. Durch das Definieren und Trainieren des NN wie in Abschnitt 3.3.1 gezeigt, kann auch hier berechnet werden, wie viel Prozent der Bilder das Modell richtig vorhersagt. Um einen geeigneten Vergleich zu schaffen, werden beide Systeme mit einer verschiedenen Anzahl von Neuronen (20 bis 100) in zwei versteckten Layer trainiert und die Laufzeit für das Trainieren und die Genauigkeit festgehalten. Das sind die zwei wichtigsten Werte für die weitere Analyse. Würden sich diese Parameter in den zwei Systemen gegenseitig ausschließen, müsste überlegt werden, was für die Analyse wichtiger ist.

In Abbildung 4.6 sind diese Werte dargestellt. Es ist eindeutig zu erkennen, dass *Python* höhere Genauigkeiten und niedrigere Laufzeiten als *ROOT* aufweist. Die Genauigkeit liegt für alle *Python*-Daten zwischen 95 % und 99 %, wobei *ROOT* maximale Genauigkeiten von 98 % erreicht. Zusätzlich überschreitet die Laufzeit der *Python* NN nicht die 40 s Marke. *ROOT* hingegen erreicht Laufzeiten von 120 s.

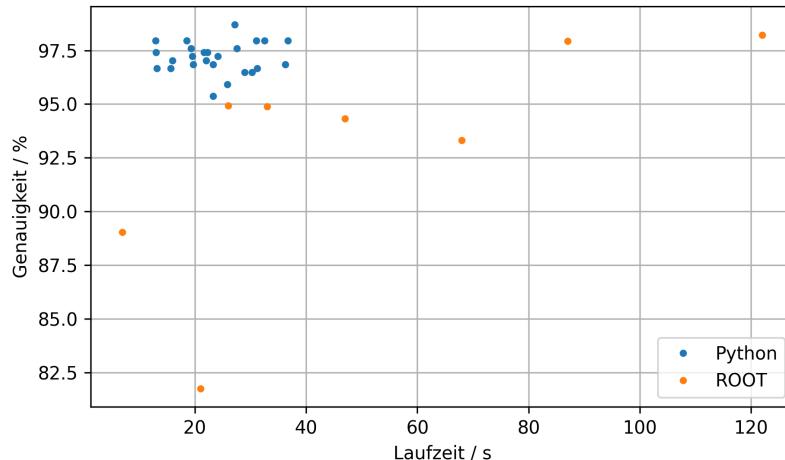


Abbildung 4.6: Vergleich von Modelle in *Root* und *Python*. Aufgetragen ist die Genauigkeit der Vorhersage und die Laufzeit für das Trainieren des NN für verschiedene Anzahl von Neuronen. *Python* überwiegt dabei stark in beiden Punkten.

Für die Untergrundminimierung von $\gamma p \rightarrow p\eta'$ wird somit das *Tensorflow*-Modul in *Python* verwendet. Für spätere Datengrößen von einigen Millionen Einträgen ist die Laufzeit ein wichtiger Faktor. Ebenso muss die Genauigkeit einen gewissen Wert erreichen, damit diese Form der Analyse Erfolg hat.

5 Analyse von $\gamma p \rightarrow p\eta'$

In diesem Kapitel soll mithilfe des NN die Reaktion $\gamma p \rightarrow p\eta'$ extrahiert werden. Dazu wird zunächst betrachtet, welche anderen Teilchen entstehen und die gewollte Reaktion überlagern. Diese Teilchen werden im Folgenden stets als Untergrund bezeichnet, da sie das η' -Signal unterdrücken.

Um das NN zu trainieren, werden Simulation für die oben genannte Reaktion angefertigt, die dem Modell beschreiben, bei welchen Daten ein η' oder ein Untergrund-Ereignis auftritt. Entspricht die Qualität dieser Simulationen der Realität, sollte das NN die genannte Reaktion von allen anderen unterscheiden können.

5.1 Reduktion des Untergrundes mithilfe Cuts

Wie in Abschnitt 2.3 bereits beschrieben, gibt es verschiedene Teilchen, die bei der Reaktion entstehen können. Um das η' in den Daten besser hervorzuheben, werden in der Analyse von J. KRAUSE [9] verschiedene Cuts⁹ angewendet, die Untergrund unterdrücken sollen:

- Ladungs Cut
- Zeit Cut
- θ Cut
- ϕ Cut
- Fehlende Masse Cut

In Abbildung 5.1 ist die invariante Masse der zwei gemessenen Photonen dargestellt. Diese Masse beschreibt die in Reaktion zur Verfügung stehende Energie im Schwerpunktssystem und somit die Ruhemasse der erzeugten Mesonen [18]. Außerdem sind die Messdaten gezeigt, die nach den verschiedenen Cuts verbleiben.

An dieser Stelle sei noch einmal erwähnt, dass die entstehenden Mesonen auch in mehr als zwei Photonen zerfallen können. Der Kanal mit genau zwei γ tritt aber dennoch auf, wenn diese Teilchen nicht gemessen oder falsch identifiziert werden. Die fehlende Photonenenergie sollte theoretisch in der Analyse auffallen, jedoch stellt es sich heraus, dass diese Energie so klein ist, dass sie für die gesamte invariante Masse vernachlässigbar ist und nicht für einen weiteren Cut verwendet werden kann [9].

Nach dem Ladungs Cut (blau) ist nur der π^0 - und η -Peak zu erkennen. Der gleichmäßige Untergrund besteht hierbei aus $2\pi^0$ und $\pi^0\eta$ Reaktionen, auf die später noch eingegangen wird. Durch die weiteren Cuts wird dieser Untergrund eliminiert und nach dem Fehlende Masse Cut sind die vier Teilchen π^0 , η , ω und η' deutlich zu erkennen. Die vertikalen Linien zeigen einen 2σ Bereich um den η' Peak bei der Schwerpunktsenergie von 957,78 MeV [4]. π^0 dominiert dieses Spektrum und könnte an dieser Stelle extrahiert werden. η' Ereignisse sind dagegen fast drei Größenordnungen seltener.

An dieser Stelle soll das NN eingesetzt werden, um das Signal-Rausch-Verhältnis weiter zu verbessern.

⁹englisch für Schnitte. Dadurch bleiben nur Ereignisse übrig, deren Parameter in einem angegebenen Bereich liegen.

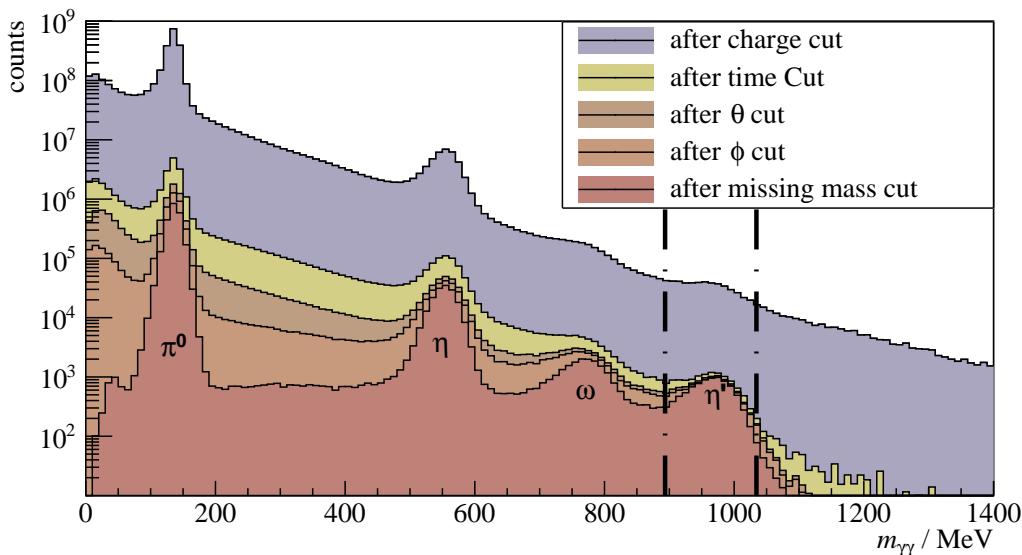


Abbildung 5.1: Invariante Masse Spektrum der Reaktion nach verschiedenen Cuts: Ladungs-, Zeit, θ , ϕ und Fehlende Masse Cut. Die rote Kontur lässt die Teilchen π^0 , η , ω und η' sehr deutlich werden, während der blaue Abschnitt viel Untergrund besitzt. Die vertikalen Linien zeigen einen 2σ Bereich um den η' Peak für einen letzten invarianten Masse Cut. Grafik aus [9].

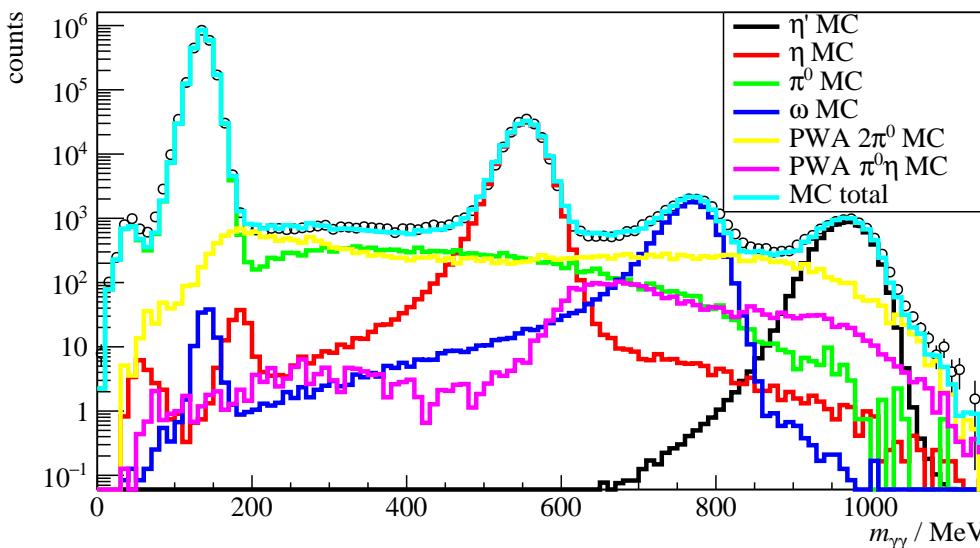


Abbildung 5.2: Invariante Masse der Monte Carlo Simulationen nach allen Cuts: η' in schwarz, η in rot, π^0 in grün, ω in blau, $2\pi^0$ in gelb und $\pi^0\eta$ in pink. Die Summe aller Histogramme ist in türkis dargestellt. Die schwarzen Kreise repräsentieren die gemessenen Daten. Grafik aus [9].

Als Trainingsdaten werden Monte Carlo (MC) Simulationen verwendet, die von F. AFZAL [1] erstellt und von J. KRAUSE [9] bereitgestellt wurden. Bei diesen Simulationen, die speziell für das Crystal Barrel Experiment angefertigt sind, werden die Reaktionen soweit rekonstruiert, dass die Viererimpulse aller gemessener Teilchen bekannt sind.

In Abbildung 5.2 ist die invariante Masse der MC Simulationen der sechs wichtigsten Teilchen nach allen oben genannten Cuts dargestellt. Hier wird auch ersichtlich, wie viel Un-

5 Analyse von $\gamma p \rightarrow p\eta'$

tergrund im η' Peak (schwarze Linie) vorhanden ist. $2\pi^0$ und $\pi^0\eta$ (gelbe und pinke Linie) sind dabei am wichtigsten. Insgesamt gleicht die Summe der Simulationen (türkise Linie) den Messdaten (schwarze Kreise).

Für das Erstellen des NN ist es notwendig, die Verhältnisse aller auftretenden Reaktionen rekonstruieren zu können, um die realen Daten bestmöglich wieder zu geben. Dafür wird über die MC Histogramme in Abbildung 5.2 in einem Bereich der Strahlenenergie von 1400 MeV bis 2800 MeV integriert und die Ergebnisse auf die Gesamtanzahl aller Ereignisse normiert. Die Ergebnisse sind in Tabelle 5.1 angegeben.

| Reaktion | Anteil |
|---|-----------|
| $\gamma p \rightarrow p\eta' \rightarrow p\gamma\gamma$ | 0,003 104 |
| $\gamma p \rightarrow p\eta \rightarrow p\gamma\gamma$ | 0,079 251 |
| $\gamma p \rightarrow p\pi^0 \rightarrow p\gamma\gamma$ | 0,896 278 |
| $\gamma p \rightarrow p\omega \rightarrow p3\gamma$ | 0,006 088 |
| $\gamma p \rightarrow p2\pi^0 \rightarrow p4\gamma$ | 0,012 620 |
| $\gamma p \rightarrow p\pi^0\eta \rightarrow p4\gamma$ | 0,002 660 |

Tabelle 5.1: Anteile einzelner Teilchenreaktionen an dem Gesamtspektrum im Energiebereich von 1400 MeV bis 2800 MeV. Werte bereitgestellt von J. KRAUSE [9].

Erste Tests haben gezeigt, dass die Effizienz des Netzwerkes verbessert werden kann, wenn die Teilchen gewichtet sind. Diese Erkenntnis ist für die Verwendung der Simulationen in Kapitel 5.3 von Bedeutung.

5.2 Erste Entwicklung des Neuronalen Netzes

Es soll nun schrittweise das NN aufgesetzt werden, um die η' Ereignisse extrahieren zu können.

5.2.1 Eingabeparameter

Für das Trainieren werden die Daten vorbereitet, damit das NN sie verarbeiten kann. Alle MC Simulationen werden von *ROOT Trees* in Textdateien umgewandelt, damit sie in *Python* eingelesen werden können. Jede Zeile beschreibt dann ausgewählte Parameter eines einzelnen Ereignisses. Also letztendlich die Eigenschaften der Teilchenreaktion. Generell gilt: Je mehr Eingabeparameter das NN erhält, desto besser kann ein Ereignis vorhergesagt werden. Vor der Analyse ist nicht klar, welche Korrelationen bestehen. Sollte sich herausstellen, dass bestimmte Parameter keinen Zusammenhang zur Ausgabe besitzen, können sie wieder entfernt werden.

Die zur Verfügung stehenden Parameter sind:

- Strahlenenergie E_γ
- Bewegungsrichtung des Mesons $\cos \theta$
- Energie E_1, E_2, E_p
- Polarwinkel $\theta_1, \theta_2, \theta_p$
- Azimutwinkel ϕ_1, ϕ_2, ϕ_p
- Clustersize CS_1, CS_2, CS_p
- Clustercount CC_1, CC_2, CC_p

Dabei beschreiben die Indizes 1 und 2 die beiden Photonen und p das Proton. E_γ steht für die Strahlenenergie des einlaufenden Photons und $\cos \theta$ beschreibt die Bewegungsrichtung des Mesons im Zwischenzustand. E_i ist die Energie, θ_i der Polarwinkel und ϕ_i der Azimutwinkel der Teilchen im Endzustand. CC steht für Cluster Count und gibt die Anzahl von Detektorkristallen an, die bei der Detektierung eines Teilchen Energie aufgenommen haben. Dieser Effekt basiert auf Paarbildung vor dem Kristall, wodurch elektromagnetische Schauer entstehen und die Energie des Teilchens aufgeteilt wird [1]. CS steht für Cluster Size und soll die Verteilung des Teilchenschauers wiedergeben.

5.2.2 Zweiteilchenzerfall

Als erster Test des NN werden zunächst zwei Teilchen betrachtet: η' und der Hauptuntergrund $2\pi^0$. Dieser Test dient dafür, ob allein durch die Parameter eines Ereignisses in Abschnitt 5.2.1 zwischen zwei Teilchen unterschieden werden kann.

In Abbildung 5.3 ist die invariante Masse der beiden Teilchen in einem Histogramm in blau und orange dargestellt. Hier werden alle zur Verfügung stehenden Daten mit 100 000 $2\pi^0$ und 100 000 η' Ereignissen untersucht. Dieses Verhältnis ist keineswegs realitätsgerecht und dient nur zu Testzwecken, ob das Netzwerk anhand der übergebenden Parametern zwischen Mesonen unterscheiden kann. Die geringe Anzahl an Daten ermöglicht außerdem schnellere Tests.

Nachdem die Daten, wie im Programmcode 3.3 bereits gezeigt, normalisiert und in Trainings- und Testdaten aufgeteilt wurden, kann ein NN aufgesetzt und trainiert werden. In Algorithmus 5.1 wird ein Modell mit 100 Neuronen im ersten und 50 Neuronen im zweiten Layer erstellt, das in 100 Epochen trainiert wird. Diese Werte wurden willkürlich gewählt. In Abschnitt 5.2.3 erfolgt eine genauere Diskussion.

```

1 # Erstellen vom NN
2 model = Sequential()
3 model.add(Dense(100, activation="relu", input_dim=len(x_train[0])))
4 model.add(Dense(50, activation="relu"))
5 model.add(Dense(1, activation="sigmoid"))
6 model.compile(loss="MeanSquaredError", optimizer="Adam")
7
8 # Trainieren
9 es = EarlyStopping(monitor="val_loss", mode="min", min_delta=1e-6,
10 patience=10)
11 model.fit(x_train, y_train, epochs=100, validation_split = 0.1, shuffle
12 =True, callbacks=[es])
13
14 # Vorhersagen
15 y_pred = np.round(model.predict(x_test)).astype(int)

```

Algorithmus 5.1: Aufsetzen eines NN zur η' und $2\pi^0$ Erkennung.

Als Ausgabelayer wird ein Neuron mit der Sigmoid Aktivierungsfunktion gewählt, sodass Werte zwischen 0 und 1 als Ausgabe erzeugt werden. Daher wird vor dem Trainieren eine Ausgabedatei für das NN erstellt, in der alle η' mit „1“ und alle $2\pi^0$ Ereignisse mit „0“ gekennzeichnet werden. Dadurch entsteht ein einfaches Klassifizierungsproblem für das Netzwerk. Um eine Vorhersage zu erhalten, wird die Ausgabe `model.predict()` in Zeile 13 gerundet, sodass Werte $\geq 0,5$ als η' zählen und alle anderen als $2\pi^0$.

In Abbildung 5.3 ist gezeigt, welche Ereignisse das NN als η' (grün) und $2\pi^0$ (rot) vorhersagt. Dargestellt ist die invariante Masse der zwei Photonen. Die Vorhersage deckt sich mit

5 Analyse von $\gamma p \rightarrow p \eta'$

den Kontrollwerten (blau und orange) und es wird eine Gesamtgenauigkeit von 93,25 % erreicht. In dem Bereich des η' Peaks werden weniger $2\pi^0$ Ereignisse erkannt und der Graph weicht nach unten hin ab. Dieses Phänomen liegt an den geringen Datengrößen und wird im nächsten Kapitel 5.3 kein Problem mehr darstellen, da dort fast 8 Millionen Ereignisse betrachtet werden.

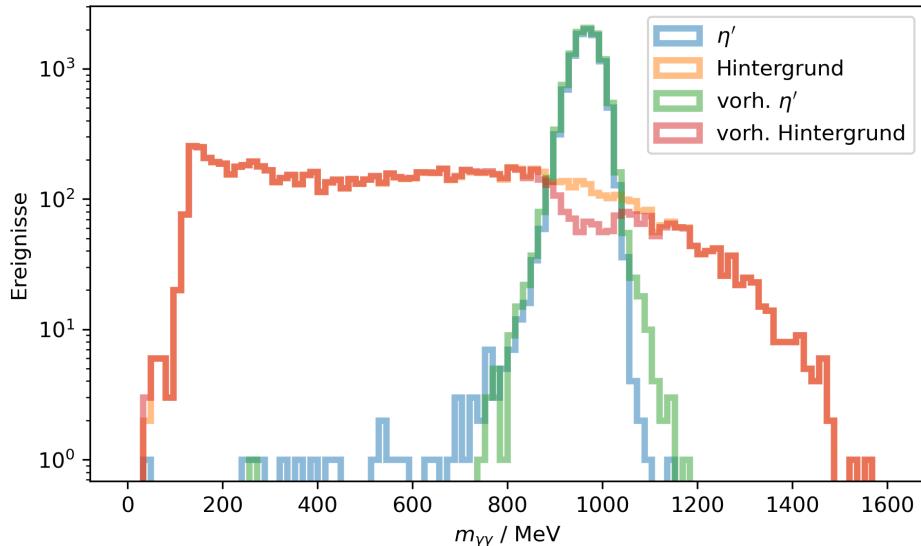


Abbildung 5.3: Invariante Masse der vorhergesagten η' (grün) und $2\pi^0$ (rot) Ereignisse vom NN. Zusätzlich sind Kontrolldaten der Ereignisse in blau und orange dargestellt. Die Gesamtgenauigkeit liegt bei 93,25 %.

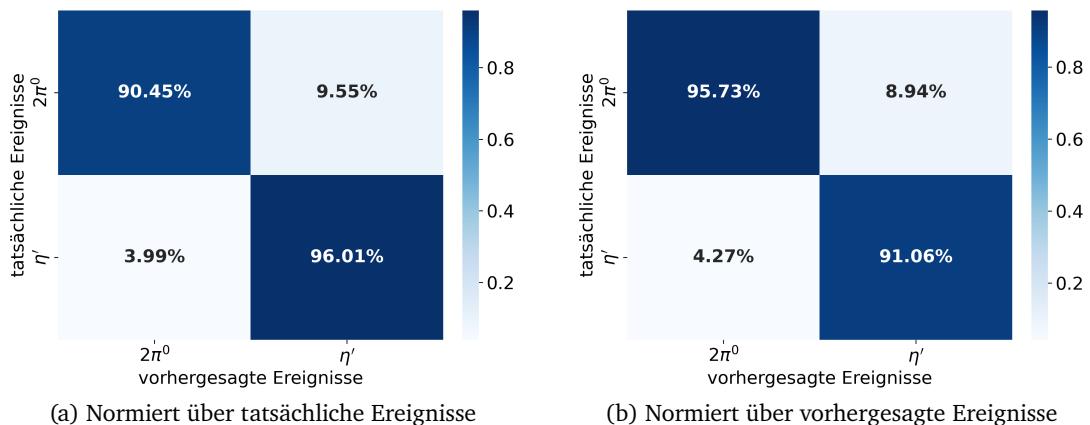


Abbildung 5.4: Konfusionsmatrizen für die η' oder $2\pi^0$ Reaktionen. Normiert sind die Matrizen zeilenweise über die tatsächlichen/wirklichen Ereignisse (a) und spaltenweise über die vorhergesagten Ereignisse (b) vom NN.

Von allen tatsächlichen η' Ereignissen werden 96,01 % als solche erkannt und von allen vorhergesagten η' Ereignissen sind 91,06 % tatsächlich solche. Diese Werte lassen sich anschaulich in einer Konfusionsmatrix 5.4 zusammenfassen. Hierbei werden auf der einen Achse alle Ereignisse aufgezählt, die tatsächlich aufgetreten sind, und auf der anderen Achse diejenigen Mesonen, die vom NN vorhersagt werden. Außerdem sind die Zahlen über alle wirklichen η' oder $2\pi^0$ Ereignisse (Abb. 5.4 a) oder über alle vom NN vorhergesagten Reaktionen (Abb. 5.4 b) normiert, sodass die Werte reihen- bzw. spaltenweise 100 % ergeben. Es sollten immer beide Normierungen betrachtet werden, um falsche Schlüsse zu

vermeiden. Es kann sein, dass der eine Wert gegen 1 läuft, was für ein gut trainiertes NN spricht. Die andere Genauigkeit könnte aber drastisch niedriger sein. Die Effizienzen für $2\pi^0$ sind ähnlich gut. Mit 90,45 % und 95,73 % wird auch dieses Ereignis erkannt. Zusammengefasst ist dem NN also möglich zwischen den zwei Teilchen zu unterscheiden. Mit der Erkenntnis wird sich nun der kompletten Reaktion gewidmet.

5.2.3 Feinjustierung

Nachdem bekannt ist, wie das NN erstellt und angewendet wird, sollen nun die Feinjustierungen am Netzwerk vorgenommen werden. Dieser Schritt ist essentiell, da verschiedene Einstellungen die Genauigkeit verändern können. Zu diesen Parameter gehören:

- Anteil Testdaten von Gesamtdaten
- Loss Funktionen
- Optimizer Funktionen

Bei der Analyse der Einstellungen muss angenommen werden, dass alle Parameter unabhängig voneinander verstellbar sind. Das heißt für verschiedene Loss Funktionen wird z.B. immer genau eine Optimizer Funktion am besten sein. Das Gegenteil würde bedeuten, dass alle Kombinationen dieser Parameter auf die Genauigkeit des NN getestet werden müssten, was einen zeitlichen Rahmen überschreitet. Durch diese Annahme reicht es, alle Einstellungen einzeln zu testen. Das bedeutet aber auch, dass sich die absoluten Genauigkeiten über verschiedene Parameter hinweg unterscheiden können. Wichtig sind nur die Maxima in einer einzelnen Analyse.

Es werden wieder die η' und $2\pi^0$ Ereignisse aus Abschnitt 5.2.2 mit der gleichen Verteilung verwendet. Außerdem wird für jede Einstellung das Netzwerk drei Mal trainiert, um einen Mittelwert und eine Standardabweichung aus dem Ergebnis zu erhalten. Das soll die zufällige Auswahl der Testdaten und daher auftauchenden Unsicherheiten abschätzbar machen.

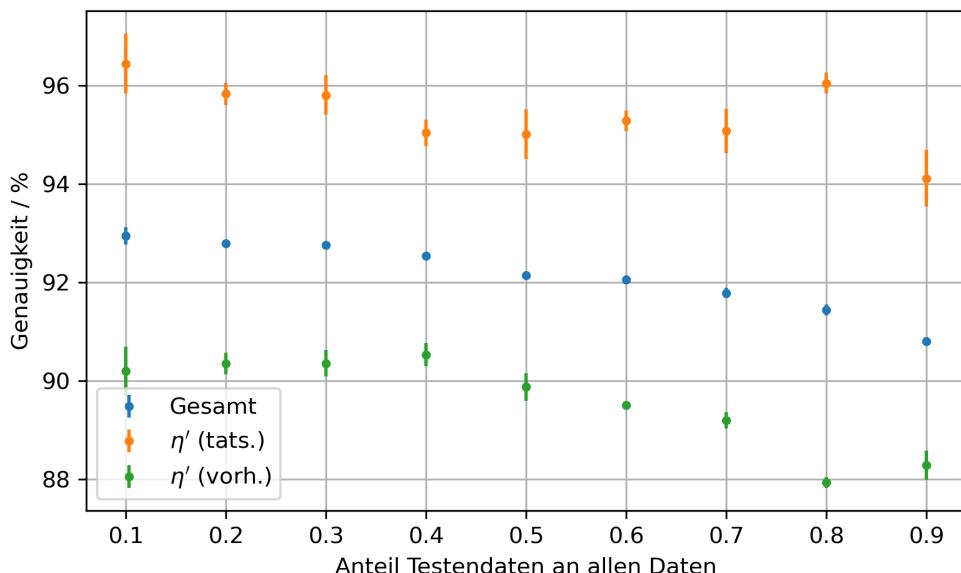


Abbildung 5.5: Genauigkeit des NN gegen verschiedene Anteile der Testdaten von den Gesamtdaten. Ab 0,3 fällt die Genauigkeit ab.

Anteil Testdaten von Gesamtdaten

In der ersten Analyse sollen die Unterschiede ermittelt werden, wenn von allen Daten verschiedene Anteile für Testdaten und Trainingsdaten verwendet werden. Das beste Verhältnis sollte gefunden werden, da mehr Trainingsdaten für potentiell höhere Genauigkeiten sorgen, während diese Genauigkeit letztendlich durch die Testdaten bestimmt wird, die den Datensatz repräsentieren sollen. Zu wenig Testdaten sprächen also für eine geringe Aussagekraft.

Im Abbildung 5.5 ist die Genauigkeit des NN gegen verschiedene Testdaten-Anteile dargestellt. Dabei ist die Genauigkeit aller und η' Ereignisse wichtig. Die letztere wird auf alle tatsächlichen und alle vorhergesagten η' Reaktionen normiert (vgl. Abb. 5.4). Für mehr Testdaten werden die Trainingsdaten weniger und die Genauigkeit nimmt daher ab. Um dennoch eine hohe Aussagekraft zu erhalten, wird sich für die spätere Analyse für den Anteil von 0,3 entschieden.

Loss-Funktionen

Durch die sog. Loss-Funktionen wird der Fehler einer Ausgabe zum Sollwert während des Trainierens festgelegt. Neben bekannten Funktionen wie z.B. der mittlere quadratische Fehler sind in der Dokumentation des Moduls noch weitere aufgelistet, die jedoch nicht weiter erklärt werden sollen. In Abbildung 5.6 wurden alle Funktionen auf Genauigkeit getestet, wobei erneut die Gesamt- und η' Genauigkeit überprüft wird. Alle Werte weichen nicht stark voneinander ab. Für einige Funktionen¹⁰ konnte das NN nicht trainiert werden, d.h. es wurde eine Gesamtgenauigkeit von genau oder leicht über 50 % erreicht. Am besten ist die LogCosh Funktion mit 92,94 % Gesamtgenauigkeit, welche von nun verwendet werden soll.

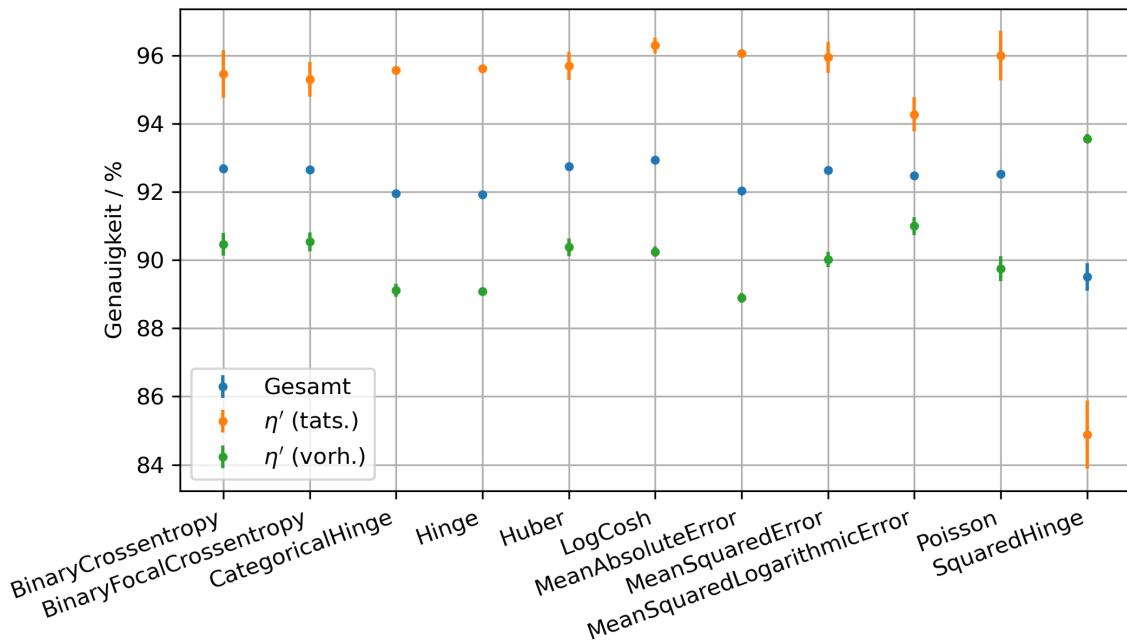


Abbildung 5.6: Gesamt- und η' Genauigkeiten des NN bei verschiedenen Loss-Funktionen. Am besten ist die LogCosh Funktion mit einer Gesamtgenauigkeit von 92,94 %.

Optimizer-Funktionen

In Abbildung 5.7 wurden verschiedene Optimizer-Funktionen verwendet, die die Korrektur von Gewichtungen und Schwellenwerte im NN nach jeder Epoche verursachen. Es wurden die Genauigkeiten bestimmt, wobei Adam mit einer Gesamtgenauigkeit von 92,83 % am besten abschließt.

¹⁰CategoricalCrossentropy, CosineSimilarity, KLDivergence, MeanAbsolutePercentageError

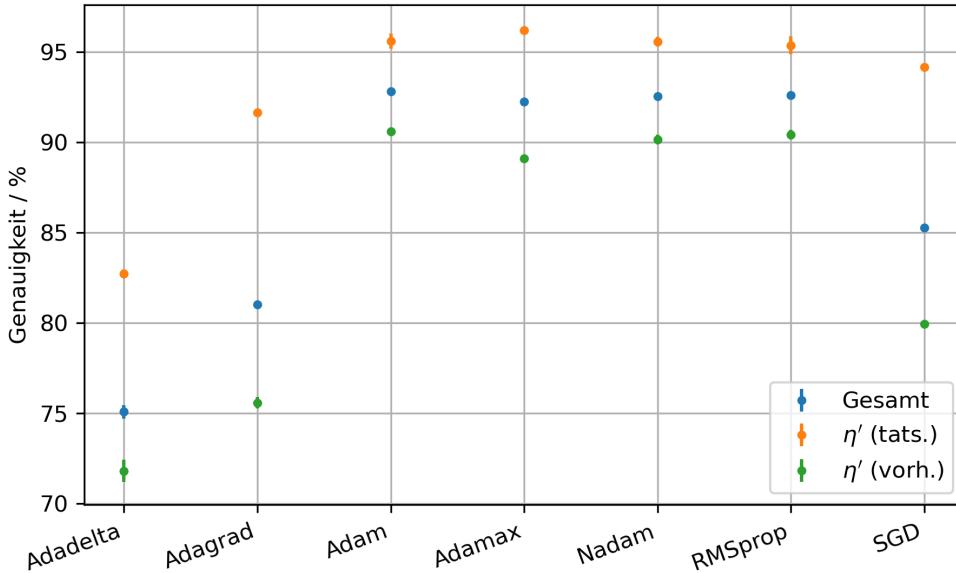


Abbildung 5.7: Gesamt und η' Genauigkeiten des NN bei verschiedenen Optimizer-Funktionen. Am besten ist die *Adam* Funktion mit einer Gesamtgenauigkeit von 92,83 %.

Zusammengefasst

Insgesamt werden also folgende Parameter für das NN ausgewählt:

- Anteil Testdaten von Gesamtdaten: 0,3
- Loss-Funktionen: *LogCosh*
- Optimizer: *Adam*

Der wichtigste, einstellbare Parameter für ein NN ist die Anzahl von Layer und Neuronen. Tests haben aber gezeigt, dass die Einstellung von den gewählten Daten abhängt und daher in diesem Abschnitt ungeeignet ist. Die Analyse wird nachgeholt, nachdem die MC Simulationen final gewichtet sind.

5.3 Finale Entwicklung des Neuronalen Netzes

Nachdem die η' und Untergrund Reaktionen diskutiert und ein erstes NN getestet und verbessert wurde, kann nur die finale Entwicklung eines Netzwerkes erfolgen, um die η' Ereignisse zu extrahieren.

5.3.1 Beachtung aller Teilchen

Im finalen Schritt der Entwicklung eines NN für die Teilchenerkennung und damit Untergrundminimierung der betrachteten Reaktion, werden alle beitragenden Teilchen hinzugefügt. In Tabelle 5.2 sind die Reaktionen aufgelistet mit den Anteilen, die in Abschnitt 5.1 bestimmt wurden. Außerdem ist die Anzahl zur Verfügung stehender Ereignisse jeder Reaktion angegeben. Da es sich bei diesen Daten weiterhin um Simulationen handelt, könnten für zukünftige Analysen auch mehr erstellt werden. In der Studie der NN stellte es sich als nicht zielführend heraus, das NN mit allen Daten zu trainieren und auf Testdaten

5 Analyse von $\gamma p \rightarrow p \eta'$

anzuwenden, die ein reales Verhältnis aufweisen. Dieser Schritt erschien sinnvoll, da so die gesamte Datenmenge¹¹ gebraucht werden konnte. Besser ist es, ebenso die Trainingsdaten mit den Anteilen aus Tabelle 5.2 zu gewichten. Der Nachteil wird ersichtlich, wenn ausgerechnet wird, wie viele Daten verwendet werden dürfen. Das beschreibt die letzte Spalte in der genannten Tabelle. Es kann nur noch ein Bruchteil aller η' Reaktionen analysiert werden. Ausschlaggebend dafür ist die geringe Menge an $\pi^0\eta$ Ereignisse, welche aber nicht vernachlässigt werden darf, da diese Reaktion den zweitgrößten Untergrund im η' Peak ausmacht, was in Abbildung 5.2 zu erkennen ist.

| Reaktion | Anteil | Ereignisse | verwendete Ereignisse |
|--|----------|------------|-----------------------|
| $\gamma p \rightarrow p \eta' \rightarrow p \gamma \gamma$ | 0,003104 | 13 792 166 | 23 833 |
| $\gamma p \rightarrow p \eta \rightarrow p \gamma \gamma$ | 0,079251 | 6 596 204 | 608 505 |
| $\gamma p \rightarrow p \pi^0 \rightarrow p \gamma \gamma$ | 0,896278 | 8 956 525 | 6 881 798 |
| $\gamma p \rightarrow p \omega \rightarrow p 3\gamma$ | 0,006088 | 983 755 | 46 745 |
| $\gamma p \rightarrow p 2\pi^0 \rightarrow p 4\gamma$ | 0,012620 | 103 675 | 96 899 |
| $\gamma p \rightarrow p \pi^0 \eta \rightarrow p 4\gamma$ | 0,002660 | 20 424 | 20 424 |

Tabelle 5.2: Ergänzung zu Tabelle 5.1 mit der Anzahl von allen zur Verfügung stehenden Ereignissen und die Anzahl, die für das Trainieren verwendet werden können, um das korrekte Verhältnis einzuhalten. Insgesamt können 7 678 204 Reaktionen verwendet werden.

In Abbildung 5.8 ist die invariante Masse aller Teilchen mit den Werten aus Tabelle 5.2 dargestellt und entspricht der Verteilung von Abbildung 5.2.

Die Aufgabe des NN wird es sein, zwischen η' und den anderen Daten, die als Hintergrund bezeichnet werden, zu unterscheiden. Ähnlich wie in Abschnitt 5.2.2 entspricht dann eine „1“ einer η' und eine „0“ einer Hintergrund Reaktion. Der restliche Aufbau gleicht dem Algorithmus 5.1.

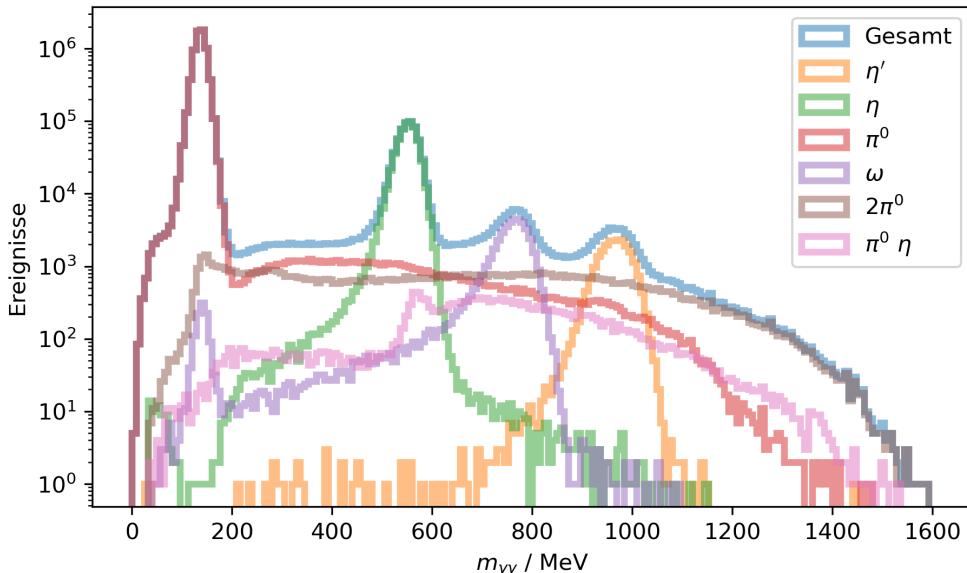


Abbildung 5.8: Invariante Masse für alle Teilchenkanäle mit dem korrekten Verhältnis. η' in orange, η in grün, π^0 in rot, ω in lila, $2\pi^0$ in braun und $\pi^0\eta$ in rosa. Die totale Verteilung ist in blau gezeigt.

¹¹Im diesen Fall entspricht das ca. 30 Millionen Ereignissen.

Es ist theoretisch auch möglich, das NN alle auftauchenden Teilchen voraussagen zu lassen. Passend zu Kapitel 4 wird ein Ausgabelayer mit sechs Neuronen erstellt, bei dem jedes für ein Teilchen steht. Dieses Verfahren wurde geprüft – jedoch mit großen Einbrüchen in der Genauigkeit. Da die Untergrundminimierung für die η' Reaktion erfolgen soll, ist die gewählte Methode die bessere Wahl.

5.3.2 Wahl von Layer und Neuronen

Um die Genauigkeit des NN letztendlich zu maximieren, muss die Anzahl von Layer und Neuronen festgelegt werden. Erste Analysen haben gezeigt, dass ein Netzwerk mit zwei Layer die beste Wahl ist, da bei einem Layer die Genauigkeit sehr gering ist und bei drei Layer teilweise keine η' vorhergesagt werden können. Diese Einschränkung sollte zunächst gemacht werden, da durch die große Datenmengen, die Laufzeit gestiegen ist. Für das Testen von verschiedenen Kombinationen an Neuronen wird also einige Zeit benötigt.

In Abbildung 5.9 ist gezeigt, wie verschiedene Modelle mit 40 bis 200 Neuronen in zwei Layer trainiert wurden. Die Gesamtgenauigkeiten in Abbildung 5.9a schwanken in einem sehr kleinen Bereich von nur 0,07 %. Das hat den Grund, da selbst eine Genauigkeit von $1 - \frac{23833}{7678204} = 99,69\%$ ¹² erreicht wird, wenn alle Ereignisse als Untergrund vorhergesagt werden. Die Genauigkeiten mit 160 und 200 Neuronen erreichen die niedrigsten Effizienzen. Eine mögliche Erklärung ist, dass die NN zu viele Neuronen haben und nicht trainiert werden können. Die restlichen Modelle liegen mit der Genauigkeit noch näher aneinander und haben eine Standardabweichung von ca. 0,005 %. Somit überlappen sich die Genauigkeiten mit den Fehlern teilweise. Der beste Wert liegt bei 160 Neuronen im ersten und 40 Neuronen im zweiten Layer mit $(99,848 \pm 0,004)\%$.

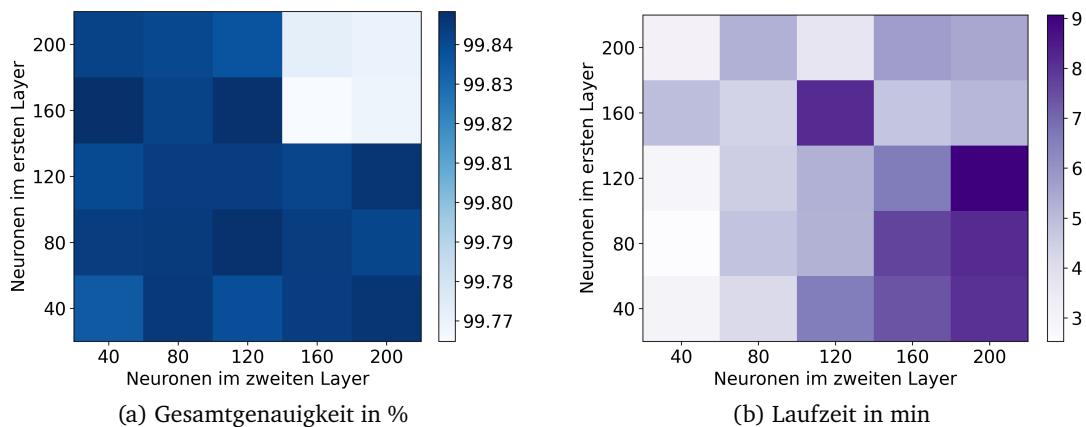


Abbildung 5.9: Gesamtgenauigkeiten (a) und Laufzeiten (b) für verschiedene Kombinationen von Neuronen in zwei Layer. Am besten sind 160 Neuronen im ersten und 40 Neuronen im zweiten Layer mit $(99,848 \pm 0,004)\%$ und $(5,0 \pm 0,6)$ min. Die NN oben rechts im Bild erreichen keine gute Genauigkeit, da das Modell zu groß ist.

Die Laufzeiten (Abb. 5.9b) dagegen erstrecken sich in einem Bereich von 3 min bis 9 min, wobei die Zeit für mehr Neuronen im zweiten Layer ersichtlich ansteigt. Die vier NN mit 160 und 200 Neuronen, haben eine geringe Laufzeit mit ca. 5 min, da sie nicht trainiert werden können.

Für die weitere Verwendung des Netzwerks werden Layer mit 160 und 40 Neuronen gewählt.

¹²Hier wurden die Werte aus Tabelle 5.2 verwendet.

5.3.3 Ergebnis

Das fertige NN mit allen Einstellungen wird auf die Testdaten angewendet und gibt die Vorhersage mit „0“ für Hintergrund und „1“ für η' Ereignisse aus. In Abbildung 5.10 wird ein Histogramm mit dieser Vorhersage (grün und rot) gefüllt und es ist eine Übereinstimmung mit den Kontrolldaten (blau und orange) zu erkennen. Anders als in Abbildung 5.3 wird hier auch der Untergrund im η' Peak erkannt.

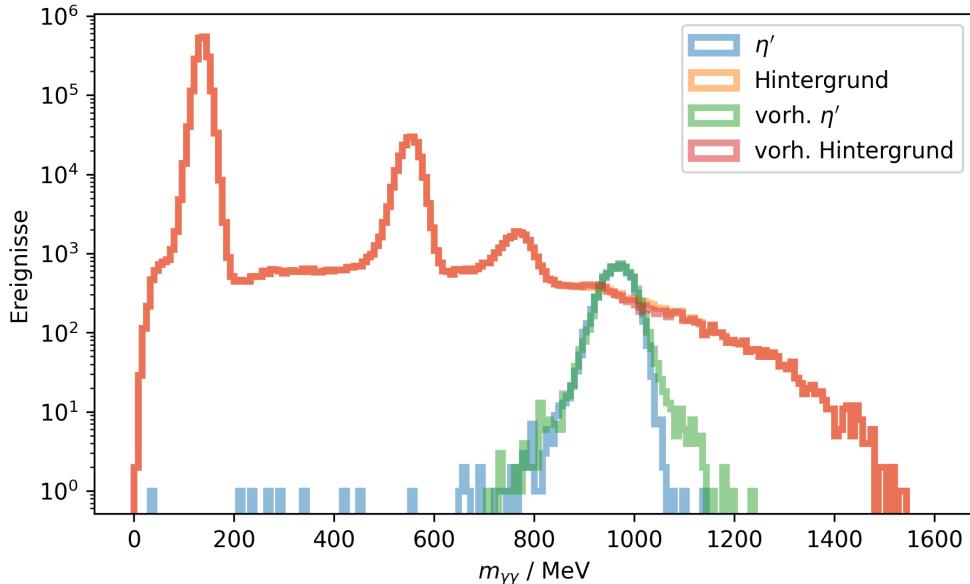


Abbildung 5.10: Vorhergesagte und tatsächliche Ereignisse in einem invarianten Massenspektrum. Der η' Verlauf wird vom NN sehr gut dargestellt.

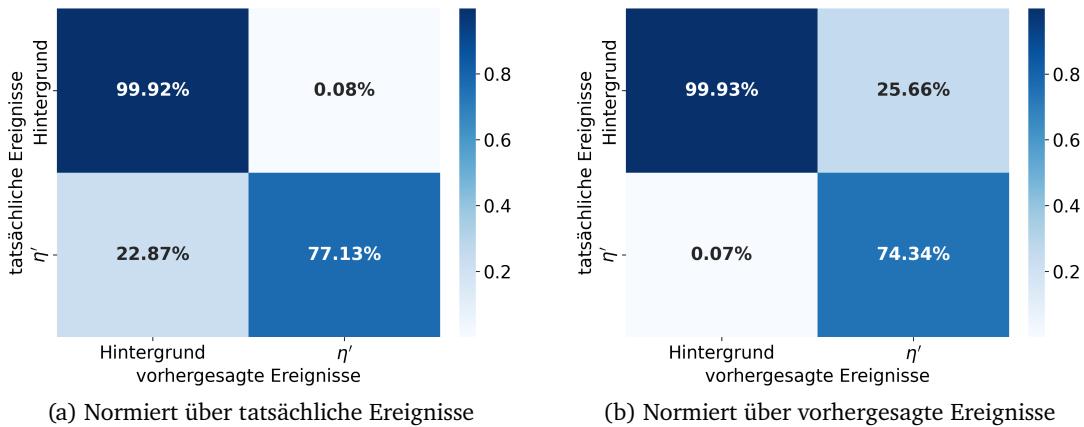


Abbildung 5.11: Konfusionsmatrizen für die η' oder Hintergrund Reaktionen. Normiert sind die Matrizen zeilenweise über die tatsächlichen Ereignisse (a) und spaltenweise über die vorhergesagten Ereignisse (b).

Insgesamt wird eine Genauigkeit von 99,85 % erreicht. Da das gesamte Spektrum jedoch von Hintergrund dominiert wird, sind die Genauigkeit für die η' Vorhersage aussagekräftiger. Diese betragen 77,13 % für alle tatsächlichen und 74,34 % für alle vorhergesagten η' Ereignisse. Eine anschauliche Darstellung dieser Werte ist in den Konfusionsmatrizen in Abbildung 5.11 zu sehen. Hier wird deutlich, dass die Hintergrundbestimmung mit 99,92 %

aller tatsächlichen und 99,93 % aller vorhergesagten Ereignissen Erfolg hat. Die η' Vorhersage hingegen wäre besser, wenn mehr Daten vorhanden wären. Zur Erinnerung: Insgesamt werden nur ca. 24 000 η' Ereignisse bei 7 Millionen Hintergrundreaktionen verwendet.

5.4 Analyse des Neuronalen Netzes

Nachdem das NN trainiert wurde, soll verstanden werden, welche Parameter für die Entscheidung ausschlaggebend sind. Dafür werden die in der Theorie 3.2 vorgestellten Methoden angewendet und die Ergebnisse verglichen.

5.4.1 Ausschließen von Parametern

Mit der ersten Methode werden mehrere NN trainiert, wobei jeweils ein Eingabeparameter weggelassen und schließlich die Genauigkeit verglichen wird. In Abbildung 5.12 sind die Gesamtgenauigkeiten für alle Parameter dargestellt. Die durchgezogene blaue Linie beschreibt die Ergebnisse für ein NN, das mit allen Parametern trainiert wurde. Durch mehrmaliges Trainieren können außerdem Fehler berechnet werden, wodurch die Breite der durchgezogenen Linie zustande kommt. E_2 fällt als erstes auf, da die Gesamtgenauigkeit bei diesem Parameter um ca. 0,04 % abnimmt. Damit ist dieser Wert besonders wichtig für das Netzwerk. CS_1 ist beispielsweise ein relativ unwichtiger Parameter, da sich hier die Genauigkeit kaum zu der Linie verändert. Vereinzelt finden sich Effizienzen, die über der Linie liegen, was bedeutet, dass ein Modell ohne diesen sogar besser wird. Die angegebenen Fehler machen allerdings deutlich, dass voreilige Schlüsse vermieden werden sollten.

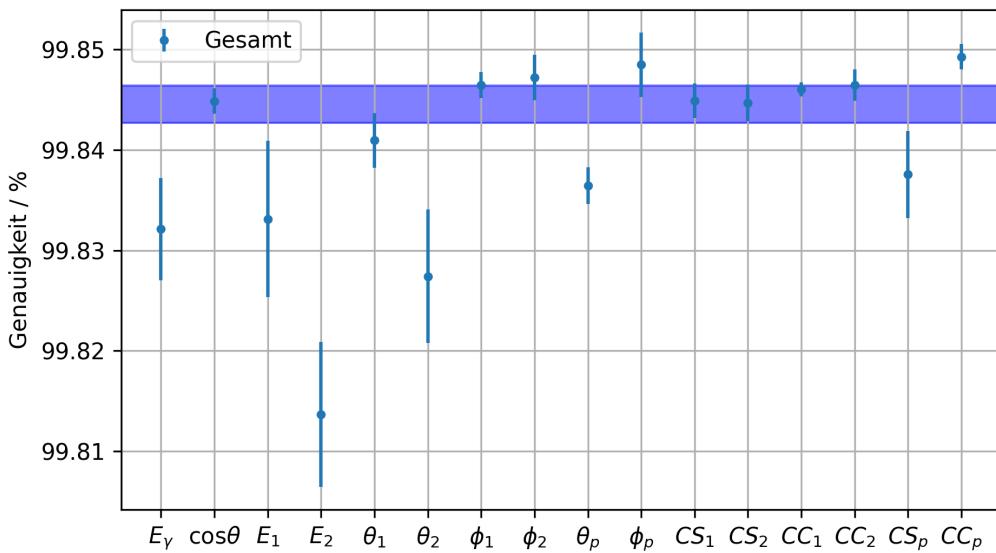


Abbildung 5.12: Gesamtgenauigkeiten vom NN, das jeweils mit einem Eingabeparameter weniger trainiert wurde. Der durchgezogene Bereich gibt das Ergebnis des Modells wider, das mit allen Parametern trainiert wurde. Starke Abweichungen von der Linie stehen für ein hohes Gewicht im NN.

Um die Bedeutsamkeiten oder auch Gewichte quantitativ zu vergleichen, wird die Differenz der Genauigkeiten in Abbildung 5.13 dargestellt, wobei auf den größten Wert normiert

5 Analyse von $\gamma p \rightarrow p \eta'$

wird. Dadurch lässt sich leicht erkennen, welche Parameter die ausschlaggebendsten sind. Die fünf mit den höchsten Gewichten sind: E_2 , θ_2 , E_γ , E_1 und θ_p .

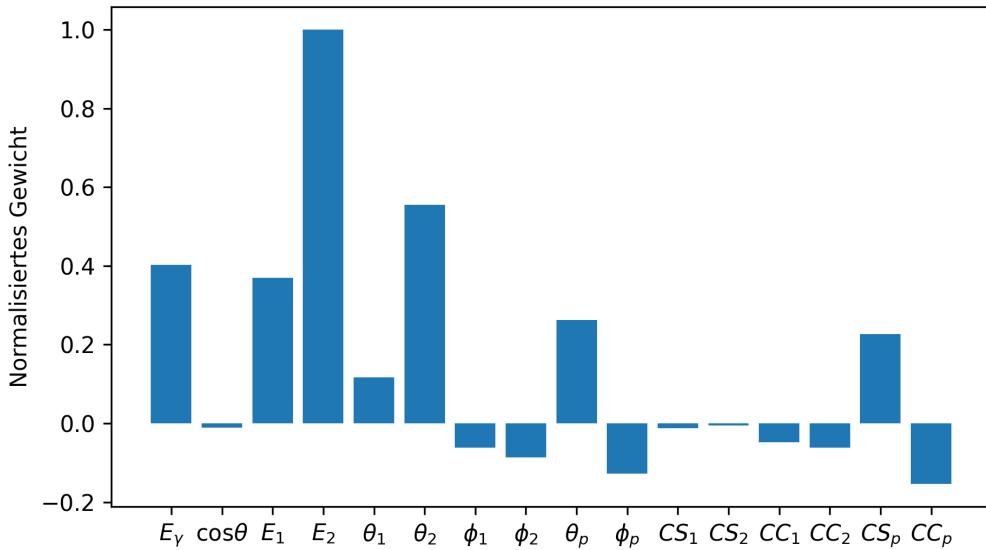


Abbildung 5.13: Die Abweichungen der Genauigkeit aus Abbildung 5.12 dargestellt. Die Gewichte wurden auf den stärksten Wert normiert. Mit dieser Methode sind E_2 , θ_2 , E_γ , E_1 und θ_p die wichtigsten Parameter.

5.4.2 Fuzzy Curves

Mithilfe der *Fuzzy Curves* Methode [2] wird für jeden Parameter ein Graph erstellt. In Abbildung 5.14 sind beispielhaft drei Eingaben gezeigt. Dabei werden die Werte $C_i(x_i)$ der Formel (3.7) gegen die normierten Daten x_i (E_γ , $\cos\theta$ und E_1) dargestellt. Es ist zu erkennen, dass sich dabei der Bereich von C unterscheidet. Durch Differenz von Maximal- und Minimalwert kann dann das Gewicht für jeden Parameter bestimmt werden.

Es wurden verschiedene Werte für b in der *Fuzzy Membership Function* (3.6) getestet und sich für $b = 1$ entschieden, da der Zähler der Funktion in der gleichen Größenordnung liegt¹³. Kleinere Änderungen von b in der Größenordnung haben keine Auswirkung auf die Kurven.

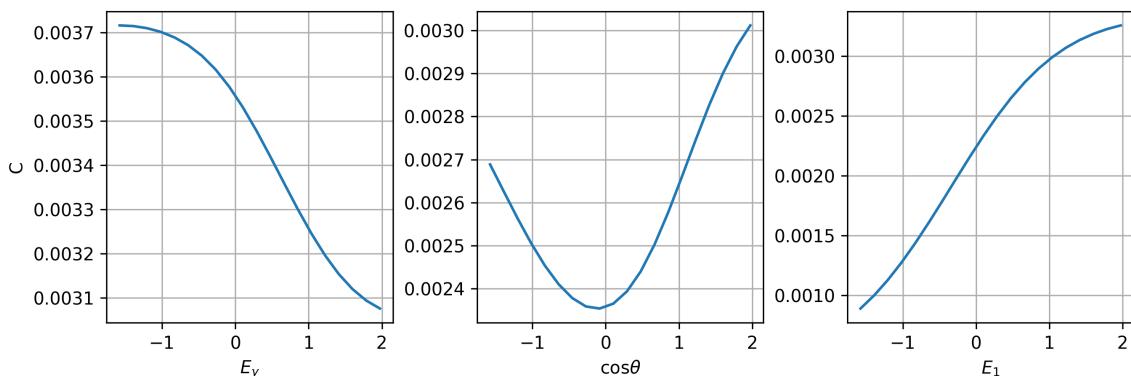


Abbildung 5.14: *Fuzzy Curves* für E_γ , $\cos\theta$ und E_1 . Die Bedeutsamkeit der Parameter wird durch die Größe des Wertebereichs definiert.

¹³Durch das Normieren der Daten wird, liegen die Werte alle in $[-2, 2]$.

In Abbildung 5.15 ist die Bedeutsamkeit normiert auf die größte Differenz dargestellt. Die zwei wichtigsten Parameter sind laut diesem Verfahren CC_1 und CC_2 . Das Ergebnis entsteht aber vermutlich dadurch, dass bei diesen Parametern keine kontinuierliche Verteilung vorhanden ist. Der Cluster Count für die zwei Photonen beträgt bei allen Ereignissen entweder 1 oder 2. Dadurch könnte bei der *Fuzzy Curves* Methode Fehler entstehen, da in Gleichung (3.6) durch das kontinuierliche x_i auch Werte zwischen 1 und 2 getestet werden. Wenn diese zwei Parameter weggelassen werden, sind die wichtigsten Eingaben: θ_1 , CS_2 , θ_p , E_2 und E_1 .

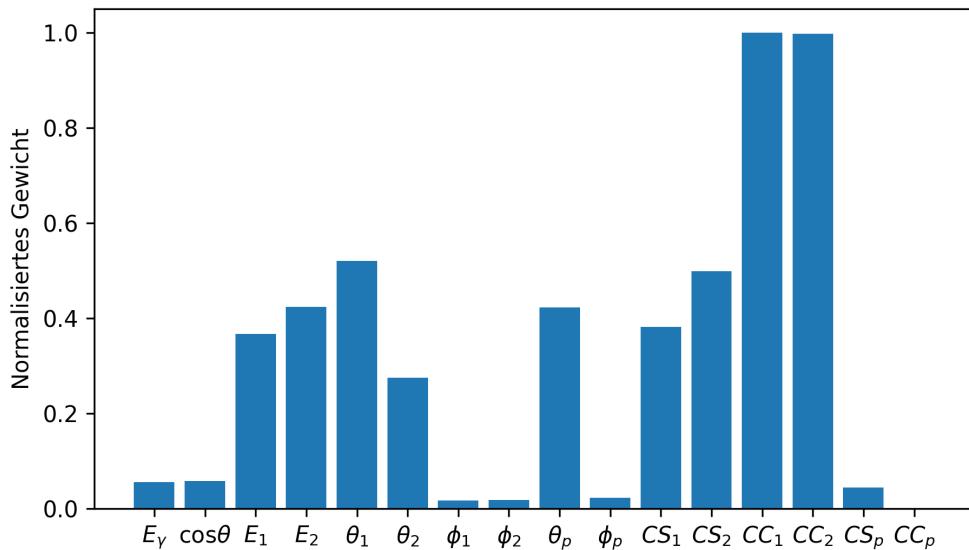


Abbildung 5.15: Gewichte der Parameter durch das *Fuzzy Curves* Verfahren. Die wichtigsten Parameter sind θ_1 , CS_2 , θ_p , E_2 und E_1 .

5.4.3 Weight of Evidence

Als letztes Verfahren wird die *Weight of Evidence* Methode betrachtet. Es bietet den Vorteil, dass einzelne Ereignisse ausgewertet werden können. Somit lassen sich spezifische Entscheidungen des NN nachvollziehen. Die Werte von Gleichung (3.8) geben daher an, wie wichtig ein Parameter *für* eine Entscheidung (positive Ausgabe) oder *gegen* eine Entscheidung (negative Ausgabe) ist.

Um einen Gesamteindruck von allen Ereignissen und allen Vorhersagen zu gewinnen, werden allerdings alle Ausgaben summiert. Spricht ein Eingabeparameter also häufig gegen eine spezifische Entscheidung, kann die Summe auch negativ sein. In Abbildung 5.16 ist das Ergebnis für alle Parameter gezeigt und auf den größten Wert normiert. Durch diese Methode werden CS_p , $\cos \theta$, θ_2 , θ_p und ϕ_2 als wichtigste Eingaben festgestellt.

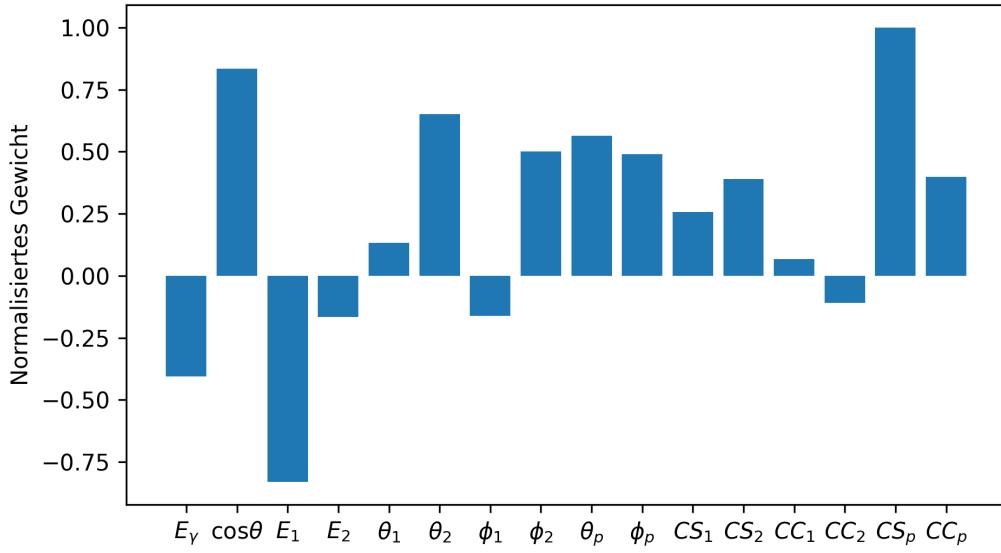


Abbildung 5.16: Gewichte der Parameter durch das *Weight of Evidence* Verfahren. Die wichtigsten Parameter sind CS_p , $\cos\theta$, θ_2 , θ_p und ϕ_2 .

5.4.4 Physikalische Erklärung

Um die Bedeutsamkeit der Parameter vergleichen zu können, sollten zunächst physikalische Zusammenhänge überlegt werden. Wie in Abbildung 5.1 zu sehen, wurden auf die Daten schon verschiedene Cuts angewendet. Ein letzter Cut würde die invariante Masse $m_{\gamma\gamma}$ der zwei Photonen betreffen, wodurch es Sinn ergibt, dass Parameter, die für $m_{\gamma\gamma}$ relevant sind, auch wichtig für das NN sind. Die invariante Masse berechnet sich gemäß Formel (5.1) [18].

$$m_{\gamma\gamma} = \sqrt{2E_1E_2(1 - \cos\alpha)} \quad (5.1)$$

Der Winkel α bezeichnet die Winkeldifferenz der beiden Photonen und lässt sich mithilfe des Detektor-Koordinatensystems in Formel (5.2) bestimmen.

$$\cos\alpha = \sin\theta_1 \cdot \sin\theta_2 \cdot \cos(\phi_1 - \phi_2) + \cos\theta_1 \cdot \cos\theta_2 \quad (5.2)$$

Damit sind alle auftauchenden Parameter in Gleichung (5.1) und (5.2) wichtig für die Reaktion. ϕ_1 und ϕ_2 sind eine Ausnahme, da die Werte aus Symmetriegründen keine Aussagekraft über bestimmte Ereignisse (η' oder Hintergrund) haben.

Die Strahlenenergie E_γ ist nur in einem gewissen Bereich interessant. Da ein η' Meson schwerer als π^0 , η oder ω Mesonen ist, muss E_γ eine gewisse Schwelle überschreiten, damit das schwere Teilchen erzeugt werden kann. Diese Größe kann auch von einem NN genutzt werden. Darüber hinaus ist die Strahlenenergie aber weniger bedeutend [7].

Durch die Koordinaten der Photonen und des Protons wird $\cos\theta$ rekonstruiert. Daher könnte der Wert für das Netzwerk überflüssig sein.

Der Polarwinkel θ_p für das Proton hängt direkt mit der Flugrichtung des Mesons im Zwischenzustand zusammen, da das Teilchen aus der Protonresonanz erzeugt wird. Je nach

Masse des Mesons könnte sich dieser Winkel unterscheiden. Für ϕ_p lässt sich das Symmetrieargument wiederholen.

Cluster Count und Cluster Size können für das Selektieren interessant sein, da sich diese Größen unterscheiden, wenn geladene oder ungeladene Teilchen registriert werden. Da bei der Reaktion zwei ungeladene Photonen und ein geladenes Proton erzeugt werden müssen, könnten diese Größen dabei helfen, falsche Photonen und Protonen zu erkennen und somit eine η' Reaktion auszuschließen [7].

5.4.5 Vergleich aller Methoden

Um alle Methoden miteinander zu vergleichen, sind in Tabelle 5.3 die jeweils fünf wichtigsten Parameter zusammengefasst. E_1 , E_2 , θ_2 und θ_p tauchen mehrmals in der Tabelle auf, wodurch diese Werte als bedeutsamsten angenommen werden können. Dass all diese Parameter bei der ersten Methode vorkommen, lässt darauf schließen, dass das *Ausschließen von Parametern* die beste Methode ist, um Gewichte im NN zu bestimmen.

| | Ausschließen von Parametern | Fuzzy Curves | Weight of Evidence |
|----|-----------------------------|--------------|--------------------|
| 1. | E_2 | θ_1 | CS_p |
| 2. | θ_2 | CS_2 | $\cos \theta$ |
| 3. | E_γ | θ_p | θ_2 |
| 4. | E_1 | E_2 | θ_p |
| 5. | θ_p | E_1 | ϕ_2 |

Tabelle 5.3: Vergleich der fünf wichtigsten Parameter, bestimmt durch *Ausschließen von Parametern*, *Fuzzy Curves* und *Weight of Evidence*. E_1 , E_2 , θ_2 und θ_p werden mehrmals erwähnt.

Nachdem die wichtigsten Eingabeparameter selektiert worden sind, kann überlegt werden die weniger wichtigen zu entfernen. Das verbessert die Laufzeit, aber beeinflusst die Genauigkeit nur geringfügig. In diesem Fall beträgt die Laufzeit für das Trainieren jedoch nur wenige Minuten. Daher muss nicht auf bestimmte Parameter verzichtet werden.

Da drei der allgemein wichtigsten Parameter in der invarianten Masse (5.1) vorkommen, sollte der Schluss nahe liegen, $m_{\gamma\gamma}$ erzielle als alleiniger Parameter dieselbe Genauigkeit. Tests haben hingegen gezeigt, dass durch Ersetzen dieser Parameter die Effizienz nachlässt. Das NN erkennt in den Verteilungen dieser Werte also über die invarianten Massen hinaus Korrelationen zur η' Reaktion.

Außerdem sind die vier besten Eingaben von der *Ausschließen von Parametern* Methode erkannt worden, sodass diese Methode am besten abschneidet. Für eine Laufzeitverbesserung sollte also erwogen werden, die Parameter mit den geringsten Gewichten aus Abbildung 5.13 zu entfernen.

An dieser Stelle kann zwar verstanden werden, worauf das NN besonders Wert legt. Eine genaue Vorstellung der Entscheidungsfindung dieses Netzwerkes bleibt jedoch unklar und wird nicht festgestellt.

5.5 Anwendung auf reale Daten

Nach einer umfassenden Analyse des NN, wird sich nun den realen Daten gewidmet. Auf diese Daten wurden die selben Cuts angewendet wie auf die MC Daten für das Trainieren. Es sei dazu gesagt, dass nicht exakt überprüft werden kann, ob das Modell die η' Ereignisse richtig erkennt. Es lässt sich lediglich das invarianten Masse Spektrum beurteilen. Des Weiteren wird angenommen, dass die Genauigkeit, die in Abschnitt 5.3 bestimmt wurde,

5 Analyse von $\gamma p \rightarrow p \eta'$

ebenso für die echten Daten gilt. Dadurch lässt sich abschätzen, wie viele η' Teilchen korrekt erkannt wurden und das Ergebnis mit anderen Methoden vergleichen.

Es wird das NN auf die gemessenen Daten angewendet, das mit 70 % der MC Daten trainiert wurde. Andererseits könnte zwar ein neues Modell mit allen Daten trainiert und die bisherige Genauigkeit als untere Grenze angenommen werden. Die exakten Genauigkeiten wären in diesem Fall aber unbekannt.

In Abbildung 5.17 ist das finale Ergebnis der invarianten Masse zu sehen. In orange sind alle vom NN erkannten Hintergrundereignisse und in blau alle η' Ereignisse dargestellt. Zusätzlich sind die realen Daten — anders als die MC Simulationen — von J. KRAUSE [9] zeitgewichtet¹⁴, was ebenfalls berücksichtigt werden muss. Das NN beachtet diesen Wert jedoch nicht, da es nicht damit trainiert werden konnte.

Der η' Peak hebt sich deutlich von den restlichen Daten ab, wobei positiv zu erwähnen ist, dass der Hintergrund im η' Bereich einen kontinuierlichen Verlauf folgt, und nicht wie in Abschnitt 5.2.2 abnimmt. Das lässt darauf schließen, dass das Modell korrekt mit den MC Simulationen trainiert wurde und daher die Ereignisse vorhersagen kann.

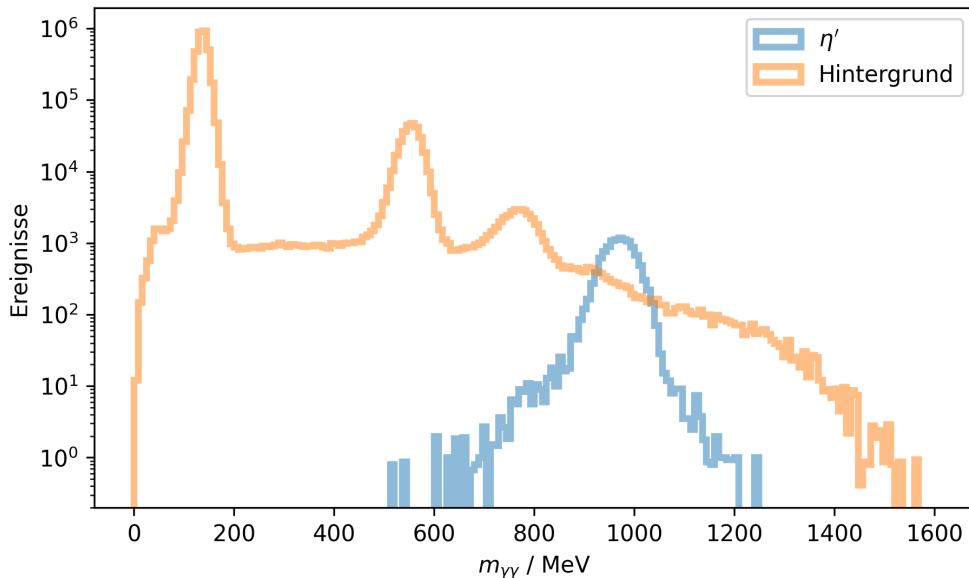


Abbildung 5.17: Die Hintergrund oder η' Vorhersage des NN für die realen Daten. Der η' Peak liegt an der erwarteten Stelle und der Hintergrund in diesem Bereich weist einen kontinuierlichen Verlauf auf. Das beides spricht für eine gute Erkennung.

Als weiteren Test der Vorhersage wird eine Gaußkurve der Form (5.3) an den η' Peak angepasst.

$$f(x) = A \cdot \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \quad (5.3)$$

In Abbildung 5.18 ist deutlich erkennbar, dass die η' Daten (in orange) der Gaußkurve (blau) entsprechen. Als Fehler der Hintergrund und η' Daten wird die Wurzel der Ereignisse verwendet. Bei dem Fit wird ein reduziertes $\chi^2 = 4,6$ erreicht. Die Parameter lauten

¹⁴Hierbei wird die Zeit, in der die Reaktion stattfindet, analysiert, um unkorrelierte Ereignisse zu eliminieren [9].

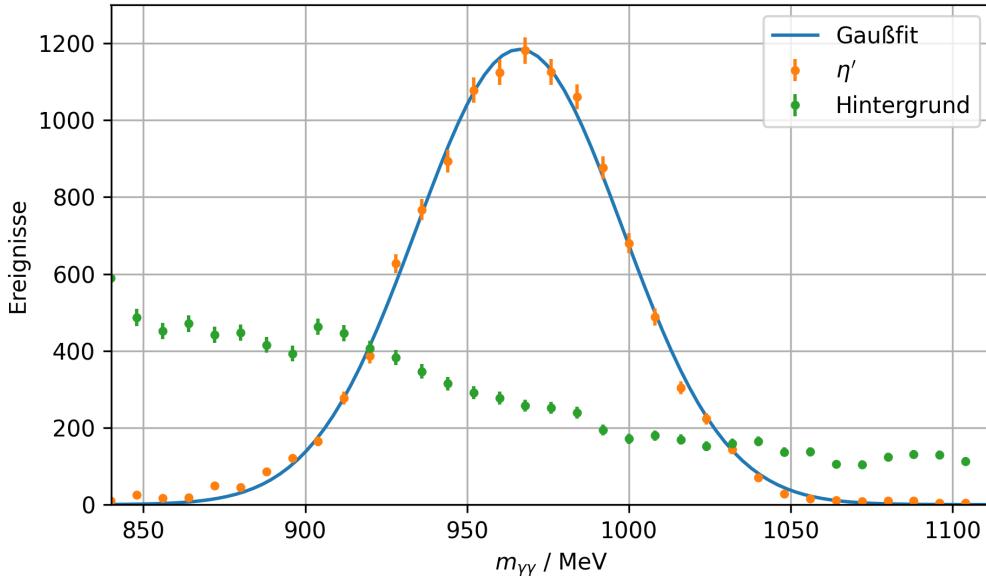


Abbildung 5.18: Invariante Masse von η' - und Hintergrund-Ereignisse mit Gaußfit. Vorhergesagt vom trainierten NN.

$$\mu = (966,21 \pm 0,31) \text{ MeV} ; \quad \sigma = (31,85 \pm 0,31) \text{ MeV} ; \quad A = 1184 \pm 10 .$$

Der Mittelwert μ der Kurve sollte bei dem Literaturwert der Masse $m_{\eta'} = 957,78 \text{ MeV}$ [4] des Mesons liegen und ist um ca. 0,9 % größer. Eine solche Untersuchung ist aber nicht Thema und Sinn dieses Verfahrens und kann als ausreichend korrekt angesehen werden. Zusammengefasst ist die Untergrundminimierung der Reaktion $\gamma p \rightarrow p\eta'$ mit einem NN gelungen.

5.5.1 Vergleich zur herkömmlichen Analyse

Um das Extrahieren der η' Ereignisse mithilfe des NN und mithilfe der Anwendung eines invarianten Massen Cuts¹⁵ vergleichen zu können, müssen zunächst die Unterschiede verdeutlicht werden. Der Cut zur Ereignis-Selektion wird in einem 2σ Bereich um das η' Maximum angewendet — es werden die Werte für μ und σ aus dem Gaußfit verwendet — und extrahiert dabei 95 % aller η' Mesonen. Dabei wird der Hintergrund in diesem Bereich nicht herausgefiltert, sondern bleibt nach diesem Cut erhalten.

Das NN hingegen wird die Anzahl aller gefundenen η' ausgeben, wobei der Fehler durch die Genauigkeit aus Abschnitt 5.3 definiert wird.

Wird der 2σ Cut auf das invarianten Massenspektrum angewendet, werden 15 802 Ereignisse extrahiert. Durch Integrieren der MC Simulationen in Abbildung 5.8 in dem gleichen Intervall wird das Verhältnis von η' zu allen Reaktionen von 57,21 % erreicht. Durch diese Methode werden also 9 041 η' entdeckt mit einem Signal-Hintergrund-Verhältnis von 1,34. Das trainierte Netzwerk sagt insgesamt 11 936 η' Ereignisse voraus. Mit der Genauigkeit von 74,34 % (siehe Abb. 5.11b) entspricht dies 8 873 tatsächlichen η' Reaktionen. Das entspricht einem Signal-Hintergrund-Verhältnis von 2,90.

¹⁵Gemeint ist ähnlich zu Abschnitt 5.1 das Filtern nach bestimmten Werten einer Reaktion, um Ereignisse ausschließen zu können.

5 Analyse von $\gamma p \rightarrow p \eta'$

In Tabelle 5.4 ist zusammengefasst, dass mit dem Cut zwar mehr η' in den extrahierten Daten enthalten sind, das Signal-Rausch-Verhältnis ist beim NN aber mehr als doppelt so hoch. Damit schneidet das Netzwerk in dieser Analyse besser ab.

| | 2σ Cut | Neuronales Netz |
|---------------------|---------------|-----------------|
| Extrahierte η' | 9 041 | 8 873 |
| SNR | 1,34 | 2,90 |

Tabelle 5.4: Extrahierte η' und Signal-Rausch-Verhältnis (SNR) für den 2σ Cut und das NN. Das Netzwerk ist wegen des hohen SNR besser.

6 Zusammenfassung und Ausblick

Die Untergrundminimierung mithilfe eines NN, das letztendlich zwischen η' Mesonen und anderen Teilchen unterscheidet, ist gelungen.

Nach dem Vergleich von *ROOT* und *Python* Systemen bei der Bilderkennung von Zahlen, erwies sich letzteres als zielführend. Damit das System in dieser Arbeit verwendet werden kann, wurden die in *Trees* formatierte Daten in für *Python* lesbare Textdateien umgewandelt, um einen Zugriff auf einzelne Reaktionen zu gewährleisten. Dabei sind für jede Reaktion Parameter angegeben, die Energie und Flugrichtung der Teilchen im Endzustand (ein Proton und zwei Photonen) beschreiben. Außerdem werden Cluster Size und Cluster Count des Detektorsystems für die Analyse verwendet. Das Neuronale Netz wurde mit MC Simulationen trainiert, die die genannten Parameter für alle relevanten Mesonen enthalten: η' , η , π^0 , $2\pi^0$, $\pi^0\eta$ und ω .

Sobald die Anteile dieser Datensätze der realen Verteilung im Energiebereich von 1 400 MeV bis 2 800 MeV entspricht, erreichte das Netzwerk eine Gesamtgenauigkeit von 99,85 % und 77,13 % aller η' Reaktionen wurden als solche erkannt. Mit diesem trainierten Modell konnten die realen Daten untersucht werden, und die Vorhersage von η' und Untergrund ist gelungen. Mit den extrahierten Teilchen wird ein Signal-Rausch-Verhältnis von 2,90 erreicht, während ein alternativer 2σ Cut auf die invariante Masse der zwei Photonen im Endzustand nur 1,34 erreicht.

Nachdem sich diese Methode der Ereignisselektion als erfolgreich herausgestellt hat, lassen sich NN dieser Art auch auf andere Reaktionen anwenden, deren Untergrundanalyse umfangreich ist. So kann das Programm, auf das in der Einleitung (Kapitel 1) verwiesen wurde, dabei helfen, bessere Ergebnisse zu erhalten. Für die weitere Analyse in *ROOT* können die Ereignisse ebenfalls in einen *Tree* gespeichert werden.

Sollte die Laufzeit für das Trainieren bei größeren Datenmengen zu einem Problem werden, bietet es sich an, Systeme mit Grafikkarten (GPU) zu verwenden. Diese besitzen mehr Recheneinheiten als Prozessoren (CPU) und eignen sich besser für ein NN [7]. Neuste Modelle von GPUs sind außerdem speziell für maschinelles Lernen ausgelegt.

Abbildungsverzeichnis

| | |
|--|----|
| 2.1 ELSA | 4 |
| 2.2 CBELSA/TABS Experiment | 5 |
| 2.3 Crystal Barrel | 5 |
| 3.1 Schematische Darstellung eines Künstlichen Neurons | 7 |
| 3.2 Darstellung Neuronales Netz | 8 |
| 3.3 Aktivierungsfunktionen (Sigmoid und ReLU) | 9 |
| 4.1 Datenset mit handgeschriebenen Zahlen | 14 |
| 4.2 Softmax-Funktion Beispiel | 15 |
| 4.3 Korrekt vorhergesagte Ziffern | 16 |
| 4.4 Falsch vorhergesagte Ziffern | 16 |
| 4.5 Konfusionsmatrix der Ziffern | 16 |
| 4.6 Vergleichung von Python und Root in Genauigkeit und Laufzeit | 17 |
| 5.1 Invariante Masse der gemessenen Daten mit angewandten Cuts | 19 |
| 5.2 Monte Carlos aller Teilchenreaktionen | 19 |
| 5.3 Invariante Masse der vorhergesagten η' und $2\pi^0$ Ereignisse vom NN | 22 |
| 5.4 Konfusionsmatrizen für die η' oder $2\pi^0$ Reaktionen. | 22 |
| 5.6 Genauigkeiten bei verschiedenen Loss-Funktionen | 24 |
| 5.8 Invariante Masse für alle Teilchenkanäle mit dem korrekten Verhältnis. | 26 |
| 5.10 Vorhergesagte invariante Masse mit allen Reaktionen | 28 |
| 5.11 Konfusionsmatrizen für die η' oder Hintergrund Reaktionen. | 28 |
| 5.12 Genauigkeiten des NN, das mit einem Parameter weniger trainiert wurde | 29 |
| 5.13 Gewichte der Parameter durch Trainieren mit einem Parameter weniger | 30 |
| 5.14 Fuzzy Curves für drei Beispiele | 30 |
| 5.15 Fuzzy Curves Verfahren | 31 |
| 5.16 Weight of Evidence | 32 |
| 5.17 Vorhersage des NN mit realen Daten | 34 |
| 5.18 Gaußfit vom untersuchten Teilchen | 35 |

Tabellenverzeichnis

| | |
|--|----|
| 2.1 Wahrscheinlichste η' Zerfälle | 6 |
| 5.1 Anteile der Teilchenreaktionen | 20 |
| 5.2 Anzahl der verwendeten Ereignisse | 26 |
| 5.4 Vergleich von Cut und NN | 36 |

Algorithmenverzeichnis

| | | |
|-----|---|----|
| 3.1 | Das Modell mit 50 Neuronen im ersten Layer und 20 Neuronen im zweiten Layer wird in Zeile 1 deklariert. In Zeile 2 wird es mit 100 Epochen trainiert und in Zeile 3 eine Vorhersage erzeugt. | 12 |
| 3.2 | Das Modell mit 50 Neuronen im ersten Layer und 20 Neuronen im zweiten Layer wird in Zeile 5 bis 9 deklariert. In Zeile 12 wird es mit 100 Epochen trainiert und in Zeile 15 eine Vorhersage berechnet. | 12 |
| 3.3 | Aufteilen der Rohdaten in Test- und Trainings-Pakete in Zeile 5. Normalisierung der Daten in Zeile 8 bis 11. | 13 |
| 4.1 | NN zur Ziffererkennung von 0 bis 9. Es besteht aus zwei Layer mit 100 und 50 Neuronen und aus Ausgabe-Layer mit zehn Neuronen. Trainiert wird das Modell in 100 Epochen. Für die Vorhersage wird außerdem die <i>Softmax</i> -Funktion verwendet. | 15 |
| 5.1 | Aufsetzen eines NN zur η' und $2\pi^0$ Erkennung. | 21 |

Literatur

- [1] Farah Noreen Afzal. *Measurement of the beam and helicity asymmetries in the reactions $\gamma p \rightarrow p\pi^0$ and $\gamma p \rightarrow p\pi^0$* . 2019. URL: <https://bonndoc.ulb.uni-bonn.de/xmlui/handle/20.500.11811/8064> (besucht am 11.08.2022).
- [2] A. H. Sung et al. *Ranking importance of input parameters of neural networks*. URL: <https://www.sciencedirect.com/science/article/pii/S0957417498000414> (besucht am 11.08.2022).
- [3] L. M. Zintgraf et al. *Visualizing Deep Neural Network Decision: Prediction Difference Analysis*. 2017. URL: <https://arxiv.org/pdf/1702.04595.pdf> (besucht am 22.08.2022).
- [4] R. L. Workman et al. *Particle Data Group*. 2022. URL: <https://pdg.lbl.gov/> (besucht am 06.09.2022).
- [5] R. L. Workman et al. „Review of Particle Physics“. 2022. URL: <https://academicoup.com/ptep/article/2022/8/083C01/6651666> (besucht am 22.09.2022).
- [6] Maximilian Both. *Ein einfacher Einstieg in künstliche neuronale Netze – Einführung in den Aufbau und die Notation von KNN*. URL: <https://www.tecislava.com/blog/knn-einfuehrung> (besucht am 09.08.2022).
- [7] Sebastian Ciupka. Private Kommunikation.
- [8] Reinhard Jahn. *Wie Nervenzellen miteinander reden*. URL: <https://www.mpg.de/synapse> (besucht am 29.08.2022).
- [9] Jakob Michael Krause. „Determination of the beam asymmetry Σ in η -and η' -photoproduction using Bayesian statistics“. 2022.
- [10] *Künstliches neuronales Netz*. URL: https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz (besucht am 09.08.2022).
- [11] Saber Malekzadeh Nabi Nabiyev. *Anomalous Sound Localization Estimation*. 2021. URL: https://www.researchgate.net/publication/349662206_Anomalous_Sound_Localization_Estimation (besucht am 05.09.2022).
- [12] Physikalisches Institut der Rheinischen Friedrich-Wilhelms-Universität Bonn. *Elektronen-Stretcher-Anlage ELSA*. URL: <https://www-elsa.physik.uni-bonn.de/index.html> (besucht am 29.08.2022).
- [13] Dr. rer. nat. Rüdiger Brause. *Neuronale Netze. Eine Einführung in die Neuroinformatik*. 1995.
- [14] Bacher Said. *Funktionsprinzip und Training neuronaler Netze*. URL: https://www.researchgate.net/publication/323586167_Funktionsprinzip_und_Training_neuronaler_Netze (besucht am 30.08.2022).
- [15] scikit-learn. URL: <https://scikit-learn.org/stable/> (besucht am 02.09.2022).
- [16] Tensorflow-Keras Sequential Model. URL: https://www.tensorflow.org/api_docs/python/tf/keras/Sequential (besucht am 01.09.2022).
- [17] The Digit Dataset. URL: https://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html#sphx-glr-auto-examples-datasets-plot-digits-last-image-py (besucht am 22.08.2022).

Literatur

- [18] Prof. Dr. Ulrike Thoma. „Vorlesung Experimentalphysik V, Kerne und Teilchen“. Universität Bonn. 2021–2022.
- [19] *TMultiLayerPerceptron Class Reference*. URL: <https://root.cern.ch/doc/master/classTMultiLayerPerceptron.html> (besucht am 01.09.2022).
- [20] Hochschule Trier. *Grundlagen Neuronaler Netze*. URL: https://www.hochschule-trier.de/fileadmin/Hauptcampus/Fachbereich_Informatik/Fernstudium/Dokumente/Leseproben/bdl_grundlagen.pdf (besucht am 30.08.2022).
- [21] Yannick Wunderlich. „Measurement of Double Polarization Observables in the Reactions $\gamma p \rightarrow p\pi^0$ and $\gamma p \rightarrow p\eta$ with the Crystal Barrel/TAPS Experiment at ELSA“. 2018.