TECHNISCHE
UNIVERSITÄT
DRESDEN

Anwendung von künstlichen neuronalen Netzen zur Regression am Beispiel des Diphoton-Prozesses

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

vorgelegt von

Andreas Weitzel
geboren am 10.08.1999 in Fulda

Institut für Kern- und Teilchenphysik
Fakultät Physik
Bereich Mathematik und Naturwissenschaften
Technische Universität Dresden
2021

Eingereicht am 25. Mai 2021

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

Zusammenfassung
Der differentielle Wirkungsquerschnitt für die Produktion von Photon-Paaren bei der Partonstreuung $q\bar{q} \to \gamma\gamma$ wird berechnet. Daraus wird mithilfe von Partondichtefunktionen der Wirkungsquerschnitt für den hadronischen Prozess $pp \to \gamma\gamma$ ermittelt. Die Anwendung von Methoden des Deep-Learning zur Näherung der differentiellen Wirkungsquerschnitte wird untersucht. Dabei bieten die exponentiellen Variationen der Wirkungsquerschnitte eine wesentliche Herausforderung. Weiterhin wird die Eignung und Anwendbarkeit von Transfer-Learning zur schnellen Adaption des Lernergebnisses an andere Partondichtefunktionen untersucht. Sowohl die Regression, als auch das Transfer-Learning, liefern gute Ergebnisse. Schließlich wird von Monte-Carlo-Methoden Gebrauch gemacht, um die differentiellen Wirkungsquerschnitte zu integrieren.

Abstract
The differential cross section for the production of photon-pairs in parton scattering $q\bar{q} \to \gamma\gamma$ is calculated. The result is applied to the hadronic process $pp \to \gamma\gamma$ using parton distribution functions. The application of deep learning methods to approximate the differential cross sections is investigated. Here, the exponential variations of the cross sections offer a significant challenge. The applicability and suitability of transfer learning to quickly adept the training outcome to other parton density functions is examined. The regression, aswell as transfer-learning, yield good results. Eventually, Monte-Carlo-Methods are used to integrate the differential cross sections.

Contents

1 Einleitung

Maschinelles Lernen (ML) ist ein Schlagwort und Konzept, das zwar schon lange im Umlauf ist, jedoch seit einiger Zeit extrem an Beliebtheit gewinnt. Auch in der Physik haben verschiedene Methoden bereits Einzug gehalten. Eine davon ist Deep-Learning, das einen Bereich des maschinellen Lernens bezeichnet, in dem künstliche neuronale Netze verwendet werden. In dieser Arbeit soll die Eignung neuronaler Netze zur Regression von differentiellen Wirkungsquerschnitten untersucht werden. Dies wird am Beispiel des nicht-resonanten Diphoton-Prozesses durchgeführt, dessen differentieller Wirkungsquerschnitt sowohl auf partonischer Ebene als auch auf hadronischer Ebene in führender Ordnung analytisch hergeleitet wird.

Die nicht-resonante Photon-Paar-Produktion in Proton Kollisionen stellt den Hintergrund zur Diphoton-Produktion mit resonanten Propagatoren dar, die durch den Zerfallskanal $H \to \gamma\gamma$ des Higgs-Bosons wesentlich zu dessen Nachweis [?, ?] beigetragen hat. Auch aktuell ist der Prozess interessant, so wird er dazu verwendet sehr massereiche Resonanzen, vorhergesagt von Theorien jenseits des Standardmodells, zu suchen [?]. Zur Simulation von Photon-Paar-Produktion werden numerische Methoden, wie der Event-Generator SHERPA [?], verwendet, die sehr rechenintensiv sein können. Machine-Learning-Algorithmen können im Vergleich effizienter sein, indem sie den Wirkungsquerschnitt, nach Vorarbeit eines rechnerisch anspruchsvollen Algorithmus, erlernen. Der Vorteil liegt hierbei darin, die aufwändigen numerischen Methoden zur Berechnung einer ausreichenden Anzahl an Phasenraumpunkten nur einmalig zu verwenden, um mit diesen den ML-Algorithmus zu trainieren und anschließend eine größere Zahl an Punkten zu generieren.

In chapter 2 wird mit der theoretischen Behandlung des Diphoton-Prozesses im Rahmen der Quanten-elektrodynamik begonnen, wobei Ausdrücke für den differentiellen Wirkungsquerschnitt des partonischen und hadronischen Prozesses analytisch hergeleitet werden. chapter 3 beschäftigt sich zunächst mit den Konzepten hinter Maschinellem Lernen und speziell Deep-Learning mit tiefen neuronalen Netzen (DNN). Am Ende des Kapitels werden die Grundlagen einer Monte-Carlo-Integration (MC-Integration) besprochen. Die Anwendung der DNN folgt in chapter 4, wobei zunächst die differentiellen Wirkungsquerschnitte des Diphoton-Prozesses genähert werden. Anschließend werden das Lernen von Gewichten zur Umgewichtung von Ergebnissen für unterschiedliche Sets von Partondichtefunktionen (PDF) und die Eignung von Transfer-Learning (TL) untersucht.

In dieser Arbeit verwendete Abkürzungen sind in Table A.8 zusammengefasst. Es werden durchweg natürliche Einheiten, sprich $\hbar = c = 1$, verwendet. Vektoren werden mit fett gedruckten Kleinbuchstaben (Bsp. $\mathbf{x}$) und Matrizen oder Tensoren mit fett gedruckten Großbuchstaben (Bsp. $\mathbf{W}$) notiert. Speziell Dreiervektoren werden mit einem Pfeil gekennzeichnet (Bsp. $\vec{p}$). Vierervektoren ergeben sich aus dem Kontext.
Unter `https://github.com/andiw99/Bachelor-Thesis` kann der gesamte Python-Code, der während dieser Arbeit entstanden ist, eingesehen werden. Hierbei sind alle Skripte zur Erzeugung der Diagramme im Ordner `Plotscripts` durchnummeriert zu finden. Alle mit ML in Verbindung stehenden Funktionen und Klassen sind in `ml.py` definiert. Analoges gilt für `MC.py`. Es wird TensorFlow [?] 2.4.1 genutzt, wobei die Skripte auch mit TensorFlow 2.3.1 getestet wurden.

2 Diphoton-Prozess

2.1 Matrixelement des partonischen Diphoton-Prozesses

Zunächst wird der differentielle Wirkungsquerschnitt des partonischen Diphoton-Prozesses, $q\overline{q} \rightarrow \gamma\gamma$ aus den Feynman-Regeln der Quantenelektrodynamik in führender Ordnung hergeleitet. Es werden hochrelativistische Quarks betrachtet, deren Ruhemasse vernachlässigt werden kann.

Abb. 2.1: Die Feynman-Diagramme führender Ordnung des Diphoton-Prozesses $q\overline{q} \rightarrow \gamma\gamma$ werden aufgestellt.

In Abb. 2.1 sind die Feynman Diagramme führender Ordnung gezeigt. Hieraus können die Matrixelemente aus Gl. 2.1 abgeleitet werden. Es werden die Notation $\gamma^\mu p_\mu = p$, die Mandelstam-Variablen[1] sowie $\epsilon_\mu(p_i) \equiv \epsilon_{\mu,\lambda_i}$ verwendet, um die Matrixelemente zu vereinfachen (siehe Gl. 2.2). λ_i beschreibt die Polarisation des Photons.

$$\begin{aligned}
\mathcal{M}_t &= \overline{\nu}\left(p_1\right)\left(-iQ_q e\gamma^\mu\right)\epsilon^*_\mu\left(p_3\right)\left(\frac{\gamma^\alpha\left(p_{1,\alpha}-p_{3,\alpha}\right)}{\left(p_1-p_3\right)^2}\right)\left(-iQ_q e\gamma^\nu\right)\epsilon^*_\nu\left(p_4\right)u\left(p_2\right)\\
\mathcal{M}_u &= \overline{\nu}\left(p_1\right)\left(-iQ_q e\gamma^\rho\right)\epsilon^*_\rho\left(p_4\right)\left(\frac{\gamma^\beta\left(p_{1,\beta}-p_{4,\beta}\right)}{\left(p_1-p_4\right)^2}\right)\left(-iQ_q e\gamma^\sigma\right)\epsilon^*_\sigma\left(p_3\right)u\left(p_2\right)
\end{aligned} \tag{2.1}$$

$$\begin{aligned}
\mathcal{M}_t &= -\frac{Q_q^2 e^2}{t}\left[\overline{\nu}(p_1)\gamma^\mu\epsilon^*_{\mu,\lambda_3}\left(p_1-p_3\right)\gamma^\nu\epsilon^*_{\nu,\lambda_4}u(p_2)\right]\\
\mathcal{M}_u &= -\frac{Q_q^2 e^2}{u}\left[\overline{\nu}(p_1)\gamma^\rho\epsilon^*_{\rho,\lambda_4}\left(p_1-p_4\right)\gamma^\sigma\epsilon^*_{\sigma,\lambda_3}u(p_2)\right]
\end{aligned} \tag{2.2}$$

Die Vierervektoren sind wie in Abb. 2.2 gewählt und in Gl. 2.3 aufgeführt. Die Mandelstam-

[1] $t = (p_1 - p_3)^2$, $u = (p_1 - p_4)^2$

Variablen ergeben sich zu Gl. 2.4.

$$p_1 = \begin{pmatrix} p \\ 0 \\ 0 \\ p \end{pmatrix} \quad p_2 = \begin{pmatrix} p \\ 0 \\ 0 \\ -p \end{pmatrix} \quad p_3 = \begin{pmatrix} p \\ \sin(\theta)p \\ 0 \\ \cos(\theta)p \end{pmatrix} \quad p_4 = \begin{pmatrix} p \\ -\sin(\theta)p \\ 0 \\ -\cos(\theta)p \end{pmatrix} \tag{2.3}$$

$$t = -4p^2 \cos^2\left(\frac{\theta}{2}\right) \quad \text{und} \quad u = -4p^2 \sin^2\left(\frac{\theta}{2}\right) \ . \tag{2.4}$$

Das totale Matrixelement wird durch Summation der Anteile des u- und t-Kanals berechnet:

$$\begin{aligned} \mathcal{M} = \mathcal{M}_u + \mathcal{M}_t &= \mathcal{F}\left[\overline{\nu}(p_1)\left(\frac{\Gamma_t}{a} + \frac{\Gamma_u}{b}\right)u(p_2)\right] \\ &= \mathcal{F}\left[\overline{\nu}(p_1)\Gamma u(p_2)\right] \ , \end{aligned} \tag{2.5}$$

wobei die Ersetzungen aus Gl. 2.6 gewählt wurden.

$$\begin{aligned} \Gamma_t = \gamma^\mu \epsilon^*_{\mu,\lambda_3}(p_1 - p_3)\gamma^\nu \epsilon^*_{\nu,\lambda_4} \quad &\text{und} \quad \Gamma_u = \gamma^\rho \epsilon^*_{\rho,\lambda_4}(p_1 - p_4)\gamma^\sigma \epsilon^*_{\sigma,\lambda_3} \\ \text{sowie} \quad \mathcal{F} = \frac{Q_q^2 e^2}{4p^2} \quad &\text{und} \quad \Gamma = \frac{\Gamma_t}{\cos^2\left(\frac{\theta}{2}\right)} + \frac{\Gamma_u}{\sin^2\left(\frac{\theta}{2}\right)} \\ \cos^2\left(\frac{\theta}{2}\right) = a \quad &\text{und} \quad \sin^2\left(\frac{\theta}{2}\right) = b \end{aligned} \tag{2.6}$$

Bei der Berechnung des gemittelten Quadrats des Betrages des Matrixelementes müssen die möglichen Anfangszustände der Quarks und Endzustände der Photonen berücksichtigt werden. Während die Endzustände eine Summe über mögliche Helizitäten s_3, s_4 und Polarisationen λ_3, λ_4 ergeben, können die Quarks drei verschiedene Farbzustände und jeweils zwei verschiedene Helizitäten annehmen, sodass die Anfangszustände einen Faktor 1/12 liefern:

$$\left\langle \mathcal{M}^2 \right\rangle = \frac{1}{12} \sum_{s_3,s_4} \sum_{\lambda_3,\lambda_4} \mathcal{M}^2 \ . \tag{2.7}$$

Um die Summe über die Helizitäten auszuführen, wird Casimirs Trick verwendet:

$$\sum_{s_3,s_4} \mathcal{M}^2 = \mathcal{F}^2 \sum_{s_3,s_4} \left[\overline{\nu}(p_1)\Gamma u(p_2)\right]\left[\overline{\nu}(p_1)\Gamma u(p_2)\right]^* = \mathcal{F}^2 \ \mathrm{Tr}\left[\Gamma p_2 \overline{\Gamma} p_1\right] \ , \tag{2.8}$$

wobei $\overline{\Gamma} = \gamma^0 \Gamma^\dagger \gamma^0 = \frac{\overline{\Gamma}_t}{a} + \frac{\overline{\Gamma}_u}{b}$ die Dirac-Adjungierte bezeichnet. Für die Dirac-adjungierten

$\overline{\Gamma}_t, \overline{\Gamma}_u$ ergibt sich:

$$\overline{\Gamma}_t = \gamma^\nu \epsilon_{\nu,\lambda_4}(p_1 - p_3)\gamma^\mu \epsilon_{\mu,\lambda_3} \quad \text{und} \quad \overline{\Gamma}_u = \gamma^\sigma \epsilon_{\sigma,\lambda_3}(p_1 - p_4)\gamma^\rho \epsilon_{\rho,\lambda_4} \ . \tag{2.9}$$

Gl. 2.8 wird damit zu:

$$\mathrm{Tr}\left[\Gamma p_2 \overline{\Gamma} p_1\right] = \mathrm{Tr}\left[\frac{1}{a^2}\Gamma_t p_2 \overline{\Gamma}_t p_1 + \frac{1}{ab}\Gamma_t p_2 \overline{\Gamma}_u p_1 + \frac{1}{ba}\Gamma_u p_2 \overline{\Gamma}_t p_1 + \frac{1}{b^2}\Gamma_u p_2 \overline{\Gamma}_u p_1\right] \ . \tag{2.10}$$

Bei Einsetzen von Gl. 2.10 in Gl. 2.7 ergeben sich Terme in folgendem Schema:

$$T_{ij} = \frac{1}{12}\sum_{\lambda_3,\lambda_4} \mathcal{F}^2 \ \mathrm{Tr}\left[\frac{1}{ij}\Gamma\left(i\right) p_2 \overline{\Gamma}\left(j\right) p_1\right] \quad \text{mit} \quad i,j \in \{a,b\} \ . \tag{2.11}$$

Hierbei wird $\Gamma(a) = \Gamma_t$ und $\Gamma(b) = \Gamma_u$ identifiziert. Zunächst wird in Gl. 2.12 der Fall $i = j$ evaluiert, wobei diverse Spur-Methoden und Identitäten der Dirac-Matrizen verwendet werden.

$$\begin{aligned}
T_{aa} &= \frac{\mathcal{F}^2}{12a^2}\sum_{\lambda_3,\lambda_4} \mathrm{Tr}\left[\gamma^\mu \epsilon^*_{\mu,\lambda_3}(p_1 - p_3)\gamma^\nu \epsilon^*_{\nu,\lambda_4} p_2 \gamma^{\nu'} \epsilon_{\nu',\lambda_4}(p_1 - p_3)\gamma^{\mu'} \epsilon_{\mu',\lambda_3} p_1\right] \\
&= \frac{\mathcal{F}^2}{12a^2}\sum_{\lambda_3,\lambda_4} \epsilon^{*\mu}_{\lambda_3}\epsilon^{\mu'}_{\lambda_3}\epsilon^{*\nu}_{\lambda_4}\epsilon^{\nu'}_{\lambda_4} \ \mathrm{Tr}\left[\gamma_\mu(p_1 - p_3)\gamma_\nu p_2 \gamma_{\nu'}(p_1 - p_3)\gamma_{\mu'} p_1\right] \\
&\overset{(2.15)}{=} \frac{\mathcal{F}^2}{12a^2}\ g^{\mu\mu'} g^{\nu\nu'} \ \mathrm{Tr}\left[\gamma_\mu(p_1 - p_3)\gamma_\nu p_2 \gamma_{\nu'}(p_1 - p_3)\gamma_{\mu'} p_1\right] \\
&= \frac{\mathcal{F}^2}{12a^2} \ \mathrm{Tr}\left[\gamma_\mu(p_1 - p_3)\gamma_\nu p_2 \gamma^\nu (p_1 - p_3)\gamma^\mu p_1\right] \\
&= \frac{\mathcal{F}^2}{3a^2} \ \mathrm{Tr}\left[(p_1 - p_3) p_2 (p_1 - p_3) p_1\right] \\
&= \frac{8\mathcal{F}^2}{3a^2}(p_3 \cdot p_2)(p_3 \cdot p_1)
\end{aligned} \tag{2.12}$$

Hierbei wurde die Vollständigkeitsrelation für reale Photonen (siehe Gl. 2.14) verwendet. Es folgt analog:

$$T_{bb} = \frac{8\mathcal{F}^2}{3b^2}(p_4 \cdot p_2)(p_4 \cdot p_1) \ . \tag{2.13}$$

$$\sum_{\lambda=1}^{2} \epsilon^\mu_\lambda \epsilon^{*\nu}_\lambda = -g^{\mu\nu} \tag{2.14}$$

Für $i \neq j$ ergibt sich:

$$\begin{aligned}
T_{ab} &= \frac{\mathcal{F}^2}{12ab} \ \mathrm{Tr}\left[\gamma_\mu(p_1 - p_4)\gamma_\nu p_2 \gamma^\mu (p_1 - p_3)\gamma^\nu p_1\right] \\
&= \frac{4\mathcal{F}^2}{3ab}\left[(p_1 \cdot p_2)\left[-2\,(p_1 \cdot p_4) + (p_3 \cdot p_4)\right] - (p_1 \cdot p_3)(p_2 \cdot p_4) + (p_2 \cdot p_3)(p_1 \cdot p_4)\right] \ .
\end{aligned} \tag{2.15}$$

und analog:

$$T_{ba} = \frac{4\mathcal{F}^2}{3ab}\left[(p_1\cdot p_2)\left[-2\,(p_1\cdot p_3)+(p_3\cdot p_4)\right]-(p_1\cdot p_4)(p_2\cdot p_3)+(p_1\cdot p_3)(p_2\cdot p_4)\right] \tag{2.16}$$

Beim Einsetzen der expliziten Vierervektoren aus Gl. 2.3 fällt auf, dass $T_{ab}+T_{ba}=0$. Die Summe über die Helizitäten und Polarisationen wurde nun ausgeführt, sodass Gl. 2.7 umgeschrieben werden kann zu:

$$\begin{aligned}\left\langle\mathcal{M}^2\right\rangle &= \frac{8}{3}\mathcal{F}^2\left(\frac{1}{a^2}(p_3\cdot p_2)(p_3\cdot p_1)+\frac{1}{b^2}(p_4\cdot p_2)(p_4\cdot p_1)\right)\\ &= \frac{2}{3}Q_q^4e^4\left[\frac{1-\cos^2(\theta)}{\cos^4\left(\frac{\theta}{2}\right)}+\frac{1-\cos^2(\theta)}{\sin^4\left(\frac{\theta}{2}\right)}\right]\\ &= \frac{4}{3}Q_q^4e^4\frac{1+\cos^2(\theta)}{\sin^2(\theta)} = \frac{4}{3}Q_q^4e^4\cosh(2\eta)\ .\end{aligned} \tag{2.17}$$

Hierbei ist $\eta = -\ln\left(\tan\left(\frac{\theta}{2}\right)\right)$ die Pseudo-Rapidität.

2.2 Differentieller Wirkungsquerschnitt des partonischen Prozesses

Mithilfe von Fermis goldener Regel kann aus dem Betragsquadrat des Übergangsmatrixelementes der Wirkungsquerschnitt berechnet werden. Im Schwerpunktsystem mit vernachlässigbaren Ruhemassen ergibt sich der Zusammenhang in Gl. 2.18, wobei $\mathrm{d}\Omega = \sin(\theta)\mathrm{d}\theta\mathrm{d}\varphi$ das Raumwinkelelement und $s=(p_1+p_2)^2$ das Quadrat der Schwerpunktsenergie bezeichnet.

$$\sigma = \frac{1}{64\pi^2 s}\int\left\langle\mathcal{M}^2\right\rangle\mathrm{d}\Omega = \frac{1}{32\pi s}\int\left\langle\mathcal{M}^2\right\rangle\sin(\theta)\mathrm{d}\theta \tag{2.18}$$

Für den differentiellen Wirkungsquerschnitt $\sigma\theta$ ergibt sich Gl. 2.19, wobei ein Symmetriefaktor $\frac{1}{2}$ hinzukommt, da die beiden Photonen im Endzustand identisch sind.

$$\sigma\theta = \frac{1}{2}\cdot\frac{Q_q^4e^4}{24\pi s}\frac{1+\cos^2(\theta)}{\sin(\theta)} \tag{2.19}$$

Mithilfe der Kettenregel lässt sich der differentielle Wirkungsquerschnitt in Abhängigkeit von η bestimmen:

$$\sigma\eta = \theta\eta\sigma\theta = \frac{Q_q^4e^4}{48\pi s}(1+\tanh^2(\eta))\ . \tag{2.20}$$

2.3 Hadronischer Diphoton Prozess

Aufgrund des *Confinement* kommen Quarks nicht als freie Teilchen vor, sodass lediglich der hadronische Prozess $pp \to \gamma\gamma$ beobachtet werden kann. Die Protonen besitzen hierbei zwei up-Quarks
und ein down-Quark als Valenzquarks sowie verschiedene Quark-Antiquark-Paare als Seequarks. Prallen zwei Protonen mit hohen Energien aufeinander, wird die Substruktur des Protons aufgelöst und die Konstituenten des Hadrons können miteinander interagieren. Bei diesen Interaktionen können die Quarks als quasi-freie Teilchen betrachtet werden.

Abb. 2.2: Kinematik der Stoßprozesse im Labor- und Schwerpunktssystem

Das Schwerpunktsystem der Quarks unterscheidet sich im Allgemeinen vom Schwerpunktsystem der Protonen (Laborsystem). Der Impulsbruchteil eines Quarks innerhalb eines Hadrons ist nicht fest definiert, sodass ihm zunächst ein unbestimmter Bruchteil x des Gesamtimpulses zugeordnet wird. Das Proton wird nun in einem System betrachtet, indem es eine sehr hohe Energie $E \gg m_p$ besitzt, sodass seine Ruhemasse vernachlässigt und sein Viererverktor als $\mathbf{p}_p = (E, 0, 0, E)$ geschrieben werden kann. Im vorliegenden System können die Transversalimpulse der Partonen vernachlässigt werden, sodass ihr Vierervektor sich zu Gl. 2.21 ergibt.

$$\mathbf{p}_q = (xE, 0, 0, xE) = x\mathbf{p}_p \ . \tag{2.21}$$

Findet bei einer Interaktion ein Impulsübertrag $\mathbf{q}$ statt, so geht der Vierervektor des Partons in $x\mathbf{p}_p \to (x\mathbf{p}_p + \mathbf{q})$ über. Durch Einsetzen der invarianten Masse beider Zustände (siehe Gl. 2.22) ineinander kann nach dem Impusbruchteil x aufgelöst werden (siehe Gl. 2.23).

$$\left(x\mathbf{p}_p\right)^2 = m_q^2 \quad \text{und} \quad \left(x\mathbf{p}_p + q\right)^2 = \left(x\mathbf{p}_p\right)^2 + 2x\mathbf{p}_p \cdot \mathbf{q} + \mathbf{q}^2 = m_q^2 \tag{2.22}$$

$$2x\mathbf{p}_p \cdot \mathbf{q} + \mathbf{q}^2 = 0 \quad \Rightarrow \quad x = \frac{-\mathbf{q}^2}{2\mathbf{p}_p \cdot \mathbf{q}} \tag{2.23}$$

Das x in Gl. 2.23 identifiziert sich als die Bjorken-Skalenvariable. Sie repräsentiert bei hohen Proton-Impulsen den Impulsbruchteil, den ein Parton im Proton trägt.
Da es nicht möglich ist, die Impulsbruchteile vor der Reaktion zu kennen, wird der Prozess im Schwerpunktsystem der kollidierenden Protonen beschrieben. Hier folgt der Impulsbruchteil x einer Wahrscheinlichkeitsverteilung, der *Partondichtefunktion* (PDF) $f_i(x, Q^2)$ des Protons. Diese PDFs beschreiben die Wahrscheinlichkeitsdichte, bei einer Energieskala $Q^2 = -\mathbf{q}^2$ das

entsprechende Parton i mit dem Impulsbruchteil x zu finden. Da sie nicht aus ersten Prinzipien abgeleitet werden können, müssen sie experimentell bestimmt werden.

Die Partondichtefunktionen können genutzt werden, um einen Ausdruck für den totalen Wirkungsquerschnitt $pp \to \gamma\gamma$ zu finden. Ist der totale Wirkungsquerschnitt eines partonischen Prozesses zwischen den Partonen i und j bei den festgelegten Impulsbruchteilen x_1 und x_2 und der Energieskala Q^2 bekannt (genannt $\tilde{\sigma}_{i,j}(x_1, x_2, Q^2)$), dann kann mithilfe der PDF der Wirkungsquerschnitt $\sigma_{i,j}$ für die Reaktion der Partonen i und j bei dem Zusammenstoß von zwei Protonen berechnet werden (siehe Gl. 2.24).

$$\sigma_{i,j} = \int f_i(x_1, Q^2) f_j(x_2, Q^2) \tilde{\sigma}_{i,j}(x_1, x_2, Q^2) \mathrm{d}x_1 \mathrm{d}x_2 \tag{2.24}$$

Der totale Wirkungsquerschnitt ergibt sich dann als Summe aller möglichen $\sigma_{i,j}$, wobei im Diphoton-Prozess lediglich $i = q = \overline{j}$ beitragen:

$$\sigma = \sum_q \left(\sigma_{q,\overline{q}} + \sigma_{\overline{q},q}\right) \ . \tag{2.25}$$

In section 2.2 wurde der differentielle Wirkungsquerschnitt für den partonischen Prozess σ_p im Schwerpunktsystem der Konstituenten berechnet. $\tilde{\sigma}_{q,\overline{q}}(x_1, x_2, Q^2)$ ergibt sich damit nach Gl. 2.26.

$$\tilde{\sigma}_{q,\overline{q}}(x_1, x_2, Q^2) = \int \frac{\mathrm{d}\sigma_p}{\mathrm{d}\eta}\left(x_1, x_2, Q^2\right) \mathrm{d}\eta \tag{2.26}$$

Gl. 2.26 gilt im Schwerpunktsystem der Partonen und muss im Folgenden in das Laborsystem transformiert werden. Weiterhin muss die Mandelstam-Variable s in Abhängigkeit von x_1, x_2 ausgedrückt werden. Für die Partonen q und $\overline{q}$ mit den Impulsbruchteilen x_1 und x_2 lassen sich ihre Vierervektoren im Schwerpunktsystem der beiden Hadronen schreiben als Gl. 2.27, wobei E die Strahlenergie bezeichnet.

$$\mathbf{p}_q = (x_1 E, 0, 0, x_1 E) \quad \text{und} \quad \mathbf{p}_{\overline{q}} = (x_2 E, 0, 0, -x_2 E) \tag{2.27}$$

Mit Gl. 2.27 ergibt sich die Schwerpunktenergie zu:

$$\sqrt{s} = 2\sqrt{x_1 x_2} E \ . \tag{2.28}$$

Im Folgenden werden Variablen im Laborsystem ungestrichen und Variablen im Schwerpunktsystem der Partonen gestrichen benannt. Die Pseudo-Rapidität, die sich für masselose Teilchen additiv bei Inertialsystemwechsel verhält, transformiert sich, wenn sich das Schwerpunktsys-

tem der Partonen mit der Geschwindigkeit β zum Laborsystem bewegt, nach Gl. 2.29.

$$\eta' = \eta + \frac{1}{2}\ln(\frac{1-\beta}{1+\beta}) \quad \Rightarrow \quad \frac{\mathrm{d}\eta'}{\mathrm{d}\eta} = 1 \tag{2.29}$$

Damit ergibt sich für den Wirkungsquerschnitt im Laborsystem:

$$\frac{\mathrm{d}\sigma_p}{\mathrm{d}\eta} = \frac{\mathrm{d}\eta'}{\mathrm{d}\eta}\frac{\mathrm{d}\sigma_p}{\mathrm{d}\eta'} = \frac{Q_q^4 e^4}{48\pi s}(1+\tanh^2(\eta')). \tag{2.30}$$

Die Geschwindigkeit β wird aus den Dreierimpulsen $\vec{p}$ erhalten:

$$\beta = \frac{\vec{p_q} + \vec{p_{\overline{q}}}}{m_q + m_{\overline{q}}} = \frac{(x_1 - x_2)E}{(x_1 + x_2)E} = \frac{x_1 - x_2}{x_1 + x_2}\ . \tag{2.31}$$

Durch Einsetzen der gefundenen Ausdrücke für s, η' und β in Gl. 2.30, ergibt sich mit $Q^2 = 2x_1x_2E^2$ insgesamt für den differentiellen Wirkungsquerschnitt im Laborsystem:

$$\sigma_p\eta\left(x_1, x_2, Q^2, q\right) = \frac{Q_q^4 e^4}{96\pi Q^2}\left(1 + \tanh^2\left(\eta + \frac{1}{2}\ln(\frac{x_2}{x_1})\right)\right)\ . \tag{2.32}$$

Schließlich können mithilfe von Gl. 2.24, Gl. 2.26 und Gl. 2.25 die Ausdrücke Gl. 2.33 für den totalen und Gl. 2.34 für den dreifach differentiellen Wirkungsquerschnitt gefunden werden.

$$\sigma = \sum_q \int \left[f_q(x_1, Q^2) f_{\overline{q}}(x_2, Q^2) + f_{\overline{q}}(x_1, Q^2) f_q(x_2, Q^2)\right] \sigma_p \eta \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}\eta \tag{2.33}$$

$$\frac{\mathrm{d}^3\sigma}{\mathrm{d}x_1\mathrm{d}x_2\mathrm{d}\eta} = \sum_q \left[f_q(x_1, Q^2) f_{\overline{q}}(x_2, Q^2) + f_{\overline{q}}(x_1, Q^2) f_q(x_2, Q^2)\right] \sigma_p \eta \tag{2.34}$$

2.4 Umgewichtung zwischen PDF-Sets

Das quantitative Ergebnis von Gl. 2.34 hängt von der verwendeten Partondichtefunktion ab. Je nach Messung und Anpassung ergeben sich kleine Unterschiede zwischen den verschiedenen Sets. Die *Umgewichtung* entspricht einem Umrechnungsfaktor zwischen differentiellen Wirkungsquerschnitten, die mit verschiedenen Partondichtefunktionen berechnet wurden. Aus Gl. 2.34 ergeben sich die Gewichte zwischen den PDF-Sets $f^{(1)}$ und $f^{(2)}$ zu Gl. 2.35.

$$w\left(x_1, x_2, Q^2\right) = \frac{\sum_q Q_q^4 \left[f_q^{(1)}(x_1, Q^2) f_{\overline{q}}^{(1)}(x_2, Q^2) + f_{\overline{q}}^{(1)}(x_1, Q^2) f_q^{(1)}(x_2, Q^2)\right]}{\sum_q Q_q^4 \left[f_q^{(2)}(x_1, Q^2) f_{\overline{q}}^{(2)}(x_2, Q^2) + f_{\overline{q}}^{(2)}(x_1, Q^2) f_q^{(2)}(x_2, Q^2)\right]} \tag{2.35}$$

3 Maschinelles Lernen und tiefe neuronale Netzwerke

3.1 Einführung in Maschinelles Lernen

Das Konzept *Maschinelles Lernen* befasst sich damit, aus Informationen, beispielsweise Messwerte, ein statistisches Modell zu entwickeln, das die Muster hinter den Lerndaten erkennt und übertragen kann. Es werden die Teilgebiete Klassifizierung und Regression unterschieden.
Klassifizierung ordnet Objekten ihre jeweilige Gruppe, auch genannt *Label*, zu. Dies geschieht auf Grundlage der Eigenschaften eines Objektes, den sogenannten *Features*. In dieser Arbeit wird **Regression** behandelt, wobei hier anstatt einer diskreten Zuordnung eine reelle Zahl ausgegeben wird. Wird eine Funktion $f : \mathbb{R}^n \to \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x})$ betrachtet, werden die Einträge des Vektors $\mathbf{x} \in \mathbb{R}^n$ als Features und der Funktionswert $f(\mathbf{x}) \in \mathbb{R}$ als Label bezeichnet. Am Beispiel des hadronischen Diphoton-Prozesses Gl. 2.34 können x_1, x_2, η als Features und der zugehörige differentielle Wirkungsquerschnitt als Label identifiziert werden. Im **überwachten** Lernen sind alle Trainingsdaten mit Labels versehen, sodass die Vorhersage des Modells mit dem wahren Ergebnis abgeglichen werden und das Netz seine Parameter entsprechend anpassen kann, um eine minimale Abweichung zu erreichen.
Die konkrete Art des Machine-Learning, die in dieser Arbeit untersucht wird, ist das Deep-Learning, dessen Prinzip auf künstlichen neuronalen Netzwerken beruht. Diese neuronalen Netze sollen im folgenden verwendet werden, um einen Regressionsalgorithmus zu entwickeln, der gegebenenfalls hochdimensionale Funktionen erlernen und damit die Effizienz der numerischen Berechnung von differentiellen Wirkungsquerschnitten steigern kann.

3.2 Neuronale Netze

Eine Veranschaulichung des Konzeptes eines neuronalen Netzes ist in Abb. 3.1 gezeigt. Den Grundbaustein eines DNN, in dem die elementaren Berechnungen durchgeführt werden, stellt das Neuron dar, dessen Name durch das biologische Nervensystem inspiriert ist. Diese Neuronen, die auch *Units* genannt werden, können unterschiedlich stark aktiviert sein. Die Units sind in Schichten, genannt *Layern*, organisiert, zwischen denen die Ausgabewerte der Neuronen hin- und hertransferiert werden. Die Units des Layers l nehmen als Funktionsargumente

die Aktivierung von Neuronen der Schicht $l-1$ und geben ihrerseits wieder einen reellen Wert aus. Während im ersten Layer, genannt Input-Layer, die Aktivierung der Neuronen durch den Wert der eingehenden Features gegeben ist, beherbergt die letzte Schicht, der sogenannte Output-Layer, nur noch eine Node, dessen Aktivierung die Vorhersage des Netzes darstellt.

Es wird sich im Folgenden auf vollständig verbundene *Feedforward*-Netze beschränkt. Während sich vollständig verbunden darauf bezieht, dass ein Neuron mit allen Neuronen der vorhergehenden Schicht verbunden ist, versteht man unter Feedforward-Netzen, dass die Ausgabe von Units der Schicht $l-1$ nur Neuronen in Layer l beeinflusst.

Abb. 3.1: Das Konzept eines mehrschichtigen Perzeptron [**?**] mit Kreisen als Neuronen ist dargestellt.

Jedes Neuron stellt zunächst eine lineare Funktion von den Ausgaben des vorhergegangen Layers $l-1$ dar, die im Vektor $\mathbf{y}_{l-1}$ zusammengefasst sind. Die Ausgabe des n-ten Neurons der Schicht l, bezeichnet mit y_l^n, wird als Skalarprodukt zwischen den Gewichten der Node $\mathbf{w}_l^n$ dargestellt, wobei zusätzlich das Bias b_l^n addiert wird. Auf diese Lineare Funktion wird anschließend eine nichtlineare Aktivierungsfunktion σ angewendet, die es dem Netz ermöglicht, nichtlineare Zusammenhänge zu erlernen:

$$y_l^n = \sigma\left(\mathbf{w}_l^n \cdot \mathbf{y}_{l-1} + b_l^n\right) \ . \tag{3.1}$$

Für den ersten Layer entsprechen die Features $\mathbf{x}$ der Ausgabe $\mathbf{y}_0 = \mathbf{x}$. In einem vollständig verbundenen Netz ergibt sich pro Node eine lineare Gleichung der Form Gl. 3.1. Insgesamt können die Rechenoperationen, die in einem Layer stattfinden, als Matrixmultiplikation formuliert werden. Die Vektoren $\mathbf{w}_l^n$ werden hierbei zu den Zeilen der Matrix $\mathbf{W}_l$, die b_l^n in Vektoren zusammengefasst (siehe Gl. 3.2).

$$\mathbf{y}_l = \sigma\left(\mathbf{W}_l \cdot \mathbf{y}_{l-1} + \mathbf{b}_l\right) \tag{3.2}$$

Im Neuron des Output-Layers findet schließlich die Ausgabe des Funktionswertes y statt. Das Ziel ist es nun, die Abweichung des Ausgabewertes y vom wahren Wert $\tilde{y}$ zu minimieren. Mathematisch wird die Abweichung als eine Metrik definiert, sodass das Erlernen der freien Parameter eines künstlichen neuronalen Netzes zum Optimierungsproblem wird. Im Kontext von ML wird die zu minimierende Metrik, die abhängig von allen Gewichten $\mathbf{W}$ und Biases $\mathbf{b}$ ist, als Kostenfunktion oder Verlustfunktion (Loss-Funktion) bezeichnet, wobei hier eine

mögliche Wahl die mittlere quadratische Abweichung ist:

$$C(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - \tilde{y}^{(i)} \right)^2 \ . \tag{3.3}$$

Als Kostenfunktion kann prinzipiell jede Metrik verwendet werden, die zielführend erscheint, sodass je nach Problemstellung abgewogen werden muss, welche Wahl die besten Ergebnisse liefert. Da die analytische Berechnung von Extremstellen der Verlustfunktion von neuronalen Netzen nicht möglich ist, wird auf Gradientenabstieg (*Gradient-Descent* [?]) zurückgegriffen. Hierbei wird, beim Output-Layer beginnend, der Gradient der Kostenfunktion berechnet und per Kettenregel zum vorherigen Layer fortgepflanzt. Dieser Vorgang, für alle Units einen Gradienten zu berechnen, nennt sich *Backpropagation* und führt in der Anwendung auf die Methode des automatischen Differenzierens zurück. Die Gradienten werden gemittelt für eine Anzahl an Trainingspunkten, genannt Batch, berechnet und auf die Gewichte angewendet, sodass sich einem lokalen oder auch globalen, Minimum genähert werden kann (*Stochastic Gradient Descent*, SGD [?]).

3.3 Training und Hyperparameter

Parameter, die Programmierende vorher festlegen müssen und nicht vom Algorithmus erlernt werden, werden Hyperparameter genannt. Es werden im Weiteren die folgende Hyperparameter besprochen, wobei nicht zwischen solchen Hyperparametern, die die Architektur des Netzes bestimmen, und Trainingsparametern, die das Lernverhalten bestimmen, differenziert wird:

- Anzahl der Layer und Units
- Kostenfunktion
- Aktivierungsfunktion der Neuronen
- Initialisierung der Gewichte
- Optimizer(Lernart)
- Learning-Rate(Lernrate)
- Batch-Größe
- Anzahl der Trainingsepochen
- Normalisierung

Die Architektur eines neuronalen Netzes wird durch die Anzahl an **Layer und Units** festgelegt. Tiefere neuronale Netze mit größeren Anzahlen an Neuronen sind in der Lage, kompliziertere Sachverhalte genauer zu lernen, allerdings steigt die Anzahl zu trainierender Parameter und auszuführender Rechnungen an. Bei zu komplexen Modellen für simple Sachverhalte mit wenigen Trainingspunkten kommt es häufig zur Überanpassung, bei der sich das Modell zu sehr auf die vorliegenden Daten spezialisiert und seine Generalisierungsfähigkeit verliert.
Die **Loss-Funktion** bestimmt das Lernverhalten des Netzes maßgeblich, denn für sie werden letztendlich die Gradienten berechnet. Die in dieser Arbeit verwendeten Kostenfunktionen sind in Table A.1 definiert.

Die **Aktivierungsfunktion** bricht die Linearität des Netzes und sorgt dafür, dass dieses nichtlineare Funktionen erlernen kann. Die Form und Ableitung der Aktivierungsfunktion bestimmt den Gradienten während der Backpropagation. Für Aktivierungsfunktionen mit verschwindenden Ableitungen, besonders die namensgebende Funktion ReLU [**?**], kann das *Dying-ReLU-Problem* auftreten. Hierbei werden durch große Gradienten die Gewichte eines Neurons so verändert, dass es, fast unabhängig von seinen Funktionswerten, nur noch null ausgibt[1]. Da die Ableitung in diesem Definitionsbereich ebenfalls null ist, kann sich das Neuron nicht regenerieren und trägt im Folgenden nicht mehr zum Lernprozess des Netzes bei. Die untersuchten Aktivierungsfunktionen sind in Abb. A.1 gezeigt.

An welchem Punkt des hochdimensionalen Phasenraums der Kostenfunktion der Lernprozess beginnt, wird von der **Initialisierung** festgelegt. Die Initialisierung der Gewichte ist eng verknüpft mit der verwendeten Aktivierungsfunktion. Zum Beispiel hat sich die HeNormal-Initialisierung [**?**] für die ReLU-Aktivierungsfunktion etabliert.

Die hier besprochenen Lernalgorithmen basieren auf Gradientenabstieg. Die konkrete Implementation des Gradient-descent, die sich gegebenenfalls an die vorliegende Situation anpasst, wird **Optimizer** genannt. Die **Learning-Rate** ist hierbei der Faktor, mit dem der Gradient skaliert wird, bevor er auf die Gewichte angewendet wird. Diese muss so gewählt werden, dass das Lernen weder in einem zu hohen lokalen Maximum zum Erliegen kommt noch so groß ist, dass es zu Sprüngen über das Minimum kommt. Einige auf Gradientenabstieg basierende Optimizer sind in Table A.2 erläutert.

Die **Batch-Größe** beschreibt, wie viele Objekte in einem Durchgang vom neuronalen Netz behandelt werden. Große Batch-Größen dämpfen Ausreißer und beschleunigen die Trainingszeit, wobei ein Training mit kleineren Batches detailreicher und genauer sein kann.

Die Anzahl an **Trainingsepochen** beschreibt, wie oft während des Lernvorgangs über die Trainingsdaten iteriert wird. Die Präzision eines neuronalen Netzes konvergiert idealerweise, daher kann eine Abbruchbedingung als minimale Verbesserung zwischen Epochen festgelegt werden.

Liegen Features vor, deren numerische Reichweite stark auseinandergeht, kann es sich lohnen, die Eingabewert zu **normalisieren**. Das bedeutet, alle Features auf ein festgelegtes Intervall, zum Beispiel $I = [1, 0]$, zu transformieren. So wird verhindert, dass einem Feature mit großem numerischen Wert zu viel Bedeutung zugeordnet wird.

Das Finden der besten Hyperparameter ist ein weiteres Optimierungsproblem, das abgesehen von der Suche der besten Gewichte gelöst werden muss. Für die Methoden der Gitter- oder Zufalls-Suche wird ein Gitter an Hyperparametern festgelegt. Anschließend werden im ersten Fall alle und im zweiten lediglich zufällige Gitterpunkte trainiert und evaluiert. Fortgeschrittenere Methoden der Hyperparameteroptimierung wie Bayessche Optimierung [**?**] oder Evolutionäre Optimierung [**?**] sollen in kürzerer Zeit bessere Parameter finden. Die Hyperpa-

[1] null speziell für ReLU

rameteroptimierung kann bei vielen Parametern oder langen Lerndauern sehr aufwändig sein.
Implementierung mit Keras und Tensorflow: Die Implementierung des ML-Algorithmus ist in dieser Arbeit mit der Open-Source Python-Bibliothek TensorFlow und Keras [?] umgesetzt. Keras fungiert hierbei als eine high-level API für TensorFlow. Das Erstellen eines Netzes wird mit den Modulen vereinfacht und sowohl Loss-Funktion, Optimizer als auch Initialisierungen sind bereits implementiert. Es können sowohl vorgefertigte Layer angepasst als auch Layer und Trainingsroutine selbst geschrieben werden, wobei die vorgefertigten Layer über einige Methoden verfügen, die den Umgang mit dem Netz komfortabler machen und das Speichern und Laden vereinfachen.

3.4 Transfer-Learning

Um die hohen Zeit- und Rechenkosten des Trainings zu verringern, können bereits trainierte Modelle an ein neues Problem anpasst werden. Außerdem kann mit diesem sogenannten **Transfer-Learning** die Menge an Daten, die benötigt wird, um ein brauchbares Modell zu erhalten, signifikant verringert werden.
Die Grundidee des Transfer-Learning besteht darin, dass der Algorithmus sein bereits erlerntes statistisches Modell auf eine andere Situation überträgt und gegebenenfalls nur noch die numerischen Ausgaben anpassen muss. Es ist beobachtet worden [?], dass Transfer-Learning die folgenden Vorteile bringt:

- Höherer Start, höhere Asymptote und höhere Steigung der Lernkurve
- signifikant weniger Messwerte benötigt, um brauchbare Ergebnisse zu erreichen

Konkret wird im Laufe dieser Arbeit das Transfer-Learning verwendet, um den differentiellen Wirkungsquerschnitt, berechnet mit einem PDF-Set, auf selbige, berechnet mit einem anderen PDF-Set, zu übertragen. Dabei wird von beiden angesprochenen Aspekten profitiert, da einerseits die Trainingszeit reduziert und andererseits die Zeit zur Datengeneration verkürzt werden kann.
Transfer-Learning folgt üblicherweise dem Ablauf:

- Zunächst wird ein sogenanntes Quellen-Model an einer Quellen-Datenmenge bis zur Konvergenz trainiert.
- Eine kleinere Datenmenge an Zielwerten wird erstellt.
- Die oberste, oder einige der oberen Schichten (sprich der Output-Layer und wenige darunterliegende Layer), werden entfernt.
- Die Gewichte der restlichen Layer werden zunächst fixiert, um diese nicht durch große Gradienten zu zerstören.
- Die entfernten Schichten werden mit neuen, trainierbaren Neuronen ersetzt.
- Schließlich wird das neue Modell an der kleinern Datenmenge trainiert.

- Als optionaler letzter Schitt wird das sogenannte *Fine-Tuning*(FT) eingesetzt. Bei diesem werden die fixierten Gewichte wieder gelöst.

3.5 Monte-Carlo-Integration

Monte-Carlo-Integration unterscheidet sich von anderen numerischen Integrationsmethoden vor allem dadurch, dass die Konvergenz der Integration keine Abhängigkeit von der Dimensionalität des Integrals aufweist. Monte-Carlo-Methoden konvergieren hierbei immer mit $\propto \frac{1}{\sqrt{N}}$, wobei N die Anzahl der ausgewerteten Phasenraumpunkte ist. Es wird hierbei Gebrauch vom Gesetz der Großen Zahlen gemacht und die Integrale mittels Wahrscheinlichkeitstheorie gelöst.

Betrachte eine Funktion $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x})$ und definiere ihren Erwartungswert $\langle f(\mathbf{X})\rangle$ auf Ω, wobei $\mathbf{X}$ uniform auf Ω gezogen wird, als:

$$\langle f(\mathbf{X})\rangle = \langle f\rangle = \frac{1}{\Omega}\int_\Omega f(\mathbf{x})\mathrm{d}\mathbf{x}\ . \tag{3.4}$$

Es wird nun das Gesetz der Großen Zahlen angewendet und somit ein Schätzer für den Erwartungswert von f (Gl. 3.5) gefunden.

$$\langle\tilde{f}\rangle = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i) \quad \text{mit} \quad \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i) = \langle f\rangle \tag{3.5}$$

Da der Erwartungswert nicht exakt berechnet werden kann, weil es nicht möglich ist, f an unendlich vielen Punkten zu evaluieren, wird genutzt, dass der Schätzer gegen den Erwartungswert konvergiert und $\langle\tilde{f}\rangle \approx \langle f\rangle$ genähert. Die Geschwindigkeit der Konvergenz der Näherung kann erhöht werden, indem in Gl. 3.4 eine produktive Eins in Form einer Wahrscheinlichkeitsdichte $\rho : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}_{\geq 0}, x \mapsto \rho(x)$ mit $\int_\Omega \rho(x)\mathrm{d}x = 1$ eingeführt wird (Gl. 3.6).

$$I = \int_\Omega f(\mathbf{x})\mathrm{d}x = \int_\Omega \frac{f(\mathbf{x})}{\rho(\mathbf{x})}\rho(\mathbf{x})\mathrm{d}\mathbf{x} = \left\langle\left(\frac{f}{\rho}\right)\right\rangle_\rho \tag{3.6}$$

Dabei stellt $\left\langle\left(\frac{f}{\rho}\right)\right\rangle_\rho$ den Erwartungswert von $\frac{f}{\rho}$ unter der Bedingung dar, dass die $\mathbf{x}_i$ nach der Wahrscheinlichkeitsverteilung $\rho(\mathbf{x})$ gezogen werden. Der Schätzer ergibt sich dann zu Gl. 3.7.

$$I \approx \left\langle\left(\frac{\tilde{f}}{\rho}\right)\right\rangle_\rho = \frac{1}{N}\sum_{i=1}^{N}\frac{f(\mathbf{x}_i)}{\rho(\mathbf{x}_i)} \tag{3.7}$$

Die Konvergenz der MC-Integration ist am schnellsten, wenn die Varianz von Gl. 3.7 minimiert wird. Die Varianz ist gegeben durch Gl. 3.8.

$$\mathrm{Var}\left(\frac{f}{\rho}\right) = \left\langle \left(\frac{f}{\rho} - \left\langle \frac{f}{\rho} \right\rangle\right)^2 \right\rangle = \left\langle \left(\frac{f}{\rho}\right)^2 \right\rangle - \left\langle \frac{f}{\rho} \right\rangle^2 \approx \frac{1}{N} \sum_{i=1}^{N} \left(\frac{f(\mathbf{x}_i)}{\rho(\mathbf{x}_i)}\right)^2 - I^2 \tag{3.8}$$

Die Varianz minimiert sich also, wenn jeder Summand aus Gl. 3.8 gleich groß ist. Der Vorgang, die Wahrscheinlichkeitsdichte ρ an die Form der zu integrierenden Funktion f anzupassen, wird **Impor-tance Sampling** (IS) genannt. Hierbei werden absichtlich die $\mathbf{x}_i$ mit höheren Wahrscheinlichkeiten aus den Regionen gezogen, in denen f den größten Beitrag liefert. Die Unsicherheit auf die Integration ergibt sich aus der Standardabweichung des Mittelwerts, sprich:

$$\sigma_{\left\langle \frac{f}{\rho} \right\rangle} = \frac{1}{\sqrt{N-1}} \cdot \sqrt{\mathrm{Var}\left(\frac{f}{\rho}\right)} \,. \tag{3.9}$$

Es werden im Folgenden simple Monte-Carlo-Methoden und das Importance Sampling verwendet, um aus den differentiellen Wirkungsquerschnitten die totalen Wirkungsquerschnitte zu erhalten.

4 Anwendung von Maschinellem Lernen auf den Diphoton-Prozess

4.1 Diphoton-Prozess auf Parton-Ebene

Zunächst werden neuronale Netze zur Regression der eindimensionalen differentiellen Wirkungsquerschnitte aus section 2.2 benutzt. Dabei werden geeignete Wertebereiche gewählt, sprich $\theta \in [\epsilon, \pi - \epsilon]$ bzw. $\eta \in [-3, 3]$, wobei das Verhalten des DNN für verschiedene ϵ evaluiert wird. Kleine ϵ sind hierbei interessant, weil der Wirkungsquerschnitt für $\epsilon = 0$ am Rand des Intervalls divergiert und somit das Verhalten von neuronalen Netzen an Polstellen untersucht werden kann. Die quantitativen Werte im nächsten Abschnitt sind für einen $d\bar{d} \to \gamma\gamma$-Prozess mit einer Schwerpunktsenergie von $\sqrt{s} = 200$ GeV berechnet. Die Trainingspunkte werden zufällig nach einer, wenn nicht explizit anders angegebenen, uniformen Verteilung generiert. Es werden im Weiteren nur Netze mit der gleichen Anzahl an Neuronen in jedem Layer verwendet.
Modell für $\frac{\mathrm{d}\sigma}{\mathrm{d}\eta}$: Der differentielle Wirkungsquerschnitt in Abhängigkeit der Pseudo-Rapidität ist eine gutartige Funktion ohne Pol- oder Sprungstellen. Gl. 2.20 reduziert sich von der Form auf eine $\tanh^2$-Funktion, deren Wertebereich sich über $[0, 1)$ erstreckt und damit schon von vornherein auf einen geeigneten Wertebereich normiert ist. Der Vorfaktor wird mit einer Skalierung der Funktionswerte behandelt, auf die später weiter eingegangen wird. Für diese einfache Aufgabe werden die Hyperparameter wie in Table 4.1 gewählt und keine weiteren Optimierungen, wie für die folgenden Modelle, durchgeführt. Es werden in Zukunft standardmäßig der Kernel-Initializer HeNormal verwendet und das Bias auf null initialisiert. Das Training an sich wird für alle folgenden Modelle von den, in Keras implementierbaren, `Callbacks` geregelt:

- `LearningRateScheduler`: Ein Ablaufplan wird festgelegt, der für jede Epoche die zu verwendende Learning-Rate bestimmt.
- `ReduceLROnPlateau`: Erzielt das Training bezogen auf eine bestimmte Metrik nicht einen Mindestfortschritt, wird die Learning-Rate reduziert.
- `EarlyStopping`: Erzielt das Training bezogen auf eine bestimmte Metrik für eine gewisse Epochenanzahl keinen Mindestfortschritt, wird das Training gestoppt.

Die Wahl der genauen Konfiguration der Callbacks ist in Table A.3 festgehalten. Die gelernte Funktion im Vergleich mit den analytischen Werten ist in Abb. 4.1 (a) gezeigt. Die Werte über-

Table 4.1: Die gewählten Hyperparameter des Modells $\frac{d\sigma}{d\eta}$ sind gezeigt.

Hyperparameter	Wert
Anzahl Layer	2
Anzahl Units	64
Loss-Funktion	MAE
Optimizer	Adam
Aktivierungsfunktion	ReLU
Kernel-Initializer	HeNormal
Bias-Initializer	Zeros
Learning-Rate	0.005
Batch-Größe	128
Max. Epochen	300
Anzahl Trainingspunkte	10000

lagern sich gut, sodass auf den ersten Blick kein Unterschied festzustellen ist. Bei Betrachtung des Verhältnisses wird deutlich, dass sich der Unterschied auf ca. 0.1% beläuft.

Modell für $\frac{d\sigma}{d\theta}$: Der Wirkungsquerschnitt in Abhängigkeit von θ unterscheidet sich vom vorherigen funktionalen Zusammenhang durch seine Polstellen. Da numerisch nicht mit Polstellen umgegangen werden kann, muss der Trainingsbereich auf $[\epsilon, \pi - \epsilon]$ eingeschränkt werden. Aus physikalischer Sicht ist das legitim, da die Polstellen im Strahlengang des Speicherrings liegen und damit nicht messbar sind. Detektoren, wie ATLAS [**?**] und CMS [**?**], können Photonen mit Pseudo-Rapiditäten bis zu ca. $\eta \leq 2.5$ messen, was einem $\epsilon \approx 0.163$ entspricht. Durch Normierung der Labels auf das Intervall $[-1, 1]$ kann dem Modell der Umgang mit den Polstellen erleichtert werden. Da gute Modelle hier nicht mehr trivial gefunden werden können, wird auf eine automatische, zufällige Suche zurückgegriffen (Zufalls-Suche, siehe section 3.3). Die Such-Parameter mit Ergebnis sind in Table A.4 festgehalten.

Das Modell besitzt mit doppelt so vielen Neuronen pro Layer und zwei zusätzlichen Layern um einen Faktor zwölf mehr freie Parameter als das Modell für $\sigma\eta$. Dies kann zum einen auf die Problematik der Polstellen zurückgeführt werden, zum anderen aber auch ein Effekt der Hyperparameteroptimierung sein, da komplexere Modelle Probleme generell präziser lösen können. Die Performance des Modells ist in Abb. 4.1 (b) gezeigt. Die Präzision ist trotz der komplizierteren Funktion mit Abb. 4.1 (a) vergleichbar. Im Vergleich mit den Ergebnissen, die das Modell für $\sigma\theta$ mit der Konfiguration aus Table 4.1 erreicht (siehe **??**), zeigt sich, dass durch die zufällige Suche ein um den Faktor fünf präziseres Modell gefunden werden konnte.

Abb. 4.2: Die Verteilung des Importance Sampling, die die analytische Funktion annähert ist gezeigt (siehe Gl. A.3).

Es werden zwei weitere Modelle mit den gefunde-

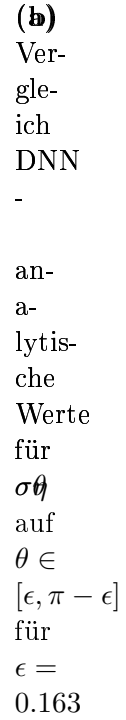

Abb. 4.1: Es sind die Vorhersagen der neuronalen Netze für den Wirkungsquerschnitt des partonischen Diphoton-Prozesses abgebildet.

nen Hyperparametern auf einem alternativen Intervall $[\epsilon', \pi - \epsilon']$ trainiert, welches näher an die Polstelle heranreicht ($\epsilon' < \epsilon$). Dabei werden die Trainingsdaten des einen Modells nach der Verteilung in Abb. 4.2 (Importance Sampling (IS)) generiert und die des anderen wie zuvor aus einer uniformen Verteilung gezogen. Zwei zusätzliche Modelle werden analog mit weniger Trainingspunkten trainiert, um eine Knappheit an Trainingspunkten zu simulieren.

In Abb. 4.3 wird die Leistung der Modelle für $\epsilon' = 0.01$ gezeigt. Wie zu erwarten weicht das Modell, das auf dem Intervall $[\epsilon, \pi - \epsilon]$ trainiert wurde, außerhalb seines Trainingsintervalls stark von der analytischen Funktion ab. Es trifft die Steigung der analytische Funktion am Startpunkt der Extrapolation, höhere Ableitungen werden vom Modell jedoch nicht erfasst. Aufgrund ihres größeren Trainingsintervalls zeigen die beiden anderen Modelle akzeptable Leistungen auch nahe an den Polstellen. In Abb. 4.3 (a) ist an einigen Stellen am Verhältnis zu sehen, dass das mit IS-generierten Trainingsdaten trainierte Netz an den Polstellen besser und im Zentrum schlechter angepasst ist. In Abb. 4.3 (b) ist der Definitionsbereich näher an die Polstelle gelegt, um die Verbesserung des IS-Modells besser aufzulösen. Anhand von Abb. 4.3 (c) wird deutlich, dass der Effekt noch größer ist, wenn keine Überfülle an Trainingsdaten vorhanden ist. Ist also der Umfang an

verfügbaren Trainingsdaten klein oder wenig Rechenleistung vorhanden, kann auf Importance Sampling zurückgegriffen werden, um Modelle mit Fokus auf wichtige Bereiche zu trainieren. Dabei ist zu beachten, dass hier ein Kompromiss zwischen Verlässlichkeit nahe der Polstelle und Verlässlichkeit im Zentrum des Definitionsbereiches eingegangen wird.
In Abb. 4.1 ist noch einmal der MAPE (Mean-Absolute-Percentage-Error) der verschiedenen Modelle für unterschiedlich generierte Testdatensätze gezeigt. Hier wird erneut deutlich, dass das Importance Sampling vor allem nützlich ist, wenn Bereiche mit hohen Funktionswerten besonders wichtig sind. Bei der Berechnung des totalen Wirkungsquerschnittes aus dem differentiellen Wirkungsquerschnitt ist genau dies der Fall.

(a) $\sigma\theta$ auf $\theta \in [\epsilon', \pi - \epsilon']$ für $\epsilon' = 0.01$

(b) Modelle mit 60000 Trainingspunkten, nahe der Polstelle

Abb. 4.3: Die Performance auf unterschiedlichen Intervallen von verschiedenen Theta-Modellen wird verglichen. In Blau: Modell, das auf $[\epsilon, \pi - \epsilon]$ mit $\epsilon = 0.163$ trainiert wurde. In Grün: Modell, das auf $[\epsilon', \pi - \epsilon']$ mit $\epsilon' = 0.01$ trainiert wurde. In Rot: Modell, das auf $[\epsilon', \pi - \epsilon']$ mit $\epsilon' = 0.01$ trainiert wurde, wobei die Punkte nach Abb. 4.2 generiert wurden.

4.2 Diphoton-Prozess auf Hadron-Ebene

4.2.1 Phasenraumselektion

Im Gegensatz zu den bisher diskutierten Prozessen ist die Reaktion $pp \to \gamma\gamma$ messbar. Der Wirkungsquerschnitt soll am Beispiel einer Messung des ATLAS-Detektors behandelt werden. Es werden einige Selektionskriterien auf die Events angewendet, die sich an einer realen Messung orientieren und durch die Detektorgeometrie motiviert sind, da dieser nicht den gesamten

Raumwinkelbereich abdecken kann. Die generierten Phasenraumpunkte werden selektiert und der Algorithmus nur an solchen Messpunkten trainiert, die auch praktisch detektierbar wären. Die verwendeten Selektionen sind angelehnt an [?] und aufgelistet in Table 4.2. Dabei sind γ und γ' die Bezeichnungen für die beiden Photonen und $p_{\mathrm{T},\gamma} = p_{\mathrm{T},\gamma'} = p_{\mathrm{T}}$ beschreibt den Impuls der produzierten Photonen transversal zum Strahlengang. Gl. 2.34 ist, aufgrund der Abhängigkeit der PDF, für große x_1, x_2 extrem klein und fällt zusätzlich stark ab, daher werden außerdem Ereignisse mit $x > 0.7$ vernachlässigt. Dieser Schnitt entfernt den Phasenraumbereich mit extrem kleinen Labels und erleichtert dem DNN den Lernprozess.

Da sich das Laborsystem vom Schwerpunktsystem der kollidierenden Quarks unterscheidet, muss sichergestellt werden, dass beide Photonen die η-Selektion erfüllen (siehe Abb. 2.2). Werden η_γ und $\eta_{\gamma'}$ in entgegengesetzte Richtungen gemessen, berechnen sich diese aus η' der Photonen im Schwerpunktsystem der Quarks nach:

$$\eta_\gamma = \eta' - \frac{1}{2}\ln(\frac{x_2}{x_1}) \quad \text{sowie} \quad \eta_{\gamma'} = \eta' + \frac{1}{2}\ln(\frac{x_2}{x_1}) \ . \tag{4.1}$$

Die Pseudo-Rapidität $\eta_{\gamma'}$ kann somit in Abhängigkeit von η_γ dargestellt werden als:

$$\eta_{\gamma'} = \eta_\gamma + \frac{1}{2}\ln(\frac{x_2^2}{x_1^2}) \tag{4.2}$$

Lorentztransformation von Gl. 2.3 liefert für den Transversalimpuls:

$$p_{\mathrm{T}} = \sqrt{x_1 x_2 \left[1 - \tanh[2](\eta_\gamma + \frac{1}{2}\ln(\frac{x_2}{x_1}))\right]} E \ , \quad \text{wobei} \quad p = \frac{1}{2}\sqrt{s} = \sqrt{x_1 x_2} E \ . \tag{4.3}$$

Für die Berechnung des Wirkungsquerschnittes nach Gl. 2.34 wird das PDF-Set CT14nnlo von LHAPDF [?] verwendet, wobei positive Beiträge von Charm- und Strange-Quark berücksichtigt und Top- und Bottom-Quark vernachlässigt werden. Die Strahlenergie beträgt $E = 6500$ GeV.

Table 4.2: Event-Selektion für den hadronischen Diphoton-Prozess in Anlehnung an Messung von ATLAS [?].

Typ	Selektion
Photon-Energie	$p_{\mathrm{T}} > 40$ GeV
Photon Winkel	$\eta_{\gamma,\gamma'} < 2.37$ ohne $1.37 < \eta_{\gamma,\gamma'} < 1.52$
Impulsbruchteil	$x_{1,2} < 0.7$

4.2.2 Schwierigkeiten und Lösungsansätze

Im Vergleich zu den vorhergegangenen Prozessen haben sich nun die Dimensionalität des Problems sowie die Gutartigkeit der Funktion verändert. Die Partondichtefunktionen, die den dreidimensionalen Wirkungsquerschnitt bestimmen, fallen exponentiell mit ihren Impulsbruchteilen x_1 und x_2 ab und besitzen Polstellen für $x \to 0$[1]. Die Labels variieren daher von 10^3 pb bis zu 10^{-20} pb. Es kommt hinzu, dass die meisten quantitativen Werte des Wirkungsquerschnittes in standardmäßig verwendeten Einheiten, sprich $1/\mathrm{GeV}^2$ und pb, viel kleiner als eins sind.

Wird ein naiver Ansatz mit gleicher Hyperparameterkonfiguration wie der des $\sigma\theta$-Modells (Table A.4), jedoch ohne Skalierung und Label-Normalisierung, unter Nutzung der ReLU-Aktivierungsfunktion verwendet, führt dies zu einem Netz, das einen konstanten Wert ausgibt (siehe Abb. 4.4 (a)). Die Ursache dafür ist das in section 3.3 angesprochene Dying-ReLU-Problem. Die quantitativ kleinen Labels führen dazu, dass durch starke Überschätzung der Funktionswerte direkt nach der Initialisierung des Netzes viele Neuronen bereits nach den ersten Batches inaktiv werden. Im vorliegenden Fall ist das Dying-ReLU-Problem so stark ausgeprägt, dass alle Neuronen des letzten Layers lediglich null zurückgeben und die Ausgabe des Netzes vom Bias des Output-Neurons bestimmt wird.

Durch die, in den bereits diskutierten Modellen verwendete, Skalierung der Labels kann sichergestellt werden, dass das Netz die Labels nach der Initialisierung nicht über-, sondern unterschätzt und die Gewichte der Neuronen zunächst anwachsen. Wie in Abb. 4.4 (b) zu beobachten ist, kann durch die Skalierung erreicht werden, dass das Modell zumindest in Phasenraumbereichen mit großen Wirkungsquerschnitten sinnvolle Ergebnisse liefert. Die starke Variation der Labels führt dazu, dass Loss-Funktionen wie der *Mean-Squared-* oder *Mean-Absolute-Error* lediglich Punkte mit hohen Wirkungsquerschnitten berücksichtigen und damit Vorhersagen schon ab einem kleinen x unbrauchbar werden.

Die Loss-Funktion *Mean-Squared-Logarithmic-Error* (MSLE) vernachlässigt Phasenraumbereiche mit kleinen Funktionswerten nicht, da es mit dieser Funktion ausschließlich auf das Verhältnis zwischen Label und Vorhersage ankommt (siehe Gl. 4.4). Um nicht mit negativen Werten zu arbeiten, wird $y \to y + 1$ transformiert. Die Skalierung unterstützt hierbei den Effekt des MSLE indirekt, da ohne die Skalierung aufgrund von $\ln(1 + y) \approx y$ bei $y \ll 1$ der MSLE stark dem MSE ähnelt. Der Logarithmus kann bereits vor der Loss-Funktion angewendet werden, was einerseits flexibler in der Verwendung der Loss-Funktion ist und andererseits den Rechenaufwand reduziert[2]. Durch die Kombination von Skalierung und Anwendung des Logarithmus können funktionierende Ergebnisse erhalten werden, wie in Abb. 4.4 (c) zu sehen

[1] wobei die PDFs numerisch ab $x_{\min} \approx 10^{-9}$ eingefroren werden.
[2] im Vergleich damit den Logarithmus einzeln für jeden Batch zu berechnen

ist.

$$C(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} \left(\ln(y^{(i)}) - \ln(\tilde{y}^{(i)}) \right)^2 = \frac{1}{N} \sum_{i=1}^{N} \left(\ln(\frac{y^{(i)}}{\tilde{y}^{(i)}}) \right)^2 \tag{4.4}$$

Andere Versuche, um mit dem Dying-ReLU-Problem umzugehen, sind die Verwendung von Aktivierungsfunktionen mit nicht-verschwindender Ableitung wie *Leaky-ReLU* oder *ELU* (siehe Abb. A.1), die es den Neuronen ermöglichen sollen, sich zu regenerieren, oder die Übergabe eines *Clipvalue*-Parameters an den Optimizer, der den Gradienten reguliert. Hierbei werden Komponenten des Gradienten, die den Clipvalue übersteigen, abgeschnitten. Im Weiteren wird jedem Optimizer ein Clipvalue von zwei übergeben.

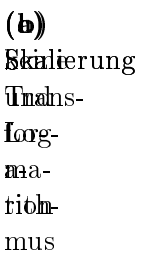

Abb. 4.4: Es ist schrittweise der Effekt der Label-Transformationen gezeigt. In (a) gibt das Netz aufgrund des Dying-ReLU-Problems lediglich null aus. In (b) wird durch die Skalierung der Phasenraumbereich mit großen Labels erlernt. Durch die Kombination von Skalierung und Logarithmus können in (c) akzeptable Genauigkeiten erreicht werden.

Die Skalierung und weitere Transformationen werden im Folgenden mit einem Transformator-Objekt realisiert, das die vorliegenden Labels nach den jeweiligen Vorschriften in Table 4.3 anpasst. Eine weitere Transformation, die untersucht wird, ist die Normalisierung der Labels auf das Intervall $[-1, 1]$, welche bereits für das Modell $\sigma\theta$ verwendet wurde.
Durch die Anwendung des Logarithmus ist sichergestellt, dass lediglich das Verhältnis zwischen

Table 4.3: Die Vorschriften der verwendeten Daten-Transformationen sind gezeigt. $\tilde{y}$ bezeichnet die Labels, x ein Feature, $\overline{x}$ den Mittelwert und σ_x die Standardabweichung des Features. $\tilde{y}^{(ln)}$ bezeichnet die bereits nach Log transformierten Labels und $\tilde{y}_{\text{min/max}}$ das größte bzw. kleinste Label des Datensatzes.

Daten-Transformationen

Abkürzung	Bezeichnung	Implementierung
No Log	Nur Skalierung	$\tilde{y} \to \frac{\tilde{y}}{\tilde{y}_{\text{min}}}$
Log	Logarithmus	$\tilde{y} \to \ln(\frac{\tilde{y}}{\tilde{y}_{\text{min}}}) =: \tilde{y}^{(\ln)}$
Base 10	Logarithmus zur Basis 10	$\tilde{y} \to \log_{10}(\frac{\tilde{y}}{\tilde{y}_{\text{min}}})$
LN	Label-Normalization	$\tilde{y} \to \frac{2\tilde{y}^{(\ln)}}{\tilde{y}^{(\ln)}_{\text{max}}} - 1$
FN [?]	Feature-Normalization	$x \to (x - \overline{x})/\sigma_x$

den rücktransformierten Labels und Vorhersagen für das Training relevant ist und nicht nur ein Phasenraumbereich exklusiv gelernt wird. Da dennoch der Bereich um die Polstelle und Phasenraumbereiche mit größeren Labels wichtiger sind, wird erneut Importance Sampling zur Generierung der Trainingspunkte verwendet. Die zur Generation der Punkte genutzten Verteilungen sind in Gl. 4.5 zusammengefasst.

$$\begin{aligned} \rho(x) &= \frac{1}{(x+\alpha)\ln(\frac{x_{\max}+\alpha}{x_{\min}+\alpha})} \qquad \text{mit} \qquad \alpha = 0.005 \\ \rho(\eta) &= \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(\eta-\eta_{\max})^2}{2\sigma^2}) & \text{für} \quad 0 \leq \eta \leq \eta_{\max} \\ \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(\eta+\eta_{\max})^2}{2\sigma^2}) & \text{für} \quad -\eta_{\max} \leq \eta < 0 \end{cases} \quad \text{mit} \quad \sigma = 1.5 \end{aligned} \tag{4.5}$$

Mit der Anzahl der generierten Phasenraumpunkte n_{total} und der Anzahl der nach Table 4.2 herausgefilterten Punkte n_{cut} beläuft sich die Effizienz der Generation k auf $k = \frac{n_{\text{total}}-n_{\text{cut}}}{n_{\text{total}}} \approx 40\%$. Im Weiteren wird die Anzahl an Trainingsdaten als n_{total} angegeben.

Training und Ergebnis: Zur Hyperparameteroptimierung wird erneut eine zufällige Suche verwendet. Die Abhängigkeit der Leistung des Netzes von den Hyperparametern und welche Parameter optimiert werden, wird in subsection 4.2.3 im Detail untersucht. Die Hyperparameter einer erfolgreichen Suche sind in Table A.5 aufgelistet.

In Abb. 4.5 sind Schnitte des dreidimensionalen Wirkungsquerschnittes an verschiedenen Phasenpunkten gezeigt. Im Vergleich mit dem zuvor diskutierten Ergebnis der naiven Hyperparameterwahl (Abb. 4.4) zeigt sich, dass die Optimierung der Hyperparameter und die Anwendung der Daten-Transformationen unerlässlich für ein anwendbares Ergebnis sind. Die maximale Abweichung beträgt an den meisten Stellen lediglich 0.5%. Für Ausnahmefälle erreicht sie bis zu $\approx 1\%$. Es lässt sich leicht der Moment erkennen, ab dem die x-Werte vernachlässigt wurden. Wie schon im Modell für $\frac{d\sigma}{d\theta}$, verläuft die Vorhersage des Modells linear weiter und entfernt sich somit von den analytischen Werten. Für große x wird das Modell den Wirkungsquerschnitt gegebenenfalls um Größenordnungen überschätzen. Da jedoch nur der integrierte Wirkungsquerschnitt über x_1, x_2 überhaupt messbar ist und sich dieser nicht merklich durch diese Abweichung beeinflussen lässt, ist die große Differenz in diesem Phasenraumbereich vernachlässigbar.

4.2.3 Evaluation der Hyperparameterwahl

Es soll nun direkt die Abhängigkeit der Leistung der Modelle von den Hyperparameter untersucht werden. Dafür wird die beste Konfiguration benutzt, die im Vorhergehenden durch die zufällige Suche gefunden wurde und in jedem Training ein Hyperparameter variiert. Da die Wahl der Anzahl an Neuronen und Layern stark korrelierten Einfluss auf die Leistung des Netzes besitzt, werden zusätzlich verschiedene Kombinationen an Layern und Units getestet. Die Modelle werden nach dem MAPE eines Satzes an Testdaten beurteilt, das genauso generiert

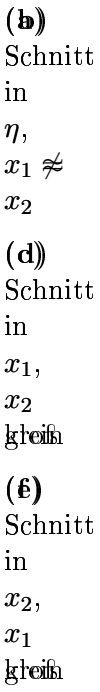

Abb. 4.5: Der Vergleich der Vorhersagen des DNN mit analytische Werten für $\frac{\mathrm{d}\sigma}{\mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}\eta}$ ist für verschiedene Phasenraumbereiche gezeigt. Die fehlenden Werte sind durch die Selektionen (Table 4.2) ausgeschlossen.

ist wie die Trainingsdaten (siehe Gl. 4.5). Jedes Modell wird fünf Mal mit zufälligen Initialisierungen an zwei Millionen gezogenen Phasenraumpunkten trainiert, um die statistische Schwankung der Güte des Modells einschätzen zu können. Die eingezeichneten Fehlerbalken (z.B. Abb. 4.6) sollen die Schwankung verdeutlichen und sind kein Maß dafür, welchen MAPE das Netz in der Praxis erreichen kann. Ausreißer werden aus den Darstellungen ausgelassen, um die wesentlichen Abhängigkeiten sichtbar zu halten.

Daten-Transformationen: Welche Daten-Transformationen für das vorliegende Problem funktionieren, ist in Abb. 4.6 gezeigt. An dieser Stelle soll erneut die Wichtigkeit dieser Transformationen hervorgehoben werden. Während die Literatur viel die Normalisierung oder das Reskalieren der Features behandelt (Bsp. [?, ?]), werden die Labels oft von Transformationen ausgenommen. Für spezielle Regressionsprobleme, wie es hier vorliegt, können diese jedoch der Schlüssel dazu sein, überhaupt konvergierende Modelle zu erhalten. Abb. 4.6 zeigt, dass verschiedene Implementationen brauchbare Ergebnisse liefern und es wichtiger ist, die Skalierung und den Logarithmus anzuwenden. Trainingsläufe weder mit Skalierung, noch Anwendung des Logarithmus sind nicht aufgeführt, da der Fehler nicht vergleichbar ist.

Architektur: Die Architektur in Abb. 4.7 zu vergleichen, ist von Interesse, da zu sehen ist,

Abb. 4.6: Die Daten-Transformationen (siehe Table 4.3) werden anhand des MAPE für eine Standardkonfiguration verglichen.

dass eine zum Problem passende Architektur effektiver ist als die Komplexität des Modells. Das Modell (32, 10)[3] mit 9665 zu trainierenden Parametern zeigt im Vergleich bessere Leistung als das Modell (1024, 2) mit 1054721 freien Parametern[4]. Die Modelle (128, 6), (256,5), (384, 4) zeigen, trotz stark variierenden Anzahlen an freien Parametern, sehr gute Genauigkeit. Beachtet werden muss jedoch, dass die rechnerische Effizienz nicht nur aus der Anzahl an freien Parametern bestimmt werden kann, sondern eine komplizierte Abhängigkeit von verschiedenen Einflussfaktoren zeigt.

Aktivierungsfunktionen: Die Abwandlungen der ReLU-Funktion zeigen sehr gute Ergebnisse, einsehbar in Abb. 4.8 (a). Die gewöhnliche ReLU zeigt jedoch schlechtere Leistung. Eine plausible Erklärung hierfür liefert wiederum das Dying-ReLU-Problem in Kombination mit der vergleichsweise hohen anfänglichen Lernrate.

Batch-Sizes: Ein Trainingsvorgang ist bei größerer Batch-Size (siehe Abb. 4.8 (b)) mit passender Hardware zwar schneller, zeigt jedoch eindeutig größere Abweichungen.

Loss-Funktion: In Abb. 4.9 (a) ist der Vergleich von drei verschiedenen Kostenfunktionen gezeigt. Für Probleme mit stark variierenden Labels setzt sich der Mean-Absolute-Error durch, da dieser weniger sensitiv auf Ausreißer oder, hier, Polstellen ist. Der Huber-Loss [?], der eine Kombination des linearen Fehlers und des quadratischen Fehlers darstellt, schlägt sich insgesamt besser als der reine quadratische Fehler, kann insgesamt jedoch nicht mit dem linearen Fehler mithalten.

Anzahl an Layer und Units pro Layer: Die Abweichung verläuft sowohl für die Anzahl an Layern als auch der Units pro Layer nach einer Kurve, die ihr Minimum bei den ermittelten optimalen Parametern hat (siehe Abb. 4.9 (b), (d)). Der Schritt zum jeweils simpleren Modell ist klein und kann bei Bedarf einen Kompromiss zwischen Geschwindigkeit und Genauigkeit darstellen.

Optimizer: Der Vergleich des Optimizers Abb. 4.9 (c) überrascht, da generell der Adam-Optimizer [?] als Weiterentwicklung des RMSprop [?] gilt. RMSprop liefert hier konstant etwas bessere Ergebnisse, wobei jedoch beachtet werden muss, dass bei solchen kleinen Abweichungen fünf Trainingsläufe nicht genug sind, um zu beurteilen, welcher Optimizer besser geeignet ist. Der Stochastic-Gradient-Descent erzielt signifikant schlechtere Ergebnisse.

Trainingsdaten: Die Größe des Sets an Trainingsdaten ist in Abb. 4.9 (e) verglichen. Wie erwartet nimmt der Fehler des Modells mit der Zahl an vorhandenen Trainingsdaten ab. Es zeigt sich, dass die Performance des Modells konvergiert und mehr als vier Millionen generierte

[3] wobei die Modelle nach dem Schema (Units, Nr of Layers) benannt sind
[4] Zur Berechnung der zu trainierenden Parameter eines Modells siehe Appendix A.

Abb. 4.7: Verschiedene Modell-Architekturen werden anhand des MAPE für eine Standardkonfiguration verglichen. Die x-Achsenbeschriftung ist geschrieben als (Units, Nr of Layers).

Daten zu keiner signifikanten Verbesserung führen.

Learning-Rate: An dem Vergleich der Learning-Rates, der in Abb. 4.9, (f) gezeigt ist, kann ein interessantes Verhalten des Netzes erkannt werden. Für eine anfängliche Learning-Rate von $1 \cdot 10^{-3}$ verändert sich der MAPE lediglich unwesentlich, woraus geschlossen werden kann, dass unabhängig der Initialisierung dasselbe lokale Minimum gefunden wird. Die Abhängigkeit der Genauigkeit von der zufälligen Initialisierung wird danach mit der Learning-Rate größer, sodass höhere und tiefere lokale Minima gefunden werden. Bei einer zu kleinen anfänglichen Lernrate kommt der Lernvorgang bereits in einem hohen Minimum zum Erliegen. Es kann daher gut sein, seine anfängliche Learning-Rate etwas größer zu initialisieren, da so diversere lokale Minima gefunden werden können. Beachtet werden muss jedoch, dass dieser Ansatz nur in Kombination mit einem Zeitplan zur Reduzierung der Lernrate funktioniert und das Dying-ReLU-Problem verstärken oder auslösen kann.

(b) Batch-Sizes

(a) Aktivierungs-funktionen

Abb. 4.8: Verschiedene Hyperparameter werden anhand des MAPE für eine Standardkonfiguration verglichen.

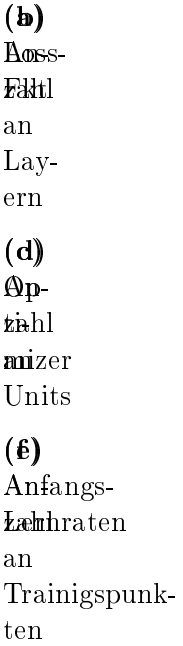

Abb. 4.9: Verschiedene Hyperparameter werden anhand des MAPE für eine Standardkonfiguration verglichen.

4.3 Umgewichtung für verschiedene PDF-Sets

Als Nächstes wird ein neuronales Netz verwendet, um die Umgewichtung (Gl. 2.35) zwischen Wirkungsquerschnitten, die mit verschiedenen Anpassungen der PDFs berechnet wurden, zu erlernen. Konkret werden die Sets *CT14nnlo* und *MMHT2014nnlo* verwendet. Die Gewichte schwanken um eins und besitzen keine Polstellen und sind damit leichter zu modellieren als das Problem im vorangegangenen Abschnitt. In Phasenraumbereichen mit großen x weichen die Sets jedoch stark voneinander ab. Angelehnt an den Phasenraumschnitt aus Table 4.2, werden die Gewichte bis zu $x_{\max} = 0.8$ trainiert, da die starken Abweichungen hier etwas später beginnen. Zusätzlich werden Ereignisse selektiert, die unabhängig von η_γ nicht messbar wären (siehe Table 4.4). Die zufällige Suche mit den besten Parametern ist in Table A.6 zu sehen.
Erwartungsgemäß ist die Genauigkeit des Modells sehr gut, wie in Abb. 4.10 beobachtet werden kann. Die Abweichung beträgt generell weniger als 0.1% und ist somit kaum von den analytisch berechneten Werten zu unterscheiden.

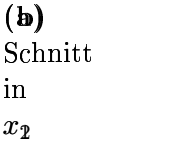

Abb. 4.10: Der Vergleich der Vorhersagen des DNN mit analytischen Werten für Schnitte der Umgewichtung w in verschiedenen Phasenraumbereichen ist gezeigt.

Table 4.4: Event-Selektion für die Umgewichtung des hadronischen Diphoton-Prozesses in Anlehnung an Messung mit ATLAS [**?**].

Typ	Selektion
Photon-Energie	$p_T > 40$ GeV $\Rightarrow$ $\sqrt{x_1 x_2} E > 40$ GeV
Photon Winkel	$\eta_{\gamma,\gamma'} < 2.37$ $\Rightarrow$ $\frac{1}{2}\ln(x_2^2/x_1^2) < 4.74$
Impulsbruchteil	$x_{1,2} < 0.8$

4.4 Transfer-Learning zwischen PDF-Sets

Eine weitere Möglichkeit, den Wirkungsquerschnitt, der mit einem anderen PDF-Set berechnet wurde, zu ermitteln, ist Transfer-Learning (siehe section 3.4). Mit Transfer-Learning können der, in den relevanten Phasenraumbereichen kleine, Unterschied ausgeglichen und mit wenig Aufwand gute Modelle für andere PDF-Sets erhalten werden. Es kommt erneut eine zufällige Suche zum Einsatz, um gute Hyperparameter für den Transfer zu finden. Es wird unter anderem optimiert, wie viele Layer aus dem Quellen-Modell entfernt und wie viele Layer hinzugefügt werden, und evaluiert, ob das Modell anschließend per Fine-Tuning (siehe section 3.4) weitertrainiert wird. Die Ergebnisse der Suche sind in Table 4.5 gezeigt. Es zeigt sich jedoch, dass der zusätzlich hinzugefügte Layer keine Verbesserung der Genauigkeit mit sich bringt, sodass im Folgenden lediglich das Output-Neuron ausgetauscht wird.
In Abb. 4.11 sind Schnitte des differentiellen Wirkungsquerschnitts des besten transferierten Modells, des Modells, das als Quelle gedient hat, und der analytischen Werten gezeigt. Es ist zu beobachten, dass die Genauigkeit des transferierten Modells fast identisch mit der des Quellen-Modells in Abb. 4.5 ist. Weder verliert das Modell beim Transfer an Genauigkeit noch kann eine Verbesserung der Performance festgestellt werden.
Sowohl das Transfer-Learning als auch die Umgewichtung sind also legitime Methoden, um den Wirkungsquerschnitt von einem PDF-Set auf das nächste zu übertragen. Ein visueller Vergleich folgt in Abb. 4.12, wobei sich hier nicht erkennen lässt, welche Methode die besseren Leistungen zeigt. Interessanterweise zu sehen ist, dass sich die Form des Ratios beider Modelle ähnelt. Der Grund hierfür liegt darin, dass das transferierte Modell vom Quellen-Modell abstammt und sich die Gewichte der Neuronen nur schwach unterscheiden.
In Table 4.6 sind einige Kenndaten der Modelle gegenübergestellt. Es kann eindeutig beobachtet werden, dass sich die Menge an Lerndaten und damit auch die Trainingsdauer durch das Transfer-Learning signifikant verringert haben. Es konnte mit einfachen Mitteln eine Reduktion um den Faktor vier an Trainingspunkten erreicht werden. Es ergibt sich, dass das präziseste und schnellste Modell die Umgewichtung der analytisch berechneten Werte ist. Steht eine große Anzahl an Werten von differentiellen Wirkungsquerschnitten zur Verfügung, dann ist dies das optimale Modell. Ist jedoch ein vollständiges Modell benötigt, das nicht auf die analytische Berechnung von differentiellen Wirkungsquerschnitten angewiesen ist, schneidet das Modell *Transfer + FT* am besten ab. Es übertrifft die Alternative *Umgewichtung + Quelle* in den bedeutenden Kriterien. Da im zweiten Fall sowohl die Gewichte als auch die Quellen-Wirkungsquerschnitten mit neuronalen Netzen berechnet werden, addiert sich hier die Berechnungszeit für 10^6 Punkte (Time per Million: TPM) im Vergleich zu den restlichen Modellen. Wird das Quellen-Modell geeignet transformiert, passt sich das Netz gut an die neuen Daten an und der MAPE bleibt minimal. Werden die Ergebnisse des Quellen-Modells neu gewichtet, pflanzen sich beide Ungenauigkeiten fort und die Unsicherheit steigt etwas.
Es kann geschlussfolgert werden, dass das Transfer-Learning eine generell bessere Methode für

Table 4.5: Gezeigt sind Parameter einer zufälligen Suche für den Transfer eines Quellen-Modells mit bester Konfiguration Transfer zwischen PDF-Sets.

Hyperparameter	Pool	Beste Konfig.
Anzahl entfernte Layer	$\{1, 2\}$	1
Anzahl hinzugefügte Layer	$\{0, 1, 2\}$	1
Units (hinzugefügte Layer)	$\{64, 128, 512\}$	128
Aktivierungsfunktion	ReLU, Leaky-ReLU, Sigmoid	ReLU
Learning-Rate	$\{10^{-2}, 5 \cdot 10^{-3}, 10^{-3}, 10^{-4}\}$	$5 \cdot 10^{-3}$
Batch-Größe	$\{128, 512, 768, 2048, 8196\}$	768
Fine-Tuning	True, False	True
Loss-Funktion	MAE	
Optimizer	Adam	
Max. Epochen	100	
Trainingspunkte	1.000.000	

Table 4.6: Vergleich von Umgewichtung- und Transfer-Modellen. TPM (Time per Million) ist die Berechnungszeit für 10^6 Punkte. Die Zeiten sind aufgenommen mit der Konfiguration in Table A.7. Der MAPE ist für ein Test-Datensatz von MMHT2014nnlo-Werten berechnet. Beachte, dass ein transferiertes Modell, das ohne FT trainiert wurde, schlechtere Ergebnisse erzielt, als das nicht transferierte Quellen-Modell.

Modell	MAPE	Training[s]	Punkte	TPM[s]
Umgewichtung + Quelle	0.076	145.38	1M	0.683
Umgewichtung + Analy.	0.017	145.38	1M	0.171
Transfer	0.228	68.61	1M	0.503
Transfer + FT	0.064	162.13	1M	0.520
Quellen-Modell	0.207	860.30	4M	0.504

den angesprochenen Zweck ist und das Erlernen der Gewichte zwar gut funktioniert, jedoch nur nützlich ist, wenn speziell die Gewichte benötigt werden.

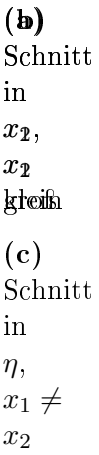

Abb. 4.11: Der Vergleich der Vorhersagen eines transferierten Modells und eines Quellen-Modells ist für verschiedene Phasenraumbereiche gezeigt.

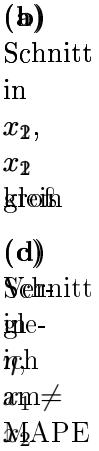

Abb. 4.12: Der Vergleich der Vorhersagen eines transferierten Modells und eines umgewichteten Quellen-Modells (rw) ist für verschiedene Phasenraumbereiche gezeigt.

4.5 Monte-Carlo-Integration

4.5.1 Partonischer Prozess

Es werden Monte-Carlo-Methoden zur Integration von Gl. 2.19, Gl. 2.20 und den zugehörigen Näherungen durch Machine-Learning-Modelle verwendet. Zur Integration von Gl. 2.19 wird das Importance Sampling (IS) aus Abb. 4.2 genutzt, um die Konvergenz des Integrals zu beschleunigen. Der Prozess $qq \to \gamma\gamma$ ist zwar nicht messbar, es muss dennoch ein Schnitt in η festgelegt werden, da der totale Wirkungsquerschnitt sonst divergiert. Es werden die

Beschränkungen Gl. 4.6 verwendet.

$$\eta \leq 2.5 \qquad \Rightarrow \qquad \theta \in [\epsilon, \pi - \epsilon] \quad \text{mit} \quad \epsilon = 0.1638 \tag{4.6}$$

Die Unsicherheit der Monte-Carlo-Integration wird aus Gl. 3.9 bestimmt. Die Integrationen wird mit 1000 Stützstellen durchgeführt und 100 Mal wiederholt. In Table 4.7 sind die erhaltenen Ergebnisse mit dem analytischen Wert verglichen. Es ist zu sehen, dass die neuronalen

Table 4.7: Die Ergebnisse der Monte-Carlo-Integration des partonischen Diphoton Prozesses sind für den analytischen und durch ML genäherten Integranden verglichen. Für $\sigma\theta$ wird der Nutzen von Importance Sampling untersucht

Integrand	σ_{tot}[pb]	
analytische Stammfunktion	0.053793	$\pm$ 0
$\sigma\theta$ analytisch + IS	0.05382	$\pm$ 0.00006
$\sigma\theta$ analytisch	0.05389	$\pm$ 0.00015
$\sigma\theta$ ML + IS	0.05386	$\pm$ 0.00005
$\sigma\eta$ analytisch	0.053796	$\pm$ 0.000034
$\sigma\eta$ ML	0.053801	$\pm$ 0.000034

Netze so präzise sind, dass ihre Abweichung in der Unsicherheit der Monte-Carlo-Integration untergeht. Das sind gute Voraussetzungen für die Anwendbarkeit von neuronalen Netzen auch bei höherdimensionalen Prozessen. Das simple Importance-Sampling bringt eine signifikante Varianz-Verringerung mit sich.

4.5.2 Hadronischer Prozess

Auch für den hadronischen Diphoton-Prozess wird Importance-Sampling genutzt. Für die Generation der Impulsbruchteile x wird die Verteilung aus Gl. 4.5 verwendet. Aufgrund der Selektionen leisten Phasenraumpunkte mit kleinem η einen größeren Beitrag zum messbaren σ_{tot}, daher werden die Pseudo-Rapiditäten aus einer Gaußverteilung um null (Gl. 4.7) gezogen.

$$\rho(\eta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{\eta^2}{2\sigma^2}) \quad \text{mit} \quad \sigma = 2 \tag{4.7}$$

Zunächst werden die Wirkungsquerschnitte über zwei Freiheitsgrade integriert und in Abhängigkeit von x_1, x_2 und η betrachtet. Dazu werden 10.000.000 Punkte generiert und der Prozess zwanzigmal wiederholt. Die Ergebnisse sind in Abb. 4.13 dargestellt. Es ist zu beobachten, dass sich die integrierten Wirkungsquerschnitte für die analytischen Werte und die Vorhersagen des DNN an vielen Phasenpunkten überdecken. Lediglich für große x überschätzt die Vorhersage wie erwartet den eigentlichen Wert. Diese Abweichung kann jedoch vernachlässigt werden, da am Ratio von Abb. 4.13 (c) zu sehen ist, dass die analytischen Werte insgesamt

(b) Integration über x_2, η

(c) Integration über x_1, x_2

Abb. 4.13: Die MC-Integration über zwei Freiheitsgrade ist gezeigt, wobei in den jeweiligen Bins über den dritten Freiheitsgrad integriert wurde. Die Unsicherheiten sind in logarithmischer Darstellung nicht sichtbar.

unterschätzt werden. Der Grund hierfür liegt vermutlich in der Polstelle an $x = 0$. Die Selektionen nehmen viele Phasenraumpunkte um $x_1 = x_2 = 0$ mit großem Wirkungsquerschnitt heraus, die die p_{T}-Hürde nicht erfüllen, und führen zu einer geringfügigen Unterschätzung des Wirkungsquerschnittes an diesen einflussreichen Stellen.
Zur Integration über alle Freiheitsgrade werden die gleichen Daten wie im vorherigen Abschnitt verwendet. Die Ergebnisse sind in Gl. 4.8 aufgeführt.

$$\begin{aligned} \sigma_{\mathrm{tot}}^{\mathrm{analytic}} &= (5.1661 \pm 0.0023)\ \mathrm{pb} \\ \sigma_{\mathrm{tot}}^{\mathrm{ml}} &= (5.1588 \pm 0.0022)\ \mathrm{pb} \end{aligned} \tag{4.8}$$

Dies entspricht einer Abweichung von 0.14%, wobei die Ergebnisse außerhalb ihrer Unsicherheiten liegen, insbesondere da dieselben zufälligen Phasenraumpunkte für die Integration der analytischen und genäherten Werte genutzt wurden und damit die statistischen Unsicherheiten stark korreliert sind. Es kann abschließend gesagt werden, dass die Präzision des DNN sehr gut und die Näherung erfolgreich gelungen ist.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

In dieser Arbeit wurde der Diphoton Prozess als $q\bar{q} \to \gamma\gamma$ und $pp \to \gamma\gamma$ in führender Ordnung behandelt und analytische Ausdrücke für die jeweiligen differentiellen Wirkungsquerschnitte hergeleitet. An diesen Beispielen wurde anschließend die Eignung von tiefen neuronalen Netzwerken zur Näherung des Integranden überprüft. Dabei mussten verschiedene Schwierigkeiten, wie quantitativ kleine Labels, bedacht und behandelt werden. In diesem Kontext wurde die Wichtigkeit von Label-Transformationen deutlich. Im Anschluss wurden die Gewichte zwischen den Sets der Partondichtefunktionen CT14nnlo und MMHT2014nnlo erlernt und angewendet. Schließlich wurde die Möglichkeit der Nutzung von Transfer-Learning zur Umgewichtung eines Modells überprüft. Mithilfe von Monte-Carlo-Methoden wurden die analytischen und vorhergesagten differentiellen Wirkungsquerschnitte integriert.

Wie erwartet, haben die neuronalen Netze keine Probleme mit einfachen Regressionsaufgaben, wie dem Wirkungsquerschnitt des $q\bar{q} \to \gamma\gamma$ Prozesses. Die Funktionswerte können mit ausgezeichneter Genauigkeit ($\approx 0.1\%$ Abweichung) und wenig Aufwand vorhergesagt werden.

Dagegen ist der differentielle Wirkungsquerschnitt des Prozesses $pp \to \gamma\gamma$ nicht trivial. Hier müssen Hürden wie das Dying-ReLU-Problem und die exponentielle Variation der Funktionswerte überwunden werden. Dabei kann es helfen, kaum beitragende Phasenraumbereiche zu vernachlässigen, um die Spanne an Größenordunungen, über die sich die Wirkungsquerschnitte verteilen, zu verkleinern. Die Label-Transformationen sind essentiell, um Modelle zu finden, die gute Genauigkeit ($\approx 0.5\%$ Abweichung) zeigen.

Das Erlernen der Umgewichtung von Wirkungsquerschnitten stellt für das neuronale Netz keine Schwierigkeit dar, solange ein geeigneter Phasenraumbereich gewählt wird. Das Netz kann die Gewichte mit exzellenter Genauigkeit ($< 0.1\%$ Abweichung) vorhersagen.

Transfer-Learning stellt sich als eine gute Möglichkeit heraus, aus einem bereits vorhandenen Modell ein Modell für ein anders Set an PDFs zu erhalten. Die Berechnungsgeschwindigkeit und Genauigkeit wird durch das Transfer-Learning im Vergleich zur Umgewichtung eines bereits vorhandenen Quellen-Modells verbessert.

5.2 Ausblick

Die in dieser Arbeit behandelten Methoden haben gute Ergebnisse an den niedrigdimensionalen Beispielen gezeigt. Als Nächstes sollte nun der Test an höherdimensionalen Prozessen mit analytisch nicht mehr oder nur aufwändig zu berechnenden Wirkungsquerschnitten folgen. Es muss noch untersucht werden, ob die neuronalen Netze ihre Genauigkeit auch in höheren Dimensionen aufrechterhalten können und ob dies mit einer realisierbaren Zahl an Trainingspunkten möglich ist. Anschließend muss überprüft werden, wie groß die Verringerung der Rechenzeit bei Nutzung von neuronalen Netzen ist. Bereits in dem behandelten Beispiel des hadronischen Diphoton-Prozesses ist die Näherung mittels DNN schneller als die Berechnung über Partondichtefunktionen. Die in dieser Arbeit erhaltenen Ergebnisse sind gute Voraussetzungen für die Funktionstüchtigkeit im Höherdimensionalen.
Auch das Transfer-Learning hat in dieser Arbeit seine Funktionalität bewiesen. Es muss jedoch nicht beim Transfer zwischen PDF-Sets bleiben. Transfer-Learning kann zwischen viel diverseren Sachverhalten eingesetzt werden. Es könnte sich lohnen, den Transfer zwischen sich ähnelnden Prozessen in der Teilchenphysik zu untersuchen.
Abgesehen von den hier untersuchten Verwendungsmöglichkeiten gibt es noch unzählige weitere Anwendungsmöglichkeiten von Machine-Learning oder tiefen neuronalen Netzen in der Teilchenphysik. Hierunter fallen beispielsweise die Umgewichtung von Simulationen oder das Training eines DNN zur Reproduktion von Dektektorverhalten, um die standardmäßige Monte-Carlo-Simulation zu ersetzen.

A Anhang

Berechnung der freien Parameter eines Modells: Betrachte ein DNN mit N versteckten Layern l, wobei n_l die Anzahl an Neuronen von Layer l bezeichnet, dann berechnet sich die Anzahl k an zu trainierenden Parametern nach Gl. A.1. Hierbei bezeichnet n_0 die Dimensionalität der Features.

$$k = \sum_{l=1}^{N} [(n_l \cdot n_{l-1}) + n_l] + (l_N + 1) \tag{A.1}$$

Gilt hierbei $n_l = n_{l+1} = n$ für $l > 0$ vereinfacht sich Gl. A.1 zu:

$$k = n \cdot (n_0 + 1) + (N - 1) \cdot n(n + 1) + n + 1 = n\,(n_0 + 2 + (n + 1)(N - 1)) + 1\ . \tag{A.2}$$

Importance Sampling: Die Verteilung, die in section 4.1 genutzt wurde, um ein Modell für $\sigma\theta$ zu trainieren, ist in Gl. A.3 definiert.

$$\rho(\theta) = a((x - \mu)^4 + b) \quad \text{mit} \quad b = 0.4,\ \mu = \frac{\pi}{2} \quad \text{und a, sodass} \quad \int_{\epsilon}^{\pi-\epsilon} \rho(\theta)\mathrm{d}\theta = 1 \tag{A.3}$$

A.1 Such- und Hyperparameter

Abb. A.1: Es wurde verschiedene Aktivierungsfunktionen während der Arbeit getestet.

Table A.1: Die konkreten Implementierungen der erwähnten und genutzten Kostenfunktionen sind aufgelistet.

Loss-Funktionen

Bezeichnung	Implementierung
Mean-Absolute-Error (MAE)	$C(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} y^{(i)} - \tilde{y}^{(i)}$
Mean-Squared-Error (MSE)	$C(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - \tilde{y}^{(i)}\right)^2$
Mean-Absolute-Percentage-Error (MAPE)	$C(\mathbf{W}, \mathbf{b}) = \frac{100}{N} \sum_{i=1}^{N} y^{(i)} - \tilde{y}^{(i)} / \tilde{y}^{(i)}$
Mean-Squared-Logarithmic-Error (MSLE)	$C(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} \left(\ln(y^{(i)}) - \ln(\tilde{y}^{(i)})\right)^2$
Huber-Loss	$C(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} L_\delta\left(y^{(i)}, \tilde{y}^{(i)}\right)$
mit	$L_\delta\left(y^{(i)}, \tilde{y}^{(i)}\right) = \begin{cases} \frac{1}{2}\left(y^{(i)} - \tilde{y}^{(i)}\right)^2 & \text{für } y^{(i)} - \tilde{y}^{(i)} \leq \delta \\ \delta y^{(i)} - \tilde{y}^{(i)} - \frac{1}{2}\delta^2 & \text{sonst} \end{cases}$

Table A.2: Die verwendeten Optimizer werden kurz erläutert.

Optimizer

Bezeichnung	Funktionsweise
2*SGD	Der Gradient für einen Batch gemittelt und auf die Gewichte angewendet: $\mathbf{W} \to \mathbf{W} - \frac{\alpha}{N} \sum_{i=1}^{N} \nabla C_i(\mathbf{W})$ mit α: Learning-Rate und C_i: Loss für Punkt i
2*RMSprop [?]	Die Learning-Rate wird für jedes Gewicht durch ein gewichtetes Mittel von einigen vorhergehenden Gradienten geteilt.
2*Adam [?]	Die Learning-Rate wird für jedes Gewicht durch ein gewichtetes Mittel von einigen vorherg. Gradient geteilt und die Gradienten sind träge (*Momentum*).

Table A.3: Es ist eine Liste mit den für alle Modelle verwendeten Callbacks zur Steuerung des Trainings angegeben.

Callbacks

Bezeichnung	Implementation
3*LearningRateScheduler	nach einer Verzögerung von 10 Epochen, wird die Learning-Rate nach jeder Epoche um 5% reduziert, bis diese auf $5 \cdot 10^{-8}$ abgefallen ist.
2*ReduceLROnPlateau	Fällt der Loss nach einer Epoche nicht um mindestens $2 \cdot 10^{-6}$, wird die Learning-Rate um 50% reduziert.
3*EarlyStopping	Fällt der Loss in drei aufeinanderfolgenden Epochen nicht um $2 \cdot 10^{-7}$ ab, wird der Trainingsvorgang gestoppt.

Table A.4: Gezeigt sind die Parameter der zufälligen Suche für $\sigma\theta$ mit bester Konfiguration.

Hyperparameter	Pool	Beste Konfig.
Anzahl Layer	$\{1, 2, 3, 4\}$	4
Anzahl Units	$\{32, 64, 128, 256\}$	128
Loss-Funktion	MAE, MSE, Huber	MAE
Optimizer	Adam, RMSprop, SGD	Adam
Aktivierungsfunktion	ReLU, Leaky-ReLU, Sigmoid	Leaky-ReLU
Learning-rate	$\{10^{-2}, 5 \cdot 10^{-3}, 10^{-3}, 10^{-4}\}$	$5 \cdot 10^{-3}$
Batch-Größe	$\{64, 128, 512, 768, 2048\}$	128
Label-Normalisierung	$\{\text{keine}, [-1, 1]\}$	$[-1, 1]$
Skalierung	True	
Max. Epochen	200	
Anzahl Trainingspunkte	60000	

Table A.5: Gezeigt sind die Parameter der zufälligen Suche für $\frac{\mathrm{d}^3\sigma}{\mathrm{d}x_1\mathrm{d}x_2\mathrm{d}\eta}$ mit bester Konfiguration.

Hyperparameter	Pool	Beste Konfig.
(Units, Nr. of Layers)	$\{(256,5),(512,3),(64,7),(1024,2),(128,6)\}$	$(256,5)$
Loss-Funktion	MAE, MSE, Huber	MAE
Optimizer	Adam, RMSprop	Adam
Aktivierungsfunktion	ReLU, Leaky-ReLU, Sigmoid, ELU, tanh	Leaky-ReLU
Learning-rate	$\{10^{-2}, 5\cdot10^{-3}, 10^{-3}, 10^{-4}\}$	10^{-2}
Batch-Größe	$\{256, 128, 512, 768, 1024\}$	256
Basis 10	True, False	True
Label-Normalisierung	$\{\text{keine}, [-1,1]\}$	keine
Feature-Normal.	True, False	True
Skalierung	True	
Logarithmus	True	
Max. Epochen	100	
Trainingspunkte	4.000.000	

Table A.6: Gezeigt sind die Parameter der zufälligen Suche für die Umgewichtung des differentiellen Wirkungsquerschnitt mit bester Konfiguration.

Hyperparameter	Pool	Beste Konfig.
Anzahl Layer	$\{1,2,3,4\}$	2
Units	$\{32, 64, 128, 256\}$	256
Loss-Funktion	MAE, MSE	MAE
Optimizer	Adam, RMSprop, SGD	Adam
Aktivierungsfunktion	ReLU, Leaky-ReLU, Sigmoid	Leaky-ReLU
Learning-rate	$\{10^{-2}, 5\cdot10^{-3}, 10^{-3}, 10^{-4}\}$	$5\cdot10^{-3}$
Batch-Größe	$\{256, 128, 512, 768, 1024\}$	512
Label-Normalisierung	$\{\text{keine}, [-1,1]\}$	keine
Feature-Normal.	True, False	True
Skalierung	False	
Logarithmus	False	
Max. Epochen	100	
Trainingspunkte	1.000.000	

Table A.7: Es ist eine Liste der verwendeten Hard- und Software angegeben, mit der alle Zeiten aufgenommen wurden.

Setup	
Betriebssystem	Ubuntu 20.04.2.0 LTS
CPU	Intel Core i5-7300HQ
GPU	NVIDIA Geforce GTX 1050 Ti
Arbeitsspeicher	8 GB DDR4 2666 MHz
Python	3.8.5
TensorFlow	2.4.1
CUDA	11.3.0

A.2 Abkürzungsverzeichnis

Table A.8: Eine Liste der in dieser Arbeit verwendeten Abkürzungen ist gezeigt.

ML	Machine-Learning
TL	Transfer-Learning
DNN	Deep-Neural-Network
PDF	Partondichtefunktion
MC	Monte-Carlo
Features	Eingabewerte eines ML-Algorithmus
Labels	wahrer Funktionswert der Features
Units	Neuronen, Grundbaustein des DNN
Layer	Schicht von Neuronen
MSE	Mean-Squared-Error, mittlere quadratische Abweichung
MAE	Mean-Absolute-Error, mittlere absolute Abweichung
MAPE	Mean-Absolute-Percentage-Error
MSLE	Mean-Squared-Logarithmic-Error
SGD	Stochastic-Gradient-Descent
RMSProp	Root Mean Square Propagation
Base 10	Daten werden mit Logarithmus zur Basis 10 transformiert
FN	Feature-Normalization
LN	Label-Normalization
Log	Nur Scaling+Logarithmus
No Log	Nur Scaling
FT	Fine-Tuning
rw	Reweighting/Umgewichtung
TPM	Time per Million, Zeit zur Eval. von 10^6 Phasenraumpunkten

B Bibliography

Danksagung

Danke an Frank Siegert, der mich während dieser Arbeit betreut und es mir möglich gemacht hat, an diesem interessanten Thema zu arbeiten.

Ich bedanke mich vielmals bei Christian Wiel für seine Unterstützung und sein offenes Ohr während meiner Arbeitsphase. Trotz Corona hatte ich so einen Ansprechpartner, der immer schnell und verlässlich Hilfe geleistet hat. Weiterhin möchte ich mich bei ihm für das Korrekturlesen meiner Arbeit bedanken.

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.

Andreas Weitzel
Dresden, Mai 2021