TECHNISCHE UNIVERSITÄT DRESDEN

Optimierung von neuronalen Netzen für Matrixelement-Surrogate in Prozessgruppen

Bachelor-Arbeit
zur Erlangung des Hochschulgrades
Bachelor of Science
im Bachelor-Studiengang Physik

vorgelegt von

MATHIS ERIK SCHENKER,
geboren am 30.06.2003 in LEIPZIG.

Institut für Kern- und Teilchenphysik
Fakultät Physik
Bereich Mathematik und Naturwissenschaften
Technische Universität Dresden
2025

Eingereicht am 27. Januar 2025

1. Gutachter: Dr. Frank Siegert
2. Gutachter: Prof. Dr. Arno Straessner

Zusammenfassung

Die intensive Nutzung von Monte-Carlo-Ereignisgeneratoren in der Elementarteilchenphysik macht den effizienten Umgang mit Rechenkapazitäten notwendig. Dafür bieten sich Surrogat-basierte, Umgewichtungsalgorithmen an. In dieser Arbeit werden neuronale Netze mit von SHERPA generierte Ereignisse der Prozessgruppe $Z + 5$jets zur Vorhersage von Ereignisgewichten trainiert. Dabei werden Faktorisationseingenschaften von QCD-Matrixelementen ausgenutzt. Es wird eine neue, auf die Verwendung im Surrogat-basierten Umgewichten angepasste, loss-Funktion vorgestellt. Mit einer veränderten Trainingsmethode und nach einer abschließenden Hyperparameternoptimierung können deutliche Einsparungen an benötigter Rechenzeit erzielt werden. Dabei konnte im Mittel ein effektiver gain-Faktor von $f_{\text{eff}} = 2.6 \pm 1.0$ für das Surrogat-basierte Umgewichten erreicht werden.

Abstract

The intensive usage of Monte Carlo event generators in elementary particle physics makes it necessary to use computing capacities efficiently. Surrogate-based unweighting algorithms are suitable for this purpose. In this work, neural networks are trained with SHERPA-generated events of the process group $Z + 5$jets to predict event weights. Factorisation properties of QCD matrix elements are exploited. A new *loss* function adapted for use in surrogate-based unweighting is presented. With an adjusted training method and after a final hyperparameter optimisation, significant savings in required computing time were achieved. A mean effective gain factor of $f_{\text{eff}} = 2.6 \pm 1.0$ could be obtained.

Inhaltsverzeichnis

1 Einleitung

Der Durchlauf des ersten Protonenstrahls durch den *Large Hadron Collider* (LHC) im September 2008 markiert den Beginn einer neuen Ära der Elementarteilchenphysik, die mit dem Nachweis des Higgs-Bosons 2012 [1], [2] ihren ersten Höhepunkt erlebte. Mit den vier großen Experimenten ALICE, ATLAS, CMS und LHCb, sowie den fünf kleineren Experimenten, die entlang der 26,7 Kilometer langen Maschine aufgebaut sind, kann das Standardmodell der Elementarteilchenphysik, präziser als zuvor möglich, vermessen werden. Die enorme Anzahl registrierter Kollisionsereignisse - für 2024 erreichten allein die Experimente ATLAS und CMS jeweils eine integrierte Luminosität von $\mathcal{L}_{\text{int}} = 124\ \text{fb}^{-1}$ [3] - ermöglichen die Untersuchung immer seltener Prozesse.
Monte-Carlo-Ereignisgeneratoren werden genutzt, um, basierend auf dem Standardmodell, Teilchenkollisionen zu simulieren und anschließend das Detektorsignal zu konstruieren. Diskrepanzen zwischen den in Experimenten aufgezeichneten Streuungen und den Simulationsdaten lassen auf notwendige Ergänzungen des Standardmodells schließen. Diese Simulationen sind rechenzeitaufwendig, sodass ein großes Interesse an effizienten Simulationsmethoden besteht. Ein vielversprechendes Feld, in dem nach anwendungsorientierten Lösungen gesucht wird, ist *machine learning*. Explizit wird in dieser Arbeit auf die Verwendung von künstlichen neuronalen Netzen in *unweighting*-Algorithmen eingegangen, da sich diese als besonders nützlich erwiesen haben [4]. Diese Algorithmen werden genutzt, um aus generierten Streuereignissen geeignete Kandidaten für die weiterführenden Analyseschritte auszuwählen und so die Rechenzeit für die aufwendigen Detektorsimulationen zu reduzieren.

Das folgende Kapitel dient der Einführung physikalischer Größen zur Charakterisierung von Streuprozessen. Des Weiteren werden Monte-Carlo-Methoden und Umgewichtungsalgorithmen besprochen, die in der Elementarteilchenphysik bei der Erzeugung und Analyse simulierter Ereignisse Anwendung finden.

Kapitel 3 behandelt die Grundlagen des *machine learning*, dabei liegen die wesentlichen Elemente des Aufbaus und des Lernens künstlicher neuronaler Netze im Fokus. Es folgt die Erläuterung spezifischer Modifikationen, die zur Verwendung neuronaler Netze als Surrogate für Matrixelemente in *unweighting*-Algorithmen vorgenommen wurden.

In Kapitel 4 wird der gain-Faktor als Kriterium zur Performanceanalyse von neuronalen Netzen im Surrogat-gestützten *unweighting* eingeführt. Darauf aufbauend werden Veränderungen im Trainingsprozess nachgezeichnet, um eine neue auf dem gain-Faktor basierende

loss-Funktion für das Lernen neuronaler Netze nutzbar zu machen.

Zuletzt werden die Abhängigkeiten der erreichbaren gain-Faktoren von Trainingsparametern, aber auch von Struktureigenschaften neuronaler Netze, die mit der neuen loss-Funktion trainiert wurden, in Kapitel 5 untersucht. Im Besonderen wird dabei auf die Einflüsse der Ereigniszahl pro batch und der Anzahl der hidden layer eingegangen.

2 Physikalische Grundlagen

2.1 Das Standardmodell der Elementarteilchenphysik

Das Standardmodell der Elementarteilchenphysik ist eine Theorie zur Beschreibung der Konstituenten der Materie und deren Interaktionen durch quantisierte Felder. Das Gleichungssystem, dem diese Felder genügen, kann aus den fundamentalen Symmetrieeigenschaften des Universums gegenüber Eichtransformationen abgeleitet werden. Abgesehen von der Gravitation, welche im Standardmodell nicht berücksichtigt wird, werden die drei fundamentalen Kräfte - starke, schwache und elektromagnetische Wechselwirkung - durch Symmetriegruppen beschrieben. Die starke Wechselwirkung wird durch 8 Sorten an Gluonen g vermittelt und durch die Gruppe SU(3) beschrieben. Die elektromagnetische und die schwache Wechselwirkung können zu einer elektroschwachen Wechselwirkung vereinheitlicht werden, die durch die Gruppe $\mathrm{SU}(2) \times \mathrm{U}(1)$ repräsentiert wird. Die Eichbosonen dieser Gruppe sind das Photon γ, die $W^{\pm}$-Bosonen und das Z-Boson. Dadurch ergibt sich die Symmetriegruppe $\mathrm{SU}(3) \times \mathrm{SU}(2) \times \mathrm{U}(1)$.
Die Unterteilung der Materie in Leptonen und Quarks wird anhand ihrer differenzierten Interaktionsmöglichkeiten mit den Eichbosonen bestimmt. Quarks, wie das *down*-Quark d, können an jedes der Eichbosonen koppeln. Sie nehmen damit an allen drei Wechselwirkungen teil. Leptonen, wie das Elektron e^- und sein Anti-Teilchen das Positron e^+, wechselwirken nur elektroschwach.
Das Standardmodell hat 26 freie Parameter, deren Werte nicht aus dem Modell heraus berechnet werden können, sondern von außen übergeben werden müssen [5]. Dazu gehören die Massen der Elementarteilchen, aber auch die Kopplungskonstanten der fundamentalen Wechselwirkungen. Das präzise Vermessen dieser Parameter ist unter anderem das Ziel der Experimente am *Large Hadron Collider* (LHC) am CERN [6], an dem Protonen mit einer Energie von 13.6 TeV zur Kollision gebracht werden [7]. Die Zusammensetzung der Protonen wird durch die *parton distribution function* (PDF) beschrieben. Die PDF gibt eine hohe Wahrscheinlichkeit für die Zusammensetzung des Protons aus u-Quarks, d-Quarks und Gluonen an.
Eine Streuung von Protonen, bei der eine große Anzahl an Teilchen im Endzustand vorliegen, gehört zur Prozessgruppe $Z + 5$jets. In dieser Gruppe sind alle Prozesse zusammengefasst,

bei denen ein Z-Boson erzeugt wird und die im Detektor fünf hadronische Shower auslösen. Ein Prozess dieser Gruppe ist die Streuung $g\ d \to e^-\ e^+\ g\ g\ g\ g\ d$. Die Streuung eines Gluons g und eines down-Quarks d aus den kollidierenden Protonen resultiert in der Produktion von 5 hadronischen *jets*, ausgelöst durch die 5 stark wechselwirkenden Teilchen im Endzustand: 4 Gluonen und ein d-Quark. Die Produktion des Z-Bosons ist nur durch die Präsenz seiner Zerfallsprodukte, hier Elektron e^- und Positron e^+, nachzuweisen. Dieser Prozess kann durch seine Feynman-Diagramme dargestellt werden.

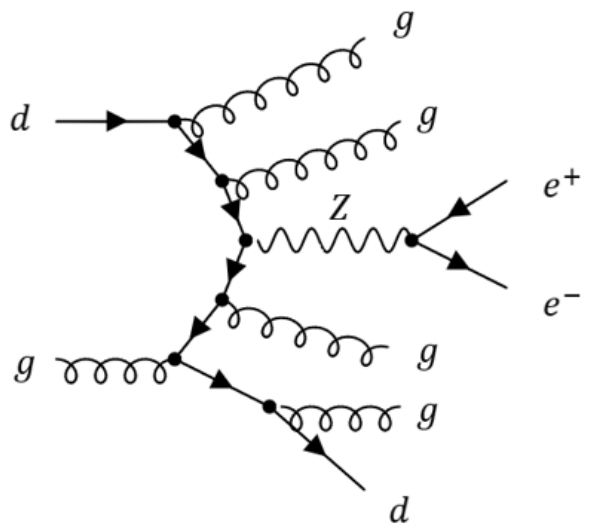


Abbildung 2.1: Eines der *leading order* Feynman-Diagramme für den Prozess $g\ d \to e^-\ e^+\ g\ g\ g\ g\ d$ aus der Prozessgruppe $Z + 5$jets. Erstellt mit *Feynman diagram maker* [8].

Das Feynman-Diagramm in Abbildung 2.1 ist eines der möglichen Diagramme in der führenden Ordnung. Linksseitig befinden sich die Teilchen im Anfangszustand des Prozesses, auf der rechten Seite die resultierenden Teilchen des Endzustandes. Das Diagramm stellt auch die Sub-Prozesse der betrachteten Streuung dar. Zusammen mit einem festen Satz an mathematischer Vorschriften, den Feynman-Regeln, kann aus dieser bildlichen Darstellung das Übergangsmatrixelement $\langle\Phi|M_{fi}|\Psi\rangle$ berechnet werden. Dieses gibt die Amplitude des Überganges des quantenmechanischen Systems von Zustand Ψ in den Zustand Φ wieder. Die Wahrscheinlichkeit des Übergangs kann aus der Übergangsamplitude bestimmt werden:

$$P(i \to f) \propto |\ \langle\Phi|M_{fi}|\Psi\rangle|^2. \tag{2.1}$$

Über Fermi's *Goldene Regel*:

$$\frac{\mathrm{d}\sigma(\theta,\phi)}{\mathrm{d}\Omega} \propto |\ \langle p'_1 \dots p'_m|M_{fi}|p_1 \dots p'_n\rangle|^2 \cdot \mathcal{PS} \tag{2.2}$$

kann diese Wahrscheinlichkeit mit dem Wirkungsquerschnitt σ verknüpft werden, eine physi-

kalische Größe die in den Streuexperimenten am LHC vermessen werden kann. Während sich der Phasenraumaktor $\mathcal{PS}$ aus der Zustandsdichte $\rho(E)$ ergibt, wird das Übergangsmatrixelement M_{fi} aus dem Zustand i nach f über die Feynman-Regeln aus den entsprechenden Diagrammen berechnet.
Der differentielle Wirkungsquerschnitt

$$\frac{\mathrm{d}\sigma(\theta,\phi)}{\mathrm{d}\Omega} = \frac{1}{\mathcal{L}}\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}\Omega}, \tag{2.3}$$

mit der Luminosität $\mathcal{L}$ des Beschleunigers, beschreibt die Streuung von $\mathrm{d}\mathcal{N}$ Teilchen im Raumwinkel $\mathrm{d}\Omega$. Mit (2.3) ist der differenzielle Wirkungsquerschnitt experimentell bestimmbar.
Für die Streuung zweier Teilchen mit m Teilchen im Endzustand muss folgendes $(3m-4)$-dimensionale Integral gelöst werden [5]:

$$\sigma = \frac{\hbar^2\mathcal{S}}{4\sqrt{(p_1\cdot p_2)^2 - {p_1}^2{p_2}^2}}\left[\int\prod_{i=1}^{m}\frac{c\,\mathrm{d}^3p_i'}{2E_i'(2\pi)^3}\right]|\mathcal{M}|^2(2\pi)^4\delta^{(4)}\left(\sum_{i=1}^{m}p_i' - p_2 - p_1\right) \tag{2.4}$$

mit

$\hbar$: reduziertes Planck'sches Wirkungsquantum

p_1, p_2: 4-er Impulse der Teilchen 1 und 2 im Anfangszustand

$\mathcal{S}$: einem statistischen Faktor zur Berücksichtigung der Ununterscheidbarkeit von Elementarteilchen des gleichen Typs

c: Lichtgeschwindigkeit

E_i': mit $E_i' = \sqrt{|\vec{\mathbf{p}}_i'|^2 + m_i^2}$ der Gesamtenergie des Teilchen i im Endzustand und

$|\mathcal{M}|^2$: dem Betragsquadrat des Matrixelementes $\mathcal{M}$.

Der Faktor $(2\pi)^4\delta^{(4)}\left(\sum_{i=1}^{m}p_i' - p_2 - p_1\right)$ gewährleistet die Energie- und Impulserhaltung. Nach Gleichung (2.4) kann der Wirkungsquerschnitt in seine physikalischen Komponenten geteilt werden: das Matrixelement $\mathcal{M}$, der Phasenraumfaktor und ein Lorentz-invarianter Normierungsfaktor. Der Phasenraumfaktor ist:

$$\mathcal{PS} = \left[\prod_{i=1}^{m}\frac{c\,\mathrm{d}^3p_i'}{2E_i'(2\pi)^3}\right](2\pi)^4\delta^{(4)}\left(\sum_{i=1}^{m}p_i' - p_2 - p_1\right) \tag{2.5}$$

und beschreibt die möglichen Aufteilungen der Energien und Impulse der Teilchen des Anfangszustandes auf die m Teilchen im Endzustand. Jede Kombination aus Energie und Impulsen entspricht einem Punkt im Phasenraum.

2.2 Monte-Carlo-Methoden und Ereignisgeneration

Monte-Carlo-Methoden nutzen Zufallszahlen zum Lösen von verschiedenen Problemstellungen, wie die Integration nicht trivialer Funktionen. Das Integral I der positiv-definiten Funktion $f(\vec{\mathbf{x}}) : \Omega \subset \mathbb{R}^n \to [0, \infty)$ über dem Integrationsbereich $\Omega = [0, 1]^n$ kann mit N uniform verteilten Zufallszahlen $\vec{\mathbf{x_i}} \in \Omega$ durch

$$I = \int_\Omega f(\vec{\mathbf{x}})\, \mathrm{d}\vec{\mathbf{x}} \approx \frac{1}{N} \sum_{i=1}^{N} f(\vec{\mathbf{x_i}}) \tag{2.6}$$

approximiert werden [9]. Das Ereignis $\vec{\mathbf{x_i}}$ trägt das Gewicht $w_i := f(\vec{\mathbf{x_i}})$. Das Integral I wird zu:

$$I \approx \frac{1}{N} \sum_{i=1}^{N} w_i, \tag{2.7}$$

dem Erwartungswert der Gewichte w_i. Für $N \to \infty$ konvergiert die rechte Seite von (2.6) gegen die linke Seite. Diese Methoden finden ihre Anwendung in der Teilchenphysik bei der Berechnung des Wirkungsquerschittes σ. In Eventgeneratoren, wie *SHERPA* [10], können mithilfe von Monte-Carlo-Methoden Wirkungsquerschnitte berechnet und Streuereignisse generiert werden.
Die generierten Ereignisse ermöglichen präzise Untersuchungen des Standardmodells [11]. Der Vergleich zwischen simulierten Daten, die auf der Theorie des Standardmodells beruhen, und Messdaten kann Hinweise auf „neue“ Physik liefern, oder auch die bestehende Theorie experimentell validieren. *SHERPA* kann verwendet werden, um zum Beispiel Proton-Proton-Kollisionen, wie sie im LHC stattfinden, zu simulieren. Die Simulation der hochenergetischen Streuprozesse ist aufgrund der großen Anzahl an möglichen Teilchen im Endzustand herausfordernd. Zusätzlich treten die zu untersuchenden Quarks und Gluonen nur in ihren gebundenen Zuständen, den Hadronen, auf. Diese können selbst weiter zerfallen. Gleichzeitig strahlen die hochenergetischen Teilchen Photonen und Gluonen ab. Zusätzlich schränkt die Detektorgeometrie den untersuchbaren Phasenraum ein.
Die erzeugten Ereignisse gehen mit einem Gewicht w in den Wirkungsquerschnitt ein. Mit den Gleichungen (2.6), (2.7) und (2.4) lässt sich das Gesamtgewicht eines solchen Ereignisses zu

$$w := \mathcal{PS} \cdot |\mathcal{M}|^2 \tag{2.8}$$

bestimmen [4], wobei $\mathcal{PS}$ das Gewicht des Phasenraumfaktors aus (2.5) ist.

2.3 Umgewichtungsprozess

Aus Gleichung (2.8) ergibt sich das Gesamtgewicht eines generierten Ereignisses. Diese können sich, wie in Abbildung 2.2 zu sehen, über viele Größenordnungen hinweg aufspannen. Die erzeugten Ereignisse können nicht nur zur Berechnung des Wirkungsquerschnittes ver-

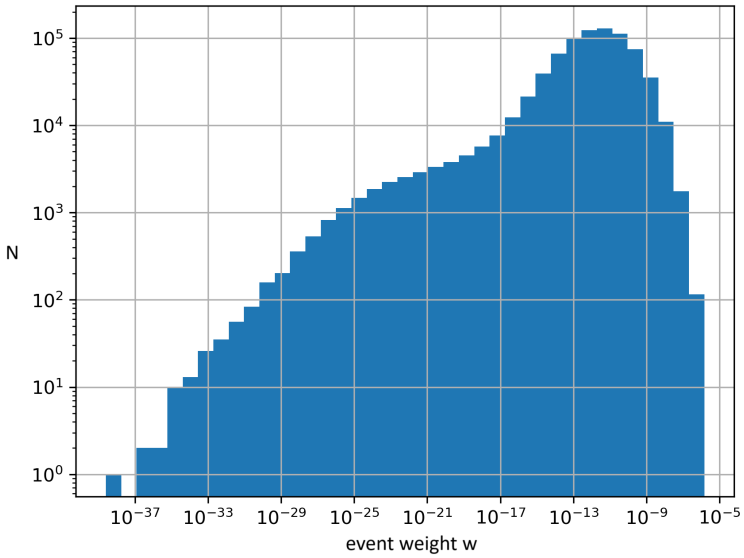


Abbildung 2.2: Verteilung der Gesamtgewichte der $1,5$ Millionen generierte Ereignisse im Test-Datensatz des Beispielprozesses $g\ d \to e^-\ e^+\ g\ g\ g\ g\ d$.

wendet werden. Mit ihnen können auch weitere Eigenschaften des Streuprozesses untersucht werden. Ist man beispielsweise an der Verteilung des Streuwinkels ϑ eines Teilchens im Endzustand interessiert, kann mit den generierten Ereignissen ein entsprechendes Histogramm gefüllt werden. Dabei müssen die Ereignisse entsprechend ihres Gewichts w berücksichtigt werden. Ereignisse mit geringen Gewichten tragen somit wenig zu der entsprechenden Statistik bei, die zu ihrer Auswertung benötigten Rechenkapazitäten sind jedoch die selben. Die anschließende Simulation der Detektor-Response kann 10 bis 100 s benötigen und trägt den größten Anteil an der benötigten Rechenzeit. Um Ressourcen zu sparen und die Auswertung zu beschleunigen, ist deshalb die Erzeugung gleich gewichteter Ereignisse notwendig.

Algorithmus 1 : einfacher Algorithmus zum Umgewichten der Events mit der Verwerfungsmethode [12].

while *true* **do**
  Erzeuge Phasenraumpunkt u;
  Berechne Ereignisgewicht w;
  Erzeuge eine uniform verteilte Zufallszahl $Z \in [0, 1)$;
  if $w > Z \cdot w_{\max}$ **then**
    return u *und* $\tilde{w} := \max(1, \frac{w}{w_{\max}})$
  end
end

Der hier dargestellte Algorithmus 1 beschreibt eine Methode zum Erzeugen gleich gewichteter Ereignisse unter Verwendung von Zufallszahlen. Die Idee besteht darin, nicht alle Ereignisse zu verwenden, sondern Ereignisse u nur mit einer Wahrscheinlichkeit $p = \frac{w}{w_{max}}$ zu akzeptieren. Dabei ist die Wahl des zu Beginn festzulegenden $w_{\max}$ entscheidend. Wird für $w_{\max}$ das absolute Maximum gewählt, so gilt für alle Ereignisse nach dem Umgewichten $\tilde{w} = 1$.
Liegen in der Mengen der generierten Ereignisse u große Ausreißer in den Gewichten w vor, das heißt einzelne Gewichte w_i, für die $w_i \gg w_{\text{rest}}$, dann wird die Wahrscheinlichkeit p akzeptiert zu werden, stark reduziert und deutlich mehr Ereignisse würden im Umgewichtungsschritt verworfen werden. Für eine aussagekräftige Analyse müsste eine erheblich größere Anzahl an Ereignissen generiert werden, wodurch die Zeitersparnis durch das Umgewichten kompensiert wird.
Eine andere Möglichkeit ist die Definition des Maximums über ein ε, sodass die über dem Maximum liegenden Werte nur einen Anteil von ε am Gesamtgewicht $\sum_i^N w_i$ besitzen. Dieses reduzierte Maximum ist somit das $(1 - \varepsilon)$-Quantil der Gewichte. Durch diese Wahl erhalten manche Ereignisse u_i ein Gewicht $\tilde{w}_i > 1$. Die akzeptierten Ereignisse sind dadurch partiell-ungewichtet.
Der beschriebene Algorithmus 1 hat den Nachteil, dass auch für verworfene Ereignisse das Phasenraumgewicht $\mathcal{PS}$ und das Matrixelement $\mathcal{M}$ ausgewertet werden müssen.

In [12] wird ein zweistufiger Algorithmus eingeführt, bei dem die ressourcenaufwendige Berechnung der Ereignisgewichte von einem geeigneten Surrogat s übernommen wird, um eine signifikante Reduktion der benötigten Rechenzeit zu erreichen. In Algorithmus 2 wird nach dem Erzeugen des Phasenraumpunktes u das Ereignisgewicht durch das Surrogat abgeschätzt. Im ersten Umgewichtungsschritt werden die Ereignisse mit einer Wahrscheinlichkeit von

$$p_1 = \frac{s}{w_{\max}} \tag{2.9}$$

Algorithmus 2 : zweistufiger Algorithmus zum Umgewichten der Ereignisse unter Verwendung eines Surrogates

```
while true do
    Erzeuge Phasenraumpunkt u;
    Approximiere Ereignisgewicht mittels Surrogat s;
    Erzeuge eine uniform verteilte Zufallszahl Z_1 ∈ [0, 1);
    if s > Z_1 · w_max then
        Berechne Ereignisgewicht w;
        Bestimme x = w/s;
        Erzeuge neue uniform verteilte Zufallszahl Z_2 ∈ [0, 1);
        if x > Z_2 · x_max then
            return u und w̃ := max(1, s/s_max) · max(1, x/x_max)
        end
    end
end
```

akzeptiert. Anschließend wird für die akzeptierten Ereignisse das exakte Gewicht w bestimmt. Um den Fehler durch das Surrogat zu korrigieren wird der Faktor $x := \frac{w}{s}$ eingeführt. In einem zweiten Verwerfungsschritt werden die übrig gebliebenen Ereignisse relativ zu x mit der Wahrscheinlichkeit

$$p_2 = \frac{x}{x_{\max}} \tag{2.10}$$

akzeptiert. Dabei sind auch hier die gewählten Maxima, absolut oder reduziert, vor der Anwendung der Methode festzulegen. Dabei bestimmen die Geschwindigkeit des Surrogates beim Auswerten der Gewichte, wie auch die Exaktheit der Vorhersage, den Rechenzeitgewinn dieser Methode.

3 Künstliche neuronale Netze

In dieser Arbeit werden tiefe künstliche neuronale Netze als Surrogate für die ressourcenaufwendige Berechnung von Matrixelementen genutzt. Die folgenden Erläuterungen bezüglich des Aufbaus und der Funktionsweise künstlicher neuronaler Netze folgen den Erklärungen des Lehrbuches „Künstliche neuronale Netze: das Lehrbuch“ von Dan Petterson [13].

3.1 Das künstlichen Neuron

Der grundlegende Baustein, aus dem künstliche neuronale Netze aufgebaut sind, ist das künstliche Neuron. Analog zu dem biologischen Vorbild akzeptiert dieses Eingabereize und erzeugt Ausgabereize, sobald die Eingabereize einen vorgegebenen Schwellenwert überschreiten. Jeder Eingabezweig i, der einen einzelnen Reiz x_i an das Neuron übergibt, ist mit einer Gewichtung g_i versehen. Gewichte und Reize sind im Allgemeinen reelle Zahlen. Der gesamte Eingabereiz ist

$$\Gamma = \sum_{\mathrm{i}=1}^{N} x_{\mathrm{i}} \cdot g_{\mathrm{i}} + b, \tag{3.1}$$

wobei b ein für jedes Neuron spezifischer Bias ist, der den Schwellenwert verschieben kann. Das Neuron kann mittels seiner Aktivierungsfunktion $A(\Gamma) = y$ beschrieben werden, die den Vektor der Eingabereize auf die Ausgabe abbildet. Für das Training tiefer neuronaler Netze hat sich die Aktivierungsfunktion „swish“, definiert durch

$$A(x, \beta) = \frac{x}{1 + \exp\{-\beta x\}}, \tag{3.2}$$

als geeignet erwiesen [14]. Der Parameter β kann konstant gewählt werden, aber auch im Verlauf des Trainings vom neuronalen Netz gelernt werden. Er bestimmt den Einfluss negativer Gesamtreize mit kleinem Betrag.

3.2 Tiefe neuronale Netze

Die individuellen künstlichen Neuronen können nicht nur einzeln eingesetzt, sondern in mehreren Schichten gruppiert werden. So können sie leistungsfähige Netzstrukturen bilden. Diese

zeigen die erstaunliche Eigenschaft, mithilfe vorgegebener Trainingsdaten, beliebige Funktionen erlernen zu können. Für die Anwendung als Matrixelement-Surrogate werden vollständig verbundene *feedforward*-Netze genutzt. Ein schematischer Aufbau der verwendeten Netze ist in Abbildung 3.1 zu finden. Die äußerste Schicht dient der Übergabe der zu verarbeitenden

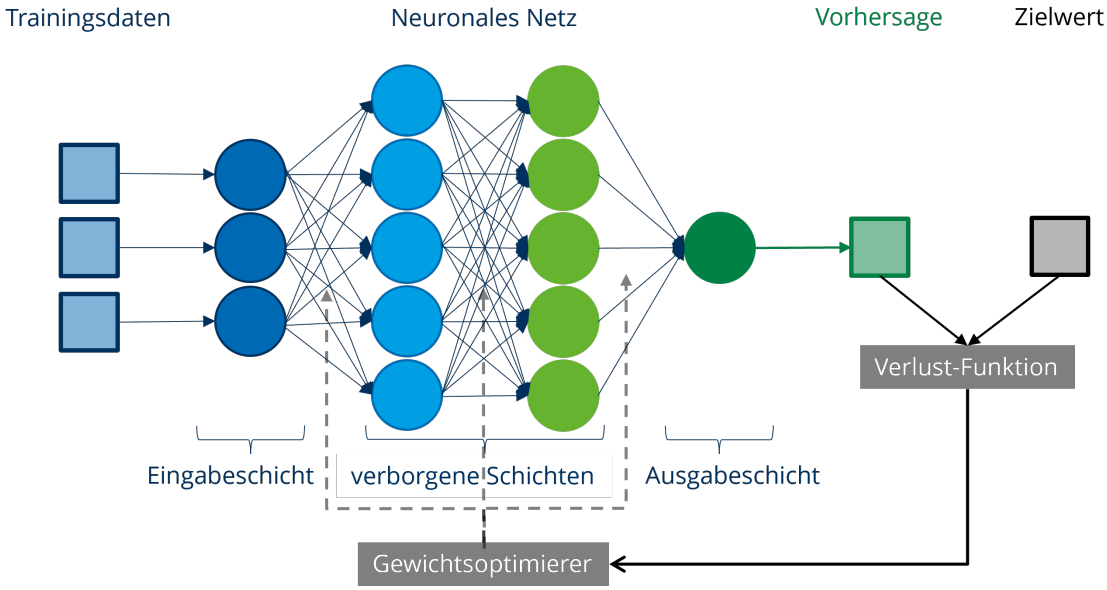


Abbildung 3.1: Schematischer Aufbau eines *feedforward*-Netzes mit der, für überwachtes Lernen notwendigen, Rückkopplungsschleife.

Daten an das Netz. Die Anzahl der Neuronen in dieser Schicht bestimmt die Anzahl der verschiedenen Parameter der Eingabe. Die Verarbeitung erfolgt in den tiefer liegenden hidden layer. Die verwendeten *feedforward*-Netze sind dadurch charakterisiert, dass der Ausgabereiz eines Neuron nur an Neuronen der direkt darunter liegenden Schicht weitergegeben wird. Vollständig verbunden sind diese Netze, wenn jedes Neuron der tiefer liegenden Schicht alle Ausgabereize der darüber liegenden Schicht zur Verfügung hat. Die Anzahl der Neuronen in der Ausgabeschicht bestimmt die Anzahl der ausgegebenen Werte, also die Struktur der Vorhersage. Jede der internen Verbindungen, mit denen der Ausgabereiz eines Neuron a der Schicht n als Eingabereiz an ein Neuron b der Schicht m weitergegeben wird, sind durch ihre Gewichte $g_{n_a m_b}$ charakterisiert. Diese Gewichte enthalten das gesamte „Wissen“ des Netzes.

3.3 Überwachtes Lernen

Das Ziel des Lernens ist die Verbesserung der Vorhersage des Netzes. Dieses Ziel ist durch das Anpassen der Gewichte $g_{n_a m_b}$ erreichbar. Beim überwachten Lernen müssen zusätzlich zu den Trainingsdaten auch die Zielwerte vorliegen. Der Vergleich der Vorhersage des Netzes mit den Zielwerten erfolgt über eine zu definierende Fehlerfunktion, auch loss-Funktion genannt. Der so bestimmte loss ist ein Maß für die Abweichungen der Vorhersage von den Zielwerten.

Eine Verbesserung der Vorhersage des Netzes ist gleichbedeutend mit der Reduktion des loss. Eine gängige loss-Funktion ist der *mean squared error* (MSE):

$$l_{\text{MSE}} = \frac{1}{2} \sum_{i=1}^{N} \left(y_i - z_i\right)^2 \tag{3.3}$$

mit dem loss l, der Anzahl der Ausgabeparameter N, der Ausgabe y_i des Neurons i der Ausgabeschicht, sowie dem entsprechenden Zielwert z_i.
Die Vorhersage des Netzes ist eine beliebig komplizierte Abbildung, die Funktion der Aktivierungsfunktion A und der internen Gewichte $g_{n_a m_b}$ ist. Der loss $l = l(A, \vec{\mathbf{x}}, g_{n_a m_b}, \vec{\mathbf{z}})$ als Funktion der Aktivierungsfunktion A, des Eingabevektors $\vec{\mathbf{x}}$ und der Zielwerte $\vec{\mathbf{z}}$ kann als Skalarfeld interpretiert werden, das jedem Punkt im hoch-dimensionalen Raum der Gewichte $g_{n_a m_b}$ den Wert des loss des Netzes zuordnet. Das globale Minimum dieses Feldes, beschrieben durch die Konfiguration der Gewichte $g_{n_a m_b}$, die den geringsten loss und damit die beste Ausgabe liefert, ist in diesem Raum analytisch nicht zu finden.
Der Backpropagation-Algorithmus ist eine iterative Methode des überwachten Lernens. Zur Minimierung des loss wird der Gradient desselben in Bezug auf die internen Gewichte $\frac{\partial l}{\partial g_{n_a m_b}}$ genutzt. Dafür muss die Aktivierungsfunktion der künstlichen Neuronen stückweise differenzierbar sein. Der Gradient zeigt für skalare Felder in Richtung des größten Anstieges. Werden die Gewichte entgegen des Gradienten verändert, verringert sich der loss. Die Veränderung der Gewichte erfolgt iterativ, nach folgender Regel:

$$g'_{n_a m_b} = g_{n_a m_b} + \Delta g_{n_a m_b} = g_{n_a m_b} - \eta \frac{\partial l}{\partial g_{n_a m_b}} \tag{3.4}$$

mit der Lernrate η, die den Einfluss des Gradienten reguliert. Die Gewichte werden, beginnend in der Ausgabeschicht, entgegen der ursprünglichen Berechnungsrichtung des Netzes relativ zum jeweiligen Einfluss auf den loss angepasst. Ein vollständiger Durchlauf aller Trainingsdaten durch das Netz einschließlich der Berechnung des loss und der Veränderung der Gewichte wird als Epoche bezeichnet.
Optimierer wie *ADAM* [15] realisieren diesen Backpropagation-Algorithmus. *Adam* nutzt dabei zwei zusätzliche Momentumsterme, welche die Gradienten der vorhergehenden Epochen berücksichtigen, um die Lernrate dynamisch anzupassen. Extremen Änderungen der Gewichte kann so entgegengewirkt werden. Das Finden des globalen Minimums ist durch *ADAM* nicht garantiert, dafür kann zumindest ein lokales Minimum gefunden werden. Die Trägheitsterme ermöglichen das „Herausspringen“ aus den lokalen Minima und das potentielle Finden einer geeigneteren Konfiguration der Gewichte.

3.4 Hyperparameter

Hyperparameter eines neuronalen Netzes sind Eigenschaften des Netzes, die während des Lernens nicht angepasst werden können [16]. Dazu gehören unter anderem:

- die Zahl der hidden layer,
- die individuelle Zahl der künstlichen Neuronen je hidden layer,
- die Aktivierungsfunktion,
- der Gewichtsoptimierer und die damit verbundene Lernmethode,
- die loss-Funktion,
- der Algorithmus zum Setzen der Gewichte bei Initialisierung des Netzes, oder
- die Zahl der Ereignisse pro batch.

Bei einer großen Zahl von Trainingsdaten erscheint es sinnvoll, nicht nach jedem Ereignis die internen Gewichte durch den Optimierer verändern zu lassen. Die benötigte Zeit, um eine Epoche abzuschließen, wäre enorm. Der andere Extremfall, nur nach dem Durchlauf aller Trainingsdaten die Gewichte anzupassen, würde den Fortschritt in den einzelnen Epochen wiederum stark einschränken. Deshalb wird der Trainingsdatensatz in batch, eingeteilt. Nach der Auswertung eines batch wird der mittlere loss auf diesem bestimmt. Die Anpassung der Gewichte erfolgt mit dem mittleren Gradienten. Die Epoche endet, nachdem jedes batch bearbeitet wurde.

3.5 Umsetzung mit OPTIMA

Die Konstruktion der neuronalen Netze und ihr Lernen werden mit *OPTIMA* [17] realisiert. Das Framework von *OPTIMA* bietet nicht nur die Möglichkeit einzelne neuronale Netze zu trainieren, sondern stellt auch Methoden zur Hyperparameter-Optimierung bereit. Die Netze können durch die Übergabe von Konfigurationsdateien individualisiert und auf die spezifische Problemstellung angepasst werden. Zur weiteren Lektüre und genaueren Diskussion der bereitgestellten Möglichkeiten sei hier auf [18] verwiesen.
Zum Training der künstlichen neuronalen Netze stehen die durch *SHERPA* [11] generierten Ereignisdaten von rund 3 Millionen Streuereignissen zur Verfügung. Diese werden zu jeweils 50 % auf einen sog. Trainingsdatensatz und einen Test-Datensatz aufgeteilt. Der Trainingsdatensatz wird für das Lernen des Netzes genutzt. Mit dem Test-Datensatz können die Fehler des trainierten neuronalen Netzes besser eingeschätzt werden. Von dem Trainingsdatensatz

werden wiederum 20 % der Daten als Validierungsdatensatz abgespalten. Diese Ereignisdaten werden während des Trainings des Netzwerkes genutzt, um parallel die Performance zu bewerten und bei Verschlechterung der Ergebnisseearly stopping [16] auszulösen. Dabei wird nach jeder Epoche k die Performance des Netzes bezüglich einer vorher zu definierenden Metrik, auch als Monitorvariable bezeichnet, auf dem Validierungsdatensatz ausgewertet, und mit dem Ergebnis der Epoche $k-1$ verglichen. Hat sich das Netz verbessert, wird die Gewichtskonfiguration des Netzes gespeichert. Wenn sich das Netz innerhalb eines vorher festgelegten Intervalls, dem Abbruch-Intervall, nicht verbessert, wird der Lernvorgang beendet. Als Ergebnis wird nun das Netz mit der besten Leistung auf den Validierungsdaten ausgegeben.

Nach Gleichung 2.8 besteht das Gesamtgewicht aus Phasenraumgewicht $\mathcal{PS}$ und dem Betragsquadrat des Matrixelementes $\mathcal{M}$. Die physikalische Information über den Streuprozesses ist dabei im Matrixelement enthalten. Die Berechnung desselben ist aufwendiger als die Berechnung des Phasenraumgewichtes. Indem man die Ausgabe des neuronalen Netzes mit dem Phasenraumgewicht des untersuchten Ereignisses multipliziert, und das Resultat als Vorhersage $\vec{\mathbf{y}}$ an die loss-Funktion übergibt, wird das Netz zum Erlernen des Matrixelementes angehalten. In [19] wird eine Methode eingeführt, wie neuronalen Netzen das Erlernen der Struktur von QCD-Matrixelementen erleichtert werden kann. Die Matrixelemente $|M_{n+1}|$ können in QCD-Dipolfunktionen entwickelt werden:

$$|M_{n+1}|^2 \simeq \sum_{\{ijk\}} C_{ijk} D_{ijk}, \tag{3.5}$$

dabei ist D_{ijk} die Dipol-Funktion der Partonen i, j und k und C_{ijk} der entsprechende Koeffizient. Die Anzahl der Neuronen in der Ausgabeschicht n_{Ausgabe} des Netzes muss nun der Zahl der verwendeten Entwicklungsfunktionen entsprechen. Der Ausgabereiz eines Ausgabeneurons wird mit immer derselben Dipolfunktion multipliziert und die resultierenden Werte werden summiert. Mit dieser Prozedur muss das neuronale Netz zur Bestimmung der Ereignisgewichte nur die Entwicklungskoeffizienten C_{ijk} vorhersagen.
Jedes Trainingsereignis besteht aus den folgenden Informationen: den vier Komponenten der 4-er Impulse der beteiligten Teilchen im Anfangs- und Endzustand, die aus den Impulsen berechnete Dipolvariablen [19] und weiteren kinematischen Invarianten. Jeder Wert wird an ein individuelles Neuron der Eingabeschicht übergeben.

4 Training mittels neuartiger loss-Funktion

In [19] wird der gain-Faktor als Größe eingeführt, um den Zeitgewinn durch die Verwendung von Matrixelement-Surrogat beim Umgewichten von Ereignissen nach Algorithmus 2 gegenüber der Standard-Verwerfungsmethode nach Algorithmus 1 zu messen. Im folgenden Abschnitt wird dieser gain-Faktor als loss-Funktion zum Training neuronaler Netze verwendet. Dabei wird auf die Anpassungen eingegangen, die für den Trainingsprozess vorgenommen wurden. Um die Veränderungen durch die neue loss-Funktion einschätzen zu können, werden die Resultate zu einer anderen, bereits etablierten loss-Funktion in Beziehung gesetzt.

4.1 Definition des gain-Faktors

Der effektive gain-Faktor[1] f_{eff} ergibt sich aus dem Vergleich zweier Zeiten, T_{Standard} und T_{Surrogat} und beschreibt die Einsparung an Computing-Ressourcen, die durch die Anwendung des zweistufigen Umgewichtungs-Algorithmus erreicht werden kann

$$f_{\text{eff}} := \frac{T_{\text{Standard}}}{T_{\text{Surrogat}}}. \tag{4.1}$$

Beide Methoden der Umgewichtung erzeugen eine Anzahl $N_{\text{akzeptiert}}$ an Ereignissen eines Prozesses, die für weiterführende Analysen verwendet werden. T_{Standard} ergibt sich aus der durchschnittlichen Zeit, die benötigt wird, um das Ereignisgewicht, bestehend aus Matrixelement und Phasenraumgewicht, zu berechnen. Mit $\langle t_{\text{PS}} \rangle$ und $\langle t_{\text{ME}} \rangle$ sowie der Gesamtzahl der Ereignisse N_{gesamt} erhält man:

$$T_{\text{Standard}} = N_{\text{gesamt}} \cdot (\langle t_{\text{ME}} \rangle + \langle t_{\text{PS}} \rangle). \tag{4.2}$$

T_{Surrogat} ergibt sich aus den mittleren Zeiten, die benötigt werden, um für jedes Ereignis das Phasenraumgewicht zu berechnen $\langle t_{\text{PS}} \rangle$ und das Surrogat zu erzeugen $\langle t_{\text{surr}} \rangle$. Zusätzlich wird die Zeit $\langle t_{\text{ME}} \rangle$ benötigt, um das Matrixelement für die im zweiten Verwerfungsschritt

[1] Im weiteren Verlauf dieser Arbeit wird der effektive gain-Faktor der Einfachheit halber nur als gain-Faktor bezeichnet.

zu berücksichtigenden Ereignisse $N_{2.,\mathrm{surr}}$ zu berechnen. Das ergibt:

$$T_{\mathrm{Surrogat}} = N_{1.,\mathrm{surr}} \cdot (\langle t_{\mathrm{PS}} \rangle + \langle t_{\mathrm{surr}} \rangle) + N_{2.,\mathrm{surr}} \cdot \langle t_{\mathrm{ME}} \rangle \,. \tag{4.3}$$

Durch die Verwendung der Effizienzen der jeweiligen Umgewichtungsschritte $\varepsilon_{\mathrm{full}}$, $\varepsilon_{1.,\mathrm{surr}}$ und $\varepsilon_{2.,\mathrm{surr}}$ kann man den gain-Faktor in die Form

$$f_{\mathrm{eff}} = \frac{1}{\frac{\langle t_{\mathrm{surr}} \rangle + \langle t_{\mathrm{PS}} \rangle}{\langle t_{\mathrm{ME}} \rangle + \langle t_{\mathrm{PS}} \rangle} \cdot \frac{\varepsilon_{\mathrm{full}}}{\varepsilon_{1.,\mathrm{surr}} \cdot \varepsilon_{2.,\mathrm{surr}}} + \frac{\langle t_{\mathrm{ME}} \rangle}{\langle t_{\mathrm{ME}} \rangle + \langle t_{\mathrm{PS}} \rangle} \cdot \frac{\varepsilon_{\mathrm{full}}}{\varepsilon_{2.,\mathrm{surr}}}} \tag{4.4}$$

bringen. Die Definitionen der Effizienzen

$$\varepsilon_{\mathrm{full}} := \frac{N_{\mathrm{akzeptiert}}}{N_{\mathrm{gesamt}}} \quad , \quad \varepsilon_{1.,\mathrm{surr}} := \frac{N_{2.,\mathrm{surr}}}{N_{1.,\mathrm{surr}}} \quad \text{sowie} \quad \varepsilon_{2.,\mathrm{surr}} := \frac{N_{\mathrm{akzeptiert}}}{N_{2.,\mathrm{surr}}} \tag{4.5}$$

werden ebenso aus [19] übernommen. Diese sind durch die Zahl der Ereignisse, die in den jeweiligen Umgewichtungsschritten $N_{1.,\mathrm{surr}}$, bzw. $N_{2.,\mathrm{surr}}$ überprüft werden müssen, und die Zahl der akzeptierten Ereignisse $N_{\mathrm{akzeptiert}}$ gegeben.

4.2 Anpassungen zur Verwendung im Training

Die in Gleichung (4.4) definierte Größe ist noch nicht zur Verwendung als loss-Funktion für das Training eines neuronalen Netzes geeignet. Wie in Abschnitt 3.3 erwähnt, basiert das Training der Netze auf einem *backpropagation*-Algorithmus zur Minimierung des loss. Im Gegensatz dazu ist die Maximierung des gain-Faktors und die damit verbundene Verbesserung des Umgewichtungsprozesses das erklärte Ziel des Trainings. Daraus ergibt sich die naheliegende Definition der loss-Funktion l zu:

$$l_{\mathrm{gain}} := \frac{1}{f_{\mathrm{eff}}}. \tag{4.6}$$

Wird nun der loss l minimiert, ist die Maximierung des gain-Faktors f_{eff} gewährleistet.
Die in (4.5) definierten Effizienzen geben in dieser Form keinen Aufschluss über die Abhängigkeit des loss l von der Vorhersage s des Netzes, welche zur Anpassung dieser benötigt wird. Durch die Verwendung der in [19] getroffenen Abschätzungen

$$\varepsilon_{\mathrm{full}} \approx \frac{\langle w \rangle}{w_{\mathrm{max}}} \tag{4.7}$$

$$\varepsilon_{2.,\mathrm{surr}} \approx \frac{\langle x \rangle}{x_{\mathrm{max}}} \tag{4.8}$$

sowie in angepasster Form

$$\varepsilon_{1.,\text{surr}} \approx \frac{\langle s \rangle}{s_{\max}} \tag{4.9}$$

ist der loss l eine Funktion der Zielgrößen w und der Vorhersagen des Surrogates s. Dabei werden die Effizienzen nur innerhalb eines batch bestimmt. $\langle w \rangle$ und $\langle s \rangle$ sind die arithmetischen Mittel innerhalb des batch, und $\langle x \rangle$ das mit s gewichtet Mittel. Die Maxima von w, s und x sind nicht die absoluten Maxima innerhalb des batch, sondern werden als 99.9-Quantil definiert. Für $x_{\max}$ wird jedoch jener Wert x_i als reduziertes Maximum verwendet, für den $s_i \cdot x_i$ das 99.9-Quantil von $\vec{s} \cdot \vec{x}$ ist. Zur Berechnung von $\varepsilon_{\text{full}}$ werden nur die Gewichte w der Ereignisse verwendet, deshalb ist diese Effizienz unabhängig von der Vorhersage s. Diese Effizienzen sind positiv definiert, da um den Charakter der Wahrscheinlichkeit des Ereignisses, welcher durch w und s widergespiegelt wird, gerecht zu werden, nur mit ihren Beträgen $|w|$ und $|s|$ gerechnet wird[2].
Der loss ist in jedem batch weiterhin beeinflusst von der Verteilung der Vorhersage s, welche in $\varepsilon_{1.,\text{surr}}$ und $\varepsilon_{2.,\text{surr}}$ eingeht. Diese können durch das Training des Netzes verbessert werden. Die in Gleichung (4.4) eingehenden Zeiten $\langle t_{\text{ME}} \rangle$ und $\langle t_{\text{PS}} \rangle$ wurden über alle Ereignisse des Trainingsdatensatzes gemittelt. Die Zeit zur Berechnung durch das Surrogat $\langle t_{\text{surr}} \rangle$ wurde mittels bereits trainierter Netze bestimmt. Diese in Tabelle 4.1 abgebildeten Zeiten werden für das gesamte Training festgesetzt und auch zur späteren Bestimmung des gain-Faktors eines trainierten Netzes verwendet.

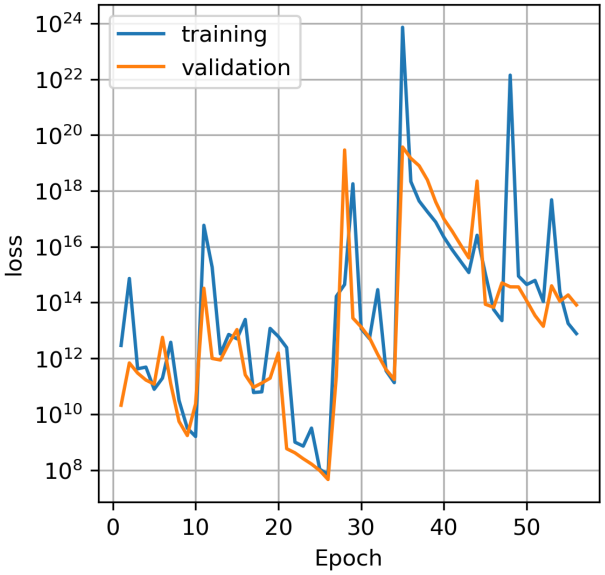
Zeit	t/ms
$\langle t_{\text{ME}} \rangle$	10.01
$\langle t_{\text{PS}} \rangle$	1.92
$\langle t_{\text{surr}} \rangle$	0.3

Tabelle 4.1: Gemittelte Zeiten, die zur Ermittlung des Gewichts eines Ereignisses benötigt werden.

4.3 Training der Netze

Der gain-Faktor ist als loss-Funktion zur Verwendung im Training eines einzelnen Netzes implementiert.

[2]Um die Übersichtlichkeit zu wahren, werden im weiteren Verlauf die Symbole w und s immer im Sinne von $|w|$ und $|s|$ verwendet.



a) Darstellung des numerischen Wertes der loss-Funktion für jede Trainingsepoche, ausgewertet auf dem Trainings- und Validierungsdatensatz.

Hyperparameter	Wert
hidden layer	4
Neuronen pro hidden layer	128
Ereignisse pro batch	512
Aktivierungsfunktion	swish
Monitorvariable	gain_{val}
Länge des Abbruch-Intervalls	30
Optimier-Algorithmus	ADAM
$\varepsilon_{\max}$	0.001

b) Wesentliche Hyperparameter für das trainierte Netz.

Abbildung 4.1: Trainingsverlauf und -parameter für das mit dem reziproken gain-Faktor trainierte neuronale Netz.

Das Netz wurde 56 Epochen für die in 4.1b angegebenen Hyperparameter trainiert. Der Abbruch des Trainings erfolgt nach dem verstreichen des Abbruch-Intervalls, da sich der loss innerhalb von 30 Epochen nicht weiter gegenüber dem bereits besten loss verbessert hat. Bezeichnend sind die Sprünge des loss über mehrere Größenordnungen, wie sie in 4.1a nach Epoche 10, 26 oder Epoche 35 zu sehen sind. Die Epochen vor den Sprüngen sind durch einen stark abfallenden loss gekennzeichnet. Dadurch wird die Lernrate durch den Optimierungsalgorithmus weiter erhöht, was schlussendlich zu einer Überkorrektur der internen Gewichte und zu einem Anstieg des loss geführt hat. Nach diesen Sprüngen konvergiert die loss-Funktion nicht schnell genug, um eine Verbesserung des Netzes bewirken zu können. Das führt zum Abbruch des Trainings.

4.4 Wechsel der loss-Funktion

Das Training mit dem gain-Faktor als loss-Funktion von Beginn an hat sich, wie im vorherigen Abschnitt 4.3 beschrieben, als nicht zielführend erwiesen. Es wird nun untersucht, ob ein Wechsel der loss-Funktionen während des Trainings die Verwendung des gain-Faktors ermöglicht. Konkret wird das Netz in der ersten Phase des Trainings mit einer anderen loss-Funktion trainiert, damit in der zweiten Phase der gain-Faktor verwendet werden kann. In der ersten Phase sollen die zuerst zufällig gewählten internen Gewichte $g_{n_a m_b}$ so verändert wer-

den, dass eine Approximation der Matrixelemente w_{ME} ermöglicht wird. In der zweiten Phase wird mithilfe des gain-Faktors die Feinabstimmung der internen Gewichte vorgenommen. Dadurch soll ein höherer gain-Faktor erreicht werden, als durch ein vollständiges Training mit der in der ersten Phase verwendeten loss-Funktion möglich ist.

4.4.1 Areasinushyperbolicus-loss-Funktion

Die erwähnte loss-Funktion basiert auf der arsinh-Funktion und ist durch

$$l_{\mathrm{arsinh}} = \frac{1}{2}\left(\operatorname{arsinh}\left(\frac{s_{\mathrm{ME}}}{w_{\mathrm{scale}}}\right) - \operatorname{arsinh}\left(\frac{w_{\mathrm{ME}}}{w_{\mathrm{scale}}}\right)\right)^2 \tag{4.10}$$

definiert. Dabei ist s_{ME} die Vorhersage des Netzes, w_{ME} das Matrixelement des Ereignisses und w_{scale}, das minimale Matrixelement, welches in den Trainingsdaten auftritt. Durch die Transformation der Matrixelemente w

$$w \to w' = \operatorname{arsinh}\left(\frac{w_{\mathrm{ME}}}{w_{\mathrm{scale}}}\right) \tag{4.11}$$

soll die Approximation der Matrixelemente vereinfacht werden. Die Skalierung reduziert die Größenordnung, in der die Vorhersagen getroffen werden müssen, während die Anwendung des arsinh den Einfluss großer Ereignisgewichte verringern soll. Eine detaillierte Beschreibung der arsinh-loss-Funktion und der mit ihr trainierten Netze ist in [19] zu finden.

Der gain-Faktor dieses Netzes nähert sich dem Wert von $f_{\mathrm{eff}} \approx 1$ an.

4.4.2 Umsetzung des Übergangs

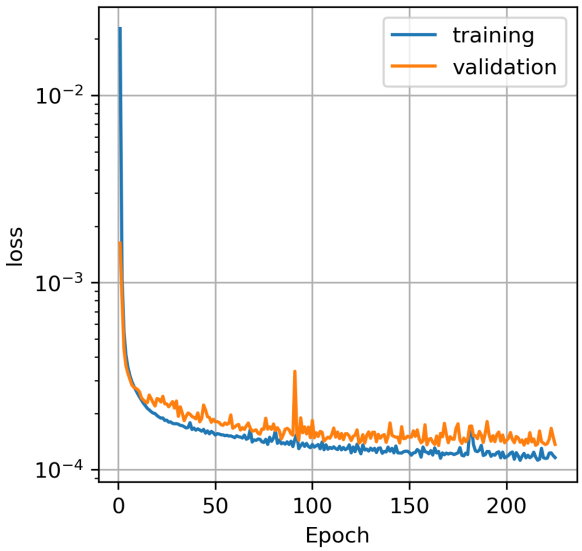
Der Wechsel zwischen den loss-Funktionen wird durch folgende Definition ermöglicht:

$$l = (1 - c) \cdot l_{\mathrm{arsinh}} + c \cdot l_{\mathrm{gain}} \tag{4.12}$$

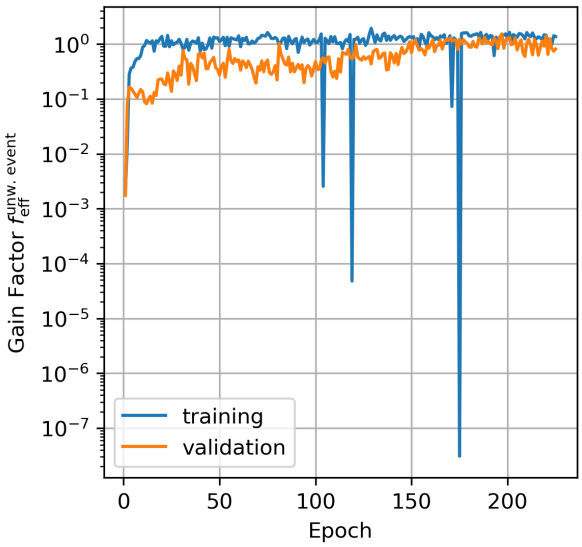
wobei

$$c = \begin{cases} 0, & k \leq k_{\mathrm{change}} \\ 1, & k > k_{\mathrm{change}} \end{cases} \tag{4.13}$$

ist, mit der aktuellen Epoche des Trainings k und der Epoche, nach der der Wechsel stattfinden soll k_{change}. Die diskreten Werte $\{0; 1\}$ von c ermöglichen, dass abhängig von der Trainingsepoche k nur eine der beiden loss-Funktionen verwendet wird. Das Resultat dieser Änderung ist in Abbildung 4.3 zu sehen.



a) loss in logarithmischer Darstellung.



b) gain-Faktor in logarithmischer Darstellung.

Abbildung 4.2: Vollständiges Training eines neuronalen Netzes mit der arsinh-loss-Funktion.

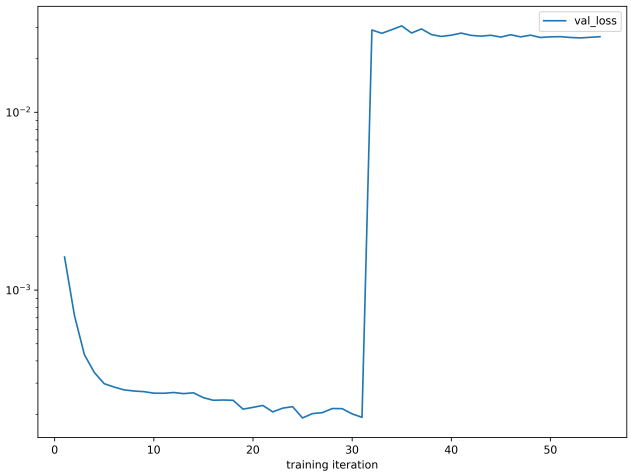


Abbildung 4.3: loss auf den Validierungsdaten mit einem Wechsel der loss-Funktionen nach Epoche 30.

Der Sprung nach Epoche 30 wird genau durch diesen Wechsel hervorgerufen, da sich die nummerischen Werte von l_{gain} und l_{arsinh} unterscheiden. Da $l_{\text{val,gain}}$ den niedrigsten Wert von $l_{\text{val,arsinh}}$ innerhalb des Abbruchintervalls von 30 Epochen nicht unterschreiten kann, wird das Training abgebrochen. Der dafür zu erreichende loss von $l_{\text{val,gain}} \approx 10^{-3}$, also ein gain-Faktor von $g \approx 1000$, ist unrealistisch. Um die Fortführung des Trainings zu gewährleisten, wird ein konstanter Übergangsfaktor T_1 eingeführt, der die Höhe des Sprungs nach dem Wechsel nach Epoche k_{change} verringert, in dem der Validierungs-loss $l_{\text{val,gain}}(k_{\text{change}}+1)$ der Epoche $k_{\text{change}}+1$ an den Validierungs-loss $l_{\text{val,arsinh}}(k_{\text{change}})$ vor dem Wechsel angeglichen wird, das

heißt

$$l_{\text{val,arsinh}}(k_{\text{change}}) \stackrel{!}{=} \mathrm{T}_1 \cdot l_{\text{val,gain}}(k_{\text{change}} + 1). \tag{4.14}$$

Da $l_{\text{val,gain}}(k_{\text{change}} + 1)$ in Epoche k_{change} noch nicht bekannt sein kann, muss eine andere Größe verwendet werden. Während des Trainings werden abgesehen vom loss weitere Metriken auf den Trainings- und Validierungsdaten ausgewertet, die nicht für die *backpropagation* genutzt werden. Dazu gehört auch der gain-Faktor auf den Validierungsdaten ausgewertet auf den Modellvorhersagen des neuronalen Netzes der Epoche k, im Folgenden als $\text{gain}_{\text{val}}(k)$ bezeichnet. Der Wert dieser und anderer Metriken steht am Ende jeder Epoche zur Verfügung. Unter der Annahme, dass es keine signifikanten Sprünge in der Größenordnung von gain_{val} von Epoche k_{change} auf $k_{\text{change}} + 1$ gibt, kann T_1 durch

$$\mathrm{T}_1 \approx l_{\text{val,arsinh}}(k_{\text{change}}) \cdot \text{gain}_{\text{val}}(k_{\text{change}}) \tag{4.15}$$

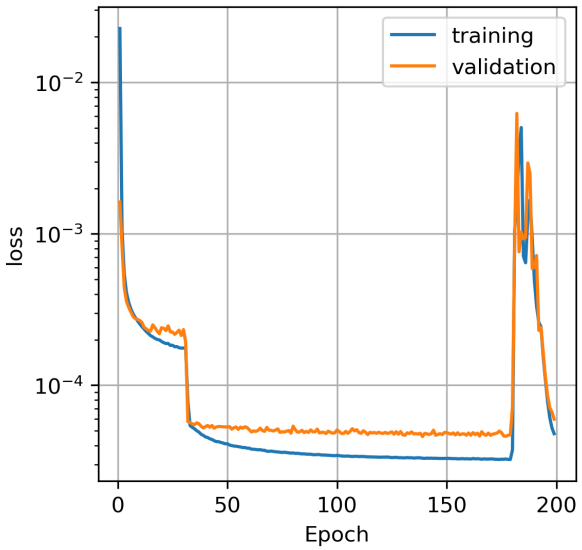
abgeschätzt werden.

4.4.3 Training des Netzes mit dem gain-Faktor

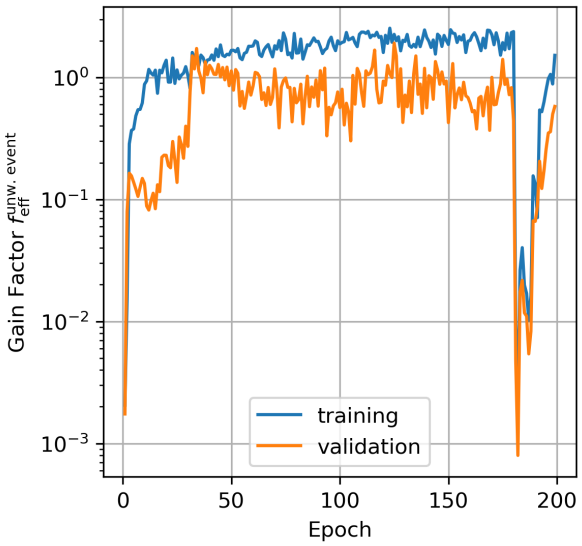
Der resultierende loss l ab Epoche $k_{\text{change}} + 1$ ist somit

$$l = \mathrm{T}_1 \cdot l_{\text{gain}}. \tag{4.16}$$

Die Effekte des Übergangs mit dem in (4.15) definierten Faktor auf das Training eines einzelnen neuronalen Netzes sind in 4.4 abgebildet. Das Training wird abgebrochen, sollte sich der Validierungs-loss innerhalb von 60 Epochen nicht verbessert haben. Die Erhöhung des Abbruch-Intervalls auf 60 Epochen soll verhindern, dass der Abbruch nicht durch die auftretenden Fluktuationen bedingt ist, sondern durch die Stagnation des Validierungs-loss.



a) loss in logarithmischer Darstellung.



b) gain-Faktor in logarithmischer Darstellung.

Abbildung 4.4: Wechsel der loss-Funktion nach Epoche 30 unter Verwendung des Übergangsfaktors T_1.

In 4.4a ist der Einfluss von T_1 deutlich an dem Sprung im Graphen bei Epoche 30 zu sehen. Der Sprung verringert den loss von $\approx 2 \cdot 10^{-4}$ auf einen numerischen Wert von $\approx 5 \cdot 10^{-5}$. Der aktuell niedrigste Validierungs-loss liegt damit in der Epoche nach dem Sprung und verhindert das vorzeitige Abbrechen des Trainings. Im weiteren Verlauf trainiert das Netz mit dem gain-Faktor. Der Abbruch erfolgt in Epoche 200, damit erreicht das Netz in Epoche 140 den geringsten loss und sollte an diesem Punkt die beste Approximation der Matrixelemente und deshalb den größten gain-Faktor liefern.
In 4.4b ist zu erkennen, das der gain-Faktor erst nach dem Wechsel der loss-Funktion den Wert von $f_{\text{eff}} \geq 1$ erreicht, ein Bereich in dem sich die Verwendung des neuronalen Netzes lohnt. Der Sprung verdeutlicht, dass ein großer Teil der Leistungsverbesserung des Netzes in der ersten Epoche erreicht wird, in der der neue loss zur *backpropagation* verwendet wird. Während der gain-Faktor auf dem Validierungsdatensatz Schwankungen um $f_{\text{eff}} = 1$ unterworfen ist, steigt dieser auf dem Trainingsdatensatz weiter an. Das sprunghafte Ansteigen des loss in Epoche 170 entstand durch eine Überkorrektur der internen Gewichte durch den Optimizer, womit das Unterschreiten des derzeit geringsten loss unmöglich wurde. Der gain-Faktor dieses Netzes in der Epoche mit dem geringsten loss ist mit $f_{\text{eff}} = 0.58$ jedoch schlechter als das Netz, welches mit der arsinh-loss-Funktion trainiert wurde.

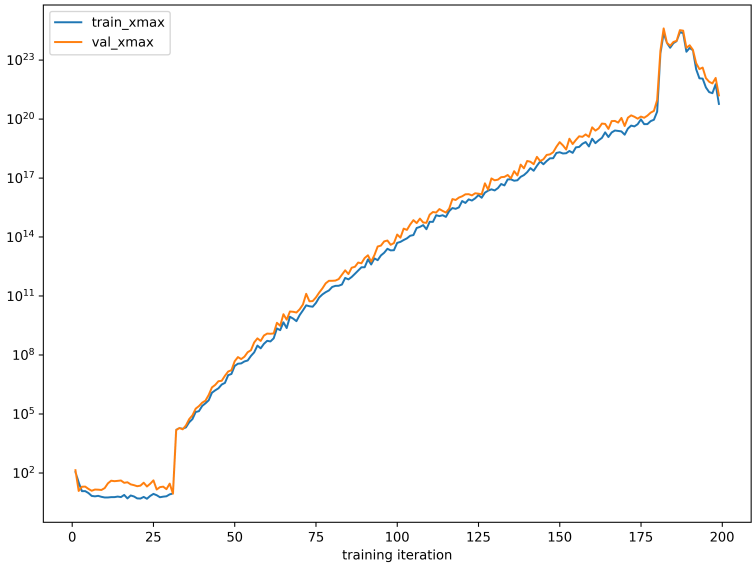


Abbildung 4.5: Verlauf des reduzierten Maximums von $x = \frac{w}{s}$ während des Training eines Netzes mit einem Wechsel der loss-Funktionen bei $k = 30$ unter Verwendung von T_1.

Das spiegelt sich auch in Abbildung 4.5 wieder. Mit dem fortschreitenden Training des Netzes mit der neuen loss-Funktion werden die Approximationen der Matrixelemente immer ungenauer, konkret werden die Ereignisgewichte stark unterschätzt. Tabelle 4.2 zeigt, dass das trainierte Netz auf dem Testdatensatz einen geringeren gain-Faktor erzielt, als das mit arsinh-loss trainierte.

4.5 Einführung eines Regularisierungsterms

Das Training des neuronalen Netzes mit dem gain-Faktors als loss-Funktion soll nicht nur den gain-Faktor des Netzes maximieren, weiterhin sollen auch die Ereignisgewichte sinnvoll approximiert werden. In der bisherigen Form ist das durch (4.16) noch nicht gegeben. Durch die dynamische Neuberechnung von $s_{\max}$ in jeder Epoche innerhalb des batch und da nur das Verhältnis $\frac{\langle s \rangle}{s_{\max}}$ für den loss relevant ist, kann die Vorhersage s um einen beliebigen Parameter erweitert werden. Das Ziel ist, die loss-Funktion um einen weiteren Term zu erweitern, der Unterschätzungen der Ereignisgewichte bestraft und dem Netz die Möglichkeit nimmt, den unbeschränkten Parameter frei zu wählen. So kann an dem gain-Faktor als loss-Funktion festgehalten werden. Es wird im folgenden die Konstruktion eines Terms motiviert, mit dem die bisherige loss-Funktion multipliziert werden kann. Eine sinnvolle Definition für den Regularisierungsterm r ist

$$r := \left(1 + \left|\frac{\langle w \rangle - \langle s \rangle}{\langle s \rangle}\right|\right), \tag{4.17}$$

bestehend aus den Mittelwerten der Ereignis- und Surrogatgewichten w, bzw. s. Wenn die Surrogatgewichte s systematisch unterschätzt werden, dann ist auch

$$\langle w \rangle \geq \langle s \rangle \Longrightarrow \frac{\langle w \rangle}{\langle s \rangle} \geq 1. \tag{4.18}$$

Die Abweichung der Mittelwerte $|\langle w \rangle - \langle s \rangle|$ dient als Maß für die Abweichung der Surrogatgewichte s von w. Werden die Vorhersagen des Modells genauer, dann

$$\langle s \rangle \longrightarrow \langle w \rangle \Longrightarrow |\langle w \rangle - \langle s \rangle| \longrightarrow 0 \tag{4.19}$$

$$\Longrightarrow r \longrightarrow 1, \tag{4.20}$$

womit der Regularisierungsterm starke Abweichungen „bestraft“ und für kleine Abweichungen verschwindet, wodurch der loss dann nur durch den reziproken gain-Faktor bestimmt wird. Mit dem gefundenen Regularisierungsterm wird der loss nach dem Wechsels weiter erhöht, sodass Gleichung (4.15) nicht ausreicht, um den Sprung zu korrigieren, da die Regularisierung nicht berücksichtigt wird. Analog zu Abschnitt 4.4.2 kann ein weiterer Übergangsfaktor T_2 konstruiert werden:

$$l_{\text{val, arsinh}}(k) \stackrel{!}{=} \mathrm{T}_2 \cdot \mathrm{T}_1 \cdot l_{\text{gain}}(k+1) \cdot r(k+1) \tag{4.21}$$

$$\Longrightarrow \mathrm{T}_2 = \frac{l_{\text{val, arsinh}}(k)}{\mathrm{T}_1 \cdot l_{\text{val,gain}}(k+1) \cdot r(k+1)}. \tag{4.22}$$

Mit Gleichung 4.15 und der Abschätzung $l_{\text{val,gain}} \approx (\text{gain}_{\text{val}})^{-1}$, die schon in Abschnitt 4.4.2 verwendet wurde, ergibt sich:

$$\mathrm{T}_2 = \frac{\mathrm{T}_1 \cdot \text{gain}_{\text{val}}(k+1)}{\mathrm{T}_1 \cdot \text{gain}_{\text{val}}(k) \cdot r(k+1)} \tag{4.23}$$

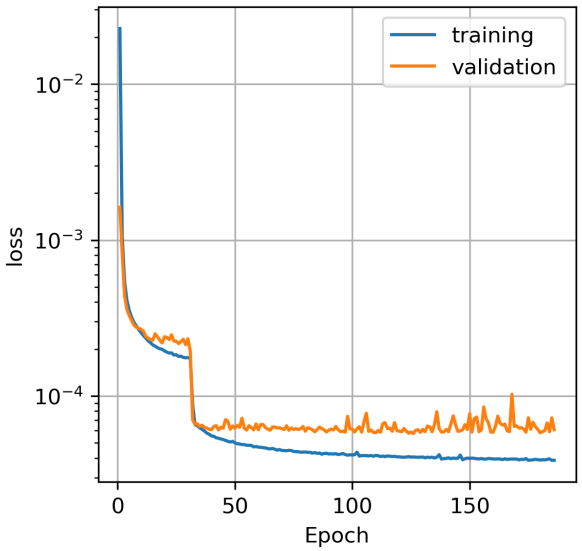
$$= \frac{\text{gain}_{\text{val}}(k+1)}{\text{gain}_{\text{val}}(k) \cdot r(k+1)} \tag{4.24}$$

$$\approx \frac{1}{r_{\text{val}}(k+1)}. \tag{4.25}$$

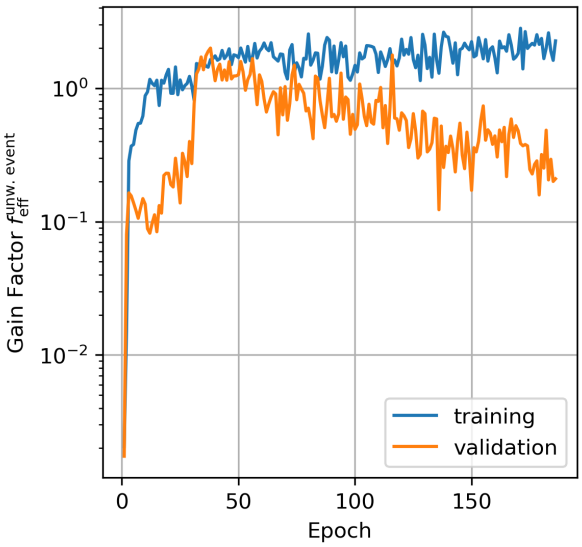
Implementiert man die Regularisierung als eine weitere Metrik, kann der Wert von $r_{\text{val}}(k+1)$ verwendet werden. Nun kann $r_{\text{val}}(k+1)$ erst nach Epoche $k+1$ berechnet werden, sodass dieser zweite Faktor T_2 erst dann an die loss-Funktion angefügt werden kann. Die loss-Funktion l hat mit diesen Veränderungen die Form

$$l = \mathrm{T}_1 \cdot \mathrm{T}_2 \cdot l_{\text{gain}} \cdot r, \tag{4.26}$$

wobei T_2 aus den benannten Gründen erst ab Epoche $k+2$ verwendet wird.



a) loss in logarithmischer Darstellung.



b) gain-Faktor in logarithmischer Darstellung.

Abbildung 4.6: Wechsel der loss-Funktion nach Epoche 30 unter Verwendung der Regularisierung r, sowie der Übergangsfaktoren T_1 und T_2.

Der Wechsel der loss-Funktion nach Epoche 30 ist in Abbildung 4.6a an dem auftretenden Sprung klar zu erkennen. Während der Validierungs-loss nach dem Wechsel annähernd konstant verläuft, verringert sich der Trainings-loss kontinuierlich. Der wachsende Abstand zwischen den Graphen ist als Overtraining zu interpretieren. Nur in den Epochen nach dem Wechsel liegen beide Größen bei ähnlichen Werten. In Abbildung 4.6b ist die Verbesserung des gain-Faktors unmittelbar nach dem loss-Wechsel zu erkennen. Über Epoche 50 hinaus liegen beide Graphen über einem gain-Faktor von $f_{\text{eff}} > 1$. Im weiteren Verlauf wächst der Trainings-gain-Faktor kontinuierlich an, während der Validierungs-gain-Faktor abfällt. Beide Graphen sind Oszillationen unterworfen.

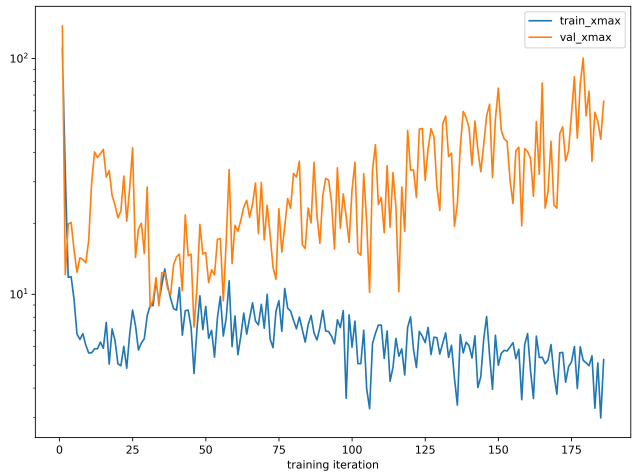


Abbildung 4.7: Verlauf des reduzierten Maximums von $x = \frac{w}{s}$ während des Training eines Netzes mit einem Wechsel der loss-Funktionen bei $k = 30$ unter Verwendung der Regularisierung.

Wie in Abbildung 4.7 zu sehen, werden die Gewichte durch die Verwendung des Regularisierungstermes in der loss-Funktion in diesem Lauf maximal um den Faktor 10^2 unterschätzt, eine deutliche Verbesserung gegenüber dem vorherigen Versuch. An dieser Stelle sei auf die Parallelen in den Abbildungen 4.6b und 4.7 hingewiesen. In beiden Fällen haben die Graphen für Training und Validierung in den ersten Epochen nach dem Wechsel der loss-Funktion ähnliche Werte und weisen danach dasselbe Verhalten auf. Das Netz verbessert sich auf den Trainingsdaten, was sich im Ansteigen des gain-Faktors und Absinken von $x_{\max}$ widerspiegelt. Auf den Validierungsdaten lässt sich ein umgekehrtes Verhalten beobachten, was ein eindeutiges Zeichen dafür ist, dass das Netz zu lange trainiert hat.

loss	gain-Faktor
arsinh	1.172
$l_{\text{nur gain}}$	1.018
$l_{\text{gain mit r}}$	1.595

Tabelle 4.2: gain-Faktoren für das Training der Netze mit den unterschiedlichen loss-Funktionen. Die gain-Faktoren ergeben sich unter Verwendung der Netze mit dem jeweilig geringsten Validierungs-loss, ausgewertet auf dem Test-Datensatz.

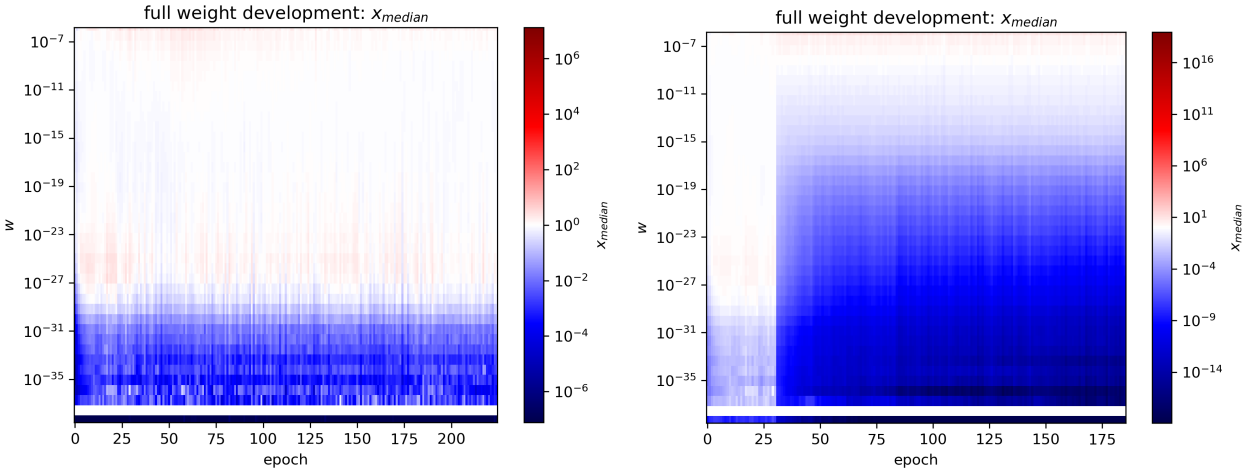
Das Netz, welches mit dem gain-Faktor unter Verwendung der Regularisierung trainiert wurde, liefert die größte Verbesserung im Umgewichtungsprozess. Für die in 4.2 aufgelisteten Werte wurde der Test-Datensatz von den trainierten Netzen ausgewertet. Danach wurde mithilfe der getroffenen Vorhersagen für jedes der Netze der gain-Faktor für den vollständigen Datensatz bestimmt. Die ermittelten gain-Faktoren müssen deshalb nicht mit dem je batch auf den Trainingsdaten berechneten und danach über alle batch gemittelten loss übereinstimmen.

Aufgrund der guten Performance wird an der Regularisierung festgehalten, da, auch unter Berücksichtigung von Abbildung 4.7, die Vorhersage der Gewichte mit Regularisierung genauer erfolgte. Aus dem Vergleich von 4.6b mit 4.6a ist zu erkennen, dass das durch early stopping gewählte Modell mit dem geringsten Validierungs-loss nicht zwangsläufig den besten gain-Faktor besitzt. Deshalb wird in den folgenden Trainingsläufen anstatt des Validierungsverustes der Validierungs-gain-Faktor als Metrik für das Abbrechen des Lernprozesses durch early stopping gewählt.

4.6 Effekte des gain-Faktors auf den Lernprozess des neuronalen Netzes

Trotz der Definition der passenden Übergangsfaktoren T_1 und T_2 ist in Abbildung 4.6a nach Epoche $k = 30$ ein deutlicher negativer Sprung zu erkennen. Dieser Sprung findet direkt in der Epoche nach dem Wechsel der loss-Funktionen statt. Relativ zu diesem Sprung findet im weiteren Verlauf des Trainings kaum weitere Verbesserungen statt.

Aus Abbildung 4.8a ist zu erkennen, dass der arsinh-loss über viele Größenordnungen hin-



a) x_{med} mit arsinh-loss.

b) x_{med} mit gain-Faktor als loss-Funktion ab Epoche 30.

Abbildung 4.8: x_{med} im Trainingsverlauf. In jeder Epoche wurde die Vorhersage des Netzes mit dem Test-Datensatz bestimmt. Die Surrogatgewichte s wurden nach den Zielwert w in Bins einsortiert und mittels der sortierten Paare wurde das jeweilige x berechnet. Die Bins sind entlang der Ordinatenachse sortiert. Für jeden der Bins wurde aus den vorliegenden x der Median x_{med} berechnet und entsprechend der Farbskala an der rechten Seite der Diagramme dargestellt.

weg die Gewichte der Ereignisse gut vorhersagt. Im Bereich von 10^{-7} bis 10^{-27} liegt in jedem Bin das Verhältnis aus Vorhersage und Surrogat bei $x_{\text{med}} \approx 1$. Erst bei Ereignisgewichten $w < 10^{-27}$ ist an der Färbung des Bereiches eine Überschätzung der Gewichte zu erkennen.

Bis zum Wechsel in Epoche 30 ist selbiges Verhalten auch in 4.8b zu beobachten. Mit dem Wechsel der loss-Funktion verändert sich die Vorhersage des neuronalen Netzes schlagartig. Das Netz scheint aus dem kollektiven Überschätzen der Ereignisgewichte einen Gewinn, eine Verringerung des loss, zu ziehen. Nur für große Gewichte im Bereich von 10^{-7} bis 10^{-11} werden die Gewichte im Mittel richtig vorhergesagt.
Interessant ist, dass das Netz nicht so verändert wurde, dass für jedes Ereignis dasselbe Gewicht vorhergesagt wird, sondern dass die Gewichte relativ zu einem Fixpunkt mit sinkenden w um einen anwachsenden Faktor überschätzt werden. Betrachtet man in diesem Kontext Abbildung (2.2) ist der Bereich in dem die Gewichte am besten vorhergesagt werden, also $x = \frac{w}{s} \approx 1$, genau der Bereich in dem sich die meisten Ereignisse befinden. Um einen geringen l_{gain} zu erreichen, scheint eine gute Vorhersage in diesem Bereich auszureichen, um den gain-Faktor für das gesamte Netz zu erhöhen.
Das Überschätzen von s führt dazu, das im ersten Umgewichtungsschritt mehr Ereignisse mit sonst geringen Ereignisgewichten w übernommen werden. Diese Ereignisse erhalten deshalb ein sehr kleines Übergewicht x und damit ist die Wahrscheinlichkeit, dass sie den zweiten Umgewichtungsschritt passieren können extrem gering, sodass sie aussortiert werden. Das bewirkt, dass hauptsächlich Ereignisse mit hohen Ereignisgewichten w den *unweighting*-Algorithmus 2 passieren.
Die starke Veränderung in 4.8b nach Epoche 30 zeigt, dass die internen Gewichte $g_{n_a m_b}$ hauptsächlich in dieser Epoche verändert und auf den neuen loss angepasst werden. Im weiteren Trainingsverlauf verändert sich die Vorhersage zwar, die größte Verbesserung zeigt sich dennoch in der Epoche des Wechsel der loss-Funktion.

5 Optimierung der Hyperparameter

Mit der Einführung des gain-Faktors und dem damit verbundenen Wechsel der loss-Funktionen wird dem Netz ein neuer Hyperparameter hinzugefügt, die Epoche des Wechsels k_{change}. In den vorangegangenen Betrachtungen in Abschnitt 4.4 wurde dieser Parameter auf $k_{\text{change}} = 30$ gesetzt. Im Folgenden soll untersucht werden, wie durch die gezielte Variation ausgewählter Hyperparameter der Gewinn durch die Verwendung des Surrogates im Umgewichtungsprozess gesteigert werden kann. Die Software *OPTIMA* wird genutzt, um mehrere Netze parallel zu trainieren. Dafür muss in *OPTIMA* der *search space* definiert werden. Den festen Hyperparametern werden im *search space* ihre diskreten Werte vorgegeben. Für die zu variierenden Hyperparameter können Intervalle oder Listen diskreter Werte definiert werden. *OPTIMA* ermöglicht es, im Modus der Hyperparameteroptimierung, mehrere Netzwerke parallel zu trainieren, wobei für die variablen Parameter zufällige Werte aus den vorgegebenen Suchbereichen ausgewählt werden.

5.1 Variation der batch-Größe

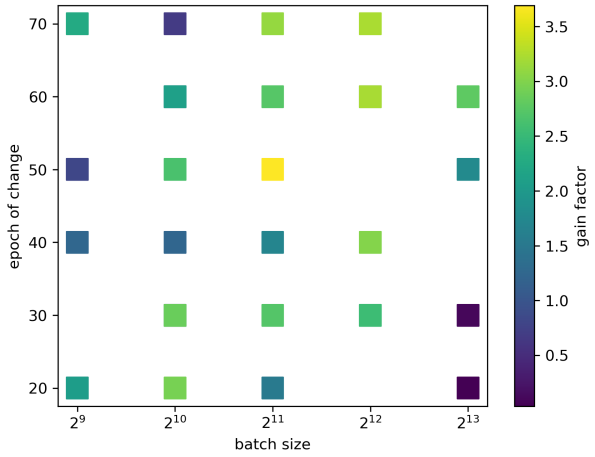
Während der bisherigen Betrachtungen waren innerhalb eines batch 512 Ereignisse zusammengefasst. Diese Wahl eignet sich für das Training mit der arsinh-loss-Funktion [19], muss deshalb aber nicht zwangsläufig für den gain-Faktor geeignet sein. Aus den Gleichungen (4.4) sowie (4.7), (4.8) und (4.9) ergibt sich, dass für die Berechnung des loss Mittelwerte und reduzierte Maxima benötigt werden, die mithilfe der Ereignisse innerhalb eines batch berechnet werden. Dabei ist vor allem die Ermittelung des reduzierten Maximums anfällig gegenüber Ausreißern, also einzelnen Ereignissen, deren Gewicht w um eine oder mehrere Größenordnungen größer ist, als das Gewicht der restlichen Ereignisse. Die Einteilung der Trainingsdaten in batch mit einer größeren Ereigniszahl erscheint deshalb sinnvoll. Dadurch verringert sich die Anzahl der Korrekturschritte, in denen die internen Gewichte des neuronalen Netzes angepasst werden können. Das Netz würde sich innerhalb der gleichen Anzahl an Trainingsiterationen weniger stark verändern. Der Wechsel der loss-Funktionen macht es notwendig, dass das neuronale Netz zum Zeitpunkt des Wechsels die Matrixelemente der Ereignisse zum einen ansatzweise gut vorhersagen kann und andererseits flexibel genug ist, sich auf die neue loss-Funktion einzustellen. Deshalb wird die Abhängigkeit des gain-Faktors von der Wahl der Größe des batch und der Epoche des loss-Wechsels *epoch of change* untersucht.

Die vorgegebenen diskreten Werte des Suchbereiches sind in Tabelle 5.1 aufgelistet. Als Mo-

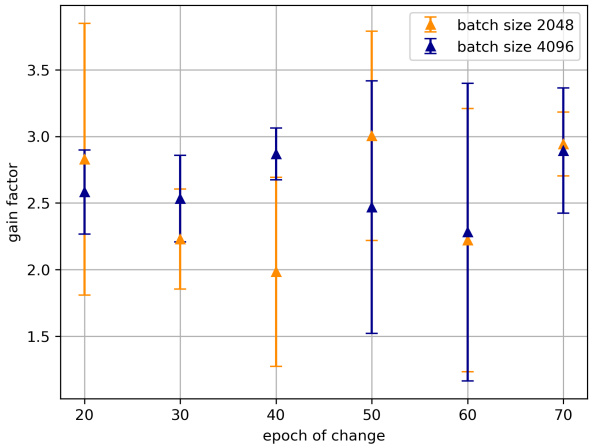
Parameter	Werte
k_{change}	{20, 30, 40, 50, 60, 70}
Ereignisse pro batch	{512, 1024, 2048, 4096, 8192}

Tabelle 5.1: Suchbereiche für die Parameter k_{change} und batch-Größe.

nitorvariable wird an dieser Stelle nicht der loss auf den Validierungsdaten l_{val}, sondern der gain-Faktor gain_{val} verwendet. Weiterhin wurde mit der in Gleichung (4.26) definierten loss-Funktion trainiert. Das soll dem in Abbildung 4.6b auftretenden Absinken des gain-Faktors entgegenwirken. Aufgrund der großen Schwankungen im gain-Faktor wird das Abbruch-Intervall I des early stopping auf $I = 60$ Epochen gesetzt, damit das Training nicht zu früh beendet wird und langfristige Verbesserungen des gain-Faktors möglich sind.



a) gain-Faktor des besten Netzes ausgewertet auf den Validierungsdaten.



b) mittlere gain-Faktoren mit ihrer Standardabweichung für die aussichtsreichen Größen des batch von 2048 und 4096.

Abbildung 5.1: Optimierung der batch-Größe bei neuronalen Netzen mit 4 hidden layer.

Die in Abbildung 5.1a dargestellten gain-Faktoren sind im Bereich der betrachteten Wechselepochen für 2048 und 4096 Ereignisse pro batch am höchsten. Die Daten dieser Netze sind in A.3 zu finden. Die Konfiguration (Ereignisse pro batch, k_{change}) = (2048, 50) erzielt mit $f_{\text{eff}} = 3.7$ den größten gain-Faktor. Die genauere Betrachtung der Ereigniszahl pro batch in Abbildung 5.1b zeigt deutliche Schwankungen in den erreichten gain-Faktoren. Die Netze wurden bis zum Abbruch durch early stopping trainiert.

Für jede Kombination (Ereignisse pro batch, k_{change}) wurde das arithmetische Mittel der erreichten gain-Faktoren ermittelt, sowie unter Verwendung der Varianz die statistische Unsicherheit abgeschätzt. Die Daten der Netze sind in A.2 zu finden. Dabei ist auf die große

Streuung der erreichten gain-Faktoren hinzuweisen. Da keine weiteren Hyperparameter variiert werden, sind diese Unterschiede auf die Variation der Startwerte der internen Gewichte $g_{n_a m_b}$ beim Zusammensetzen des Modells, sowie auf die statistischen Fluktuationen des gain-Faktors während des Trainings, zurückzuführen. Für den folgenden Optimierungsschritt wurde die Kombination (Ereignisse pro batch, k_{change}) = (2048, 50) gewählt, da diese im Mittel den größten gain-Faktor bringt.

5.2 Variation der Tiefe

Die Tiefe der bisher untersuchten Netze wird durch die Anzahl der hidden layer, die Schichten in denen die Verarbeitung der Eingabedaten stattfindet, bestimmt. Die Veränderung dieser Tiefe beeinflusst die Komplexität der Operationen, die von dem Netz durchgeführt werden kann. Durch die Variation der Tiefe des Netzes soll der von den Netzen erzielte gain-Faktor weiter erhöht werden. Dabei ist zu überprüfen, ob die bisherige Wahl von 4 hidden layer beibehalten werden kann, oder zugunsten eines höheren gain-Faktors verändert werden sollte. In Abbildung 5.2 sind für jeden gewählten Wert des Hyperparameters hidden layer große

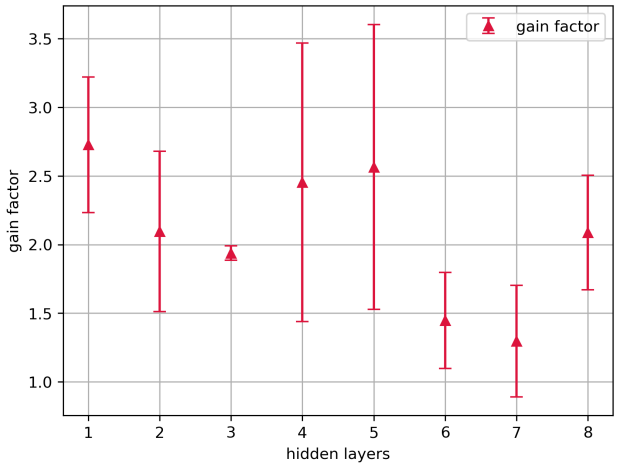


Abbildung 5.2: gain-Faktoren und statistische Unsicherheiten bei variierender Tiefe der Netze. Die Daten der Netze sind in A.1 zu finden.

Schwankungen zu erkennen. Dabei sind die geringsten gain-Faktoren entgegen der geäußerten Vermutung bei Tiefen von 6 und 7 hidden layer zu finden. Die größten Gewinne werden bei 1, 4 und 5 hidden layer erreicht, wobei die Wahl von 4 oder 5 hidden layer durch große statistische Unsicherheiten von $\Delta f_{\text{eff}} = 1$ dominiert wird. Aufgrund der großen Schwankungen erreichen die Netzwerke gain-Faktoren von $f_{\text{eff}} > 3.5$, siehe Tabelle A.1. Eine einfache Abhängigkeit der Performance des Netzes von der Tiefe ist bei den untersuchten Werten der Hyperparameter nicht zu finden.

6 Zusammenfassung

Die moderne Teilchenphysik stützt sich nicht nur auf die großen Mengen erzeugter Messdaten aus Beschleunigerexperimenten, sondern benötigt zusätzlich Daten simulierter Teilchenkollisionen. Die große Anzahl der auszuwertenden Ereignisse, die durch Monte-Carlo-Methoden erzeugt wurden, macht die effiziente Nutzung der verfügbaren Rechenkapazitäten notwendig. Dabei ist der Einsatz neuronaler Netze in *unweighting* Verfahren erfolgversprechend.
In dieser Arbeit wurden neuronale Netze als Surrogate für Matrixelementberechnungen in einem zweistufigen *unweighting*-Algorithmus verwendet. Die Optimierung dieser Netze soll in einer verbesserten Performance des *unweighting*-Algorithmus resultieren. Basierend auf dem gain-Faktor, der den Gewinn an Rechenzeit beschreibt, wurde eine neue loss-Funktion konstruiert, um maximale gain-Faktoren zu erreichen.
Durch die Aufteilung des Lernprozesses in zwei Phasen, wobei in der ersten Phase mit einer modifizierten MSE-loss-Funktion und in der zweiten Phase mit der auf dem gain-Faktor basierenden loss-Funktion trainiert wurde, konnte eine Steigerungen des gain-Faktors erreicht werden. In der anschließende Variation der Hyperparameter batch *size*, *epoch of change* und der Zahl der hidden layer wurde für die neue loss-Funktion mit (batch, *epoch of change*, hidden layer)=(2048, 50, 5) die beste Konfiguration gefunden, die auf den Validierungsdaten einen gain-Faktor von $f_{\text{eff}} = 2.6 \pm 1.0$ ermöglicht. Im Mittel verbessert sich der gain-Faktor gegenüber der vorherigen loss-Funktion um den Faktor 2.
Die untersuchten Hyperparameter sind dabei eine Auswahl der Parameter, die eng mit dem gain-Faktor und der Methode des loss-Funktion-Wechsels verknüpft sind. Abgesehen von den hier betrachteten Hyperparametern, lohnt sich die weitere Untersuchung anderer Parameter, wie die Zahl der Neuronen pro *layer*, die Konfiguration des *ADAM* Optimierers, die explizite Wahl der Lernrate, aber auch die Wahl des ε_{max} zur Bestimmung der reduzierten Maxima.
Während des parallelen Trainings hat sich gezeigt, das Netzwerke mit identischen Hyperparametern große Unterschiede in ihren erreichten gain-Faktoren aufweisen. Der gain-Faktor bewirkt als loss-Funktion in den ersten Epochen nach dem loss-Wechsel die stärksten Veränderungen an der Performance der neuronalen Netze. Im weiteren Trainingsverlauf scheint das Netz empfindlich auf die geringsten Variationen in den internen Gewichten zu reagieren, was von den Schwankungen in den Verläufen des loss widergespiegelt wird. Die Schwankungen führen beim Training einer Population von Netzen dazu, dass einzelne Netze den oben genannten Mittelwert deutlich überschreiten.

Zusätzlich zu der Streuung $g\ d \to e^-\ e^+\ g\ g\ g\ g\ d$ sollte die Verwendung der neuen loss-Funktion mit Ereignissen anderer Streuungen aus der Prozessgruppe Z + 5jets sowie für weitere Prozessgruppen, bei denen QCD-Dipole auftreten, getestet werden.

A Ergänzende Tabellen und Diagramme

hidden layer	Epochen	gain-Faktor
1	302	2.186509238
1	130	2.843817541
1	323	3.151392916
2	148	1.800795952
2	237	2.767476615
2	131	1.7188628
3	196	1.957820711
3	143	1.976909695
3	196	1.95802005
3	186	1.859289008
4	140	0.938905424
4	174	3.416233944
4	143	2.70080484
4	156	2.346866933
4	140	0.938905424
4	114	1.205516443
4	198	4.271131423
4	241	3.690990293
4	165	2.879852552
4	143	2.70080484
4	150	2.380668968
4	156	2.346866933
4	156	2.082118764
5	155	1.539811835
5	131	1.586470579
5	161	2.463903809
5	194	3.815628816
5	182	3.420791571
6	210	1.073088025
6	162	1.502607023
6	113	1.765193907
7	117	1.103533514
7	143	1.763357433
7	175	1.023080701
8	275	1.65598556
8	124	2.459238128
8	138	2.013287699
8	119	2.579580044
8	120	1.732081616

Tabelle A.1: Resultate der Netze bei Variation der hidden layer mit 128 Neuronen, $k_{\text{change}} = 50$, 2048 Ereignissen pro batch, $I = 60$ und Validierungs-gain-Faktor als Monitorvariable für Abbildung 5.2

k_{change}	Epochen	gain-Faktor
20	102	1.526950679
20	268	3.895446223
20	177	2.578781783
20	300	3.312684268
30	199	2.344226171
30	101	1.982278431
30	118	1.884090738
30	243	2.708118941
40	151	1.775189945
40	124	1.698643633
40	178	3.024345985
40	102	1.434637897
50	198	4.271131423
50	241	3.690990293
50	165	2.879852552
50	143	2.70080484
50	150	2.380668968
50	156	2.346866933
50	156	2.082118764
50	231	3.690990293
60	175	2.722347753
60	216	2.65858457
60	178	2.545954478
60	91	0.457007061
60	175	2.722347753
70	187	2.773925104
70	184	3.112452924

a) 2048 Ereignisse pro batch, 4 hidden layer mit 128 Neuronen, $I = 60$ und Validierungs-gain-Faktor als Monitorvariable.

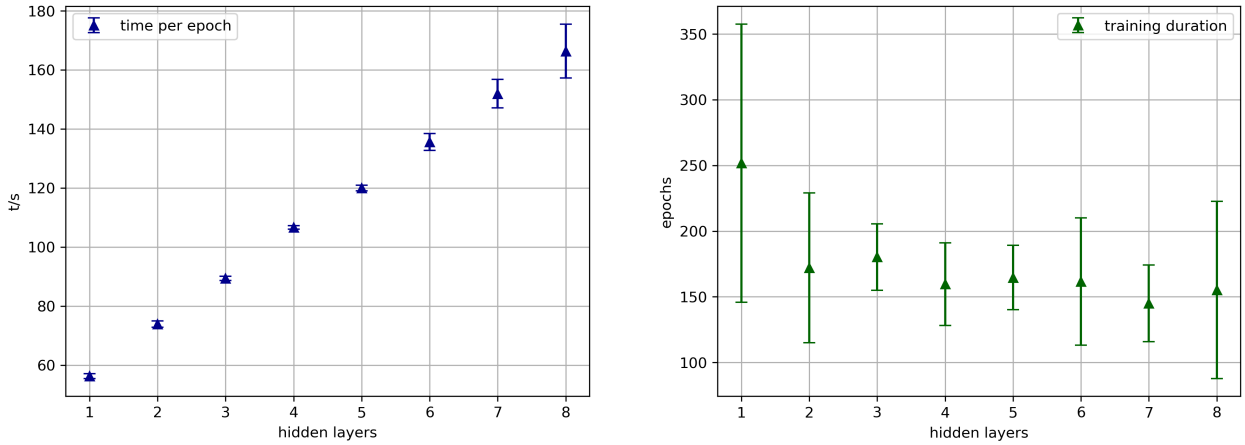
k_{change}	Epochen	gain-Faktor
20	189	2.756643511
20	286	2.772156461
20	183	2.218967736
30	163	2.8577144
30	222	2.850139657
30	302	2.203711049
30	302	2.203711049
30	201	2.545630425
40	217	2.970797065
40	179	3.015226896
40	173	2.901578459
40	237	2.584446799
50	263	3.582688449
50	250	3.265945981
50	217	3.230391523
50	144	2.298427875
50	172	2.119766826
50	203	1.893760061
50	142	0.899143117
60	281	3.118665211
60	206	2.782802282
60	259	2.765104383
60	96	0.458433807
60	151	1.351570525
60	192	3.213780692
70	228	2.56127859
70	234	3.225598348

b) 4096 Ereignisse pro batch, 4 hidden layer mit 128 Neuronen, $I = 60$ und Validierungs-gain-Faktor als Monitorvariable.

Tabelle A.2: Resultate der Netze für Abbildung 5.1b

Ereignisse pro batch	k_{change}	Epochen	gain-Faktor
512	20	98	1.288012468
512	20	188	2.060368806
512	40	105	1.242606491
512	50	116	0.596082546
512	50	120	0.819390046
512	70	150	2.257183486
1024	20	100	2.939188196
1024	30	98	0.826842619
1024	30	105	1.268584767
1024	30	104	2.857142857
1024	40	103	1.230118214
1024	50	197	2.644593129
1024	60	123	0.730438848
1024	60	163	1.573514602
1024	60	129	2.095118374
1024	70	71	0.674891342
2048	20	102	1.526950679
2048	30	243	2.708118941
2048	40	124	1.698643633
2048	50	231	3.690990293
2048	60	175	2.722347753
2048	70	184	3.112452924
4096	30	302	2.203711049
4096	30	201	2.545630425
4096	40	179	3.015226896
4096	60	151	1.351570525
4096	60	192	3.213780692
4096	70	234	3.225598348
8192	20	60	0.033815886
8192	30	91	0.091158534
8192	50	179	1.768440412
8192	60	384	2.787456446

Tabelle A.3: Resultate der Netze bei Variation der Ereignisse pro batch und *epoch of change*, 4 hidden layer mit 128 Neuronen, $I = 60$ und Validierungs-gain-Faktor als Monitorvariable für Abbildung 5.1a.



a) mittlere Trainingsdauer pro Epoche.

b) mittlere Zahl der Iterationen bis early stopping.

Abbildung A.1: Trainingsdauer der Netze bei Variation der hidden layer mit 128 Neuronen pro Schicht, $k_{\text{change}} = 50$, 2048 Ereignissen pro batch, $I = 60$ und Validierungs-gain-Faktor als Monitorvariable .

B Literatur

[1] The ATLAS Collaboration. „Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC“. In: *Physics Letters B* 716.1 (Sep. 2012), S. 1–29. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.020. arXiv: 1207.7214v2[hep-ex]. URL: http://dx.doi.org/10.1016/j.physletb.2012.08.020.

[2] The CMS Collaboration. „Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC“. In: *Physics Letters B* 716.1 (Sep. 2012), S. 30–61. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.021. arXiv: 1207.7235v2[hep-ex]. URL: http://dx.doi.org/10.1016/j.physletb.2012.08.021.

[3] Corinne Pralavorio. *Record data for the LHC in 2024*. URL: https://home.cern/news/news/accelerators/record-data-lhc-2024 (besucht am 24. 01. 2025).

[4] Katharina Danziger. „Efficiency Improvements in Monte Carlo Algorithms for High-Multiplicity Processes“. Technische Universität Dresden, 2020.

[5] Robert Mann. *An introduction to particle physics and the standard model*. CRC Press, Taylor & Francis Group, 2010. ISBN: 978-1-4200-8298-2 (Hardback).

[6] URL: https://home.cern/science/accelerators/large-hadron-collider (besucht am 20. 01. 2025).

[7] *LHC Run 3: physics at record energy starts tomorrow*. URL: https://home.cern/news/news/physics/lhc-run-3-physics-record-energy-starts-tomorrow (besucht am 21. 01. 2025).

[8] Aidan Randle-Conde. *Feynman diagram maker*. URL: https://www.aidansean.com/feynman/ (besucht am 23. 01. 2025).

[9] F James. „Monte Carlo theory and practice“. In: *Rep. Prog. Phys.* 43.1145 (1980).

[10] Enrico Bothmann u. a. *Event generation with SHERPA 3*. 2024. arXiv: 2410.22148v1[hep-ph].

[11] *SHERPA: Simulation of High-Energy Reactions of PArticles*. URL: https://sherpa-team.gitlab.io/monte-carlo.html#monte-carlo-info.

[12] Katharina Danziger u. a. „Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates“. In: *SciPost Phys.* 12.164 (2020). DOI: https://doi.org/10.21468/SciPostPhys.12.5.164. arXiv: 2109.11964v2[hep-ph].

[13] Dan Patterson. *Künstliche neuronale Netze: das Lehrbuch.* Prentice Hall Verlag GmbH, 1997. ISBN: 3-8272-9531-9.

[14] Prajit Ramachandran, Barret Zoph und Quoc V. Le. *Searching for Activation Functions.* 2017. arXiv: 1710.05941v2.

[15] Diederik P. Kingma und Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2015. arXiv: 1412.6980v9.

[16] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning.* http://www.deeplearningbook.org. MIT Press, 2016.

[17] *OPTIMA: an Optimization Platform for Tuning Input Variables and Model Parameters.* URL: https://gitlab.cern.ch/atlas-germany-dresden-vbs-group/optima.

[18] Erik Bachmann. „Polarization studies in same-sign W-boson pair production at the LHC with the ATLAS detector“. Technische Universität Dresden, 2023.

[19] T. Janßen u. a. „Unweighting multijet event generation using factorisation-aware neural Networks“. In: *SciPost Phys.* 15.107 (2023). DOI: https://doi.org/10.21468/SciPostPhys.15.3.107. arXiv: 2301.13562v2[hep-ph].

Glossar

loss

„Verlust“ oder „Fehler“, bezieht sich im Kontext der neuronalen Netze auf die Metrik, mit der Vorhersage und Zielwerte verglichen werden. iii, 2, 12–15, 17–32, 35, 36, 42

gain

„Gewinn“, bezeichnet den Rechenzeitgewinn durch Verwendung der Surrogat-Methode. iii, 1, 2, 17–33, 35, 37–40, 42

Surrogat

lat. *surrogatus* „Ersatz“. 1, 11, 12, 17, 19, 26, 29, 31, 35, 42

batch

Einheit zur Unterteilung des Trainingsdatensatzes. Diese werden während des Trainings sequenziell ausgewertet. 2, 14, 19, 20, 25, 28, 31–33, 35, 37–40, 42

hidden layer

„verborgene Schicht“, Schicht aus künstlichen Neuronen, die weder der Eingabe noch der Ausgabe dienen. 2, 12, 14, 20, 32, 33, 35, 37–40, 42

early stopping

Methode, mit der das Lernen eines neuronalen Netzes nach Erfüllen einer spezifischen Bedingung beendet wird. 15, 29, 32, 40, 42

Danksagung

Ich möchte an dieser Stelle einigen Personen meinen Dank aussprechen, die mich während des Schreibens dieser Arbeit unterstützt haben.
An erster Stelle möchte ich Frank Siegert und Tim Herrmann erwähnen, die mir die Arbeit an diesem spannenden Thema ermöglicht haben. Dabei möchte ich mich bei Tim für die Zeit bedanken, die er sich zur Beantwortung meiner Fragen und zum Korrekturlesen dieser Arbeit genommen hat.
Weiterhin danke ich Louis, Sophie und meinem Vater für ihre hilfreichen Kommentare am Text, sowie Lars, der meine Kaffee-Pausen durch intensive Tischkicker-Duelle in die Länge gezogen hat.

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit im Rahmen der Betreuung am Institut für Kern- und Teilchenphysik ohne unzulässige Hilfe Dritter verfasst und alle Quellen als solche gekennzeichnet habe.

Mathis Erik Schenker
Dresden, Januar 2025