Neuronale Netze zur Bestimmung von Jet-Impulsen im ALICE-Experiment am LHC

Neural Networks for Determination of Jet Momenta in the ALICE-Experiment at LHC

Bachelorarbeit
im Studiengang Physik an der Westfälischen Wilhelms-Universität Münster
Institut für Kernphysik

Vorgelegt von
Alexander Leifhelm

Erstgutachter: Dr. Klein-Bösing

Zweitgutachter: Dr. Rüdiger Haake

Münster, den 19. September 2017

Plagiatserklärung der / des Studierenden

Hiermit versichere ich, dass die vorliegende Arbeit über „Neuronale Netze zur Bestimmung von Jet-Impulsen im ALICE-Experiment am LHC“ selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

_____

(Datum, Unterschrift)

Ich erkläre mich mit einem Abgleich der Arbeit mit anderen Texten zwecks Auffindung von Übereinstimmungen sowie mit einer zu diesem Zweck vorzunehmenden Speicherung der Arbeit in eine Datenbank einverstanden.

_____

(Datum, Unterschrift)

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Nach aktuellen Modellen führen Kollisionen von Blei-Ionen mit einer ausreichend hohen Energie, wie sie am ALICE-Experiment am LHC untersucht werden, dazu, dass sich die in ihnen enthaltenen Nukleonen in ihre Bestandteile trennen und ein Quark-Gluon-Plasma (QGP) gebildet wird. Die zuvor durch die starke Wechselwirkung zusammengehaltenen Gluonen und Quarks streuen untereinander im Rahmen des Impulsübertrags der hochenergetischen Kollision und gehen zu Jets aus farbneutralen Mesonen und Hadronen über. Solange die Teilchen genug Energie haben, kommt es zur Entstehung weiterer Teilchen.
Für eine Untersuchung des QGP müssen die Transversalimpulsbeiträge der ursprünglichen Konstituenten daher durch Korrekturen der Messdaten von Jets mit nachträglich entstandenen Konstituenten ermittelt werden. Um dies zu erreichen bedarf es der Anwendung dafür geeigneter Korrekturmethoden.

1.2 Problemstellung

Das Thema dieser Arbeit ist die Entwicklung eines durch verschiedene Parameter und Modellstrukturen optimierten neuronalen Netzes, welches in der Lage ist, den Wert des Transversalimpulses der ursprünglichen Konstituenten $p_{T;True}$ aus den Daten der Konstituenten von mit simulierten Jets einer mit π^+ und π^- in einem Toy-Modell angereicherten Proton-Proton-Kollision aus PYTHIA 8 (Schwerpunktsenergie $\sqrt{s} = 2{,}76\,\text{TeV}$) zu ermitteln. Als Inputdaten werden hierfür der Transversalimpuls $p_{T_{Konst.}}$, die Pseudorapidität $\eta_{Konst.}$ und der Azimuthalwinkel $\phi_{Konst.}$ der Konstituenten genutzt. Die Verteilung der den pp-Kollisionsevents hinzugefügten Daten wird dabei anhand real gemessener Blei-Ionen-Kollisionsevents ($\sqrt{s} = 2{,}76\,\text{TeV}$) bestimmt.
Als grundsätzliche Netzstrukturen werden eindimensionale Convolutional Neural Networks und ein- und mehrschichtige Perzeptronen im Rahmen eines einfachen Feedforward-Netzes verwendet.

Ziel der Arbeit ist es, über mehrere Rastersuchen im Parameterraum der jeweiligen Modellstrukturen die Vorhersage des $p_{T;True}$ auf Basis der Inputdaten der Jets zu optimieren und insbesondere den *Weg* zu diesem Modell zu dokumentieren. Dies umfasst auch die Darlegung des Verhaltens der Vorhersagequalität bei Variation der Modellparameter, um die Schritte bei der Parameterfindung detailliert darzulegen. So soll ein Einblick in den Optimierungsprozess der Netz-Modelle gewährleistet werden.

2 Theorie

2.1 Physikalische Grundlagen

Obwohl der Schwerpunkt der Arbeit in der Suche nach dem für die Hintergrundkorrektur bestmöglichen neuronalen Netz liegt, ist das Verständnis der in der Arbeit genutzten Größen und Begriffe wichtig. Aus diesen Gründen soll ein kurzer Überblick darüber gegeben werden.

2.1.1 Schwerionen-Kollisionen

Bei Schwerionen-Kollisionen kommt es ab einer ausreichend hohen Stoßenergie[1] nicht zu einer elastischen, sondern zu einer inelastischen Streuung. In diesem Streuprozess reicht die Bindungsenergie durch die starke Wechselwirkung zwischen den Konstituenten der Nukleonen (Quarks und Gluonen) nicht mehr aus, um einer Trennung von Quarks und Gluonen entgegenzuwirken.

[1] Vgl. [10].

Nun ist entscheidend, dass Quarks und Gluonen eine Farbladung aufweisen, weshalb sie einzeln, also als ein freies Teilchen, nicht vorkommen können. Da jedoch die hohe Energie eine genügende räumliche Trennung beider Teilchen hervorruft, ist ein Teilchenzerfall die Folge. Bei diesem als Deconfinement bezeichneten Prozess entstehen Hadronen, die farbneutral sind und damit frei existieren können. Diese Hadronisierung soll auch im frühen Universum stattgefunden haben, womit die Untersuchung dieses Prozesses bzw. des in den ursprünglich gestreuten Partonen vorkommende Impulses auch für die Kosmologie relevant ist.

2.1.2 Jets

Mit genug Energie können aus den zuvor entstandenen Teilchen weitere Teilchen entstehen, die sich in die Richtung der ursprünglich gestreuten Partonen ausbreiten. Nun erhält man bei einer experimentellen Messung zunächst nur die bis zum Detektor entstandenen Teilchen. Diese Teilchen werden über einen Jetfinder nach bestimmten Kriterien wie dem Jetradius als Teil eines Jets zugeordnet.
Die dabei erkannten Teilchen sind diejenigen Konstituenten, welche den neuronalen Netzen in dieser Arbeit mit mehreren Variablen übergeben werden. Die hier betrachteten Variablen[2] sind die Pseudorapidität $\eta_{Konst.}$, der Transversalimpuls $p_{T_{Konst.}}$ sowie der Azimuthalwinkel $\phi_{Konst.}$.
Die Pseudorapidität ist eine dimensionslose Größe, welche als

$$\eta = -\ln\left(\tan\left(\frac{\theta}{2}\right)\right) \tag{1}$$

mit dem Polarwinkel θ definiert ist. Sie geht für geringe Ruhemassen der betrachteten Teilchen aus der Rapidität

$$y = \operatorname{artanh}\left(\frac{v}{c}\right) \tag{2}$$

mit der Teilchengeschwindigkeit v und der Lichtgeschwindigkeit c hervor.
Während ϕ den Winkel auf der Normalenebene zur Strahlenrichtung darstellt, ist η der Winkel auf der Ebene entlang der Strahlenrichtung ausgehend von der ϕ-Ebene.

2.2 Künstliche Neuronale Netze (KNN)

Viele Problemstellungen wie z.B. bildbasierte Objektklassifikation und Spracherkennung lassen sich mit zuvor explizit für diese Aufgaben implementierten Algorithmen aufgrund ihrer Komplexität und Variabilität[3] nur schwer oder nicht bewältigen. Wünschenswert wäre hierfür ein selbstständig anhand von Beispieldaten lernendes Programm, welches in der Lage ist, auch komplexe Beziehungen zwischen einer Vielzahl an Inputparametern zu erkennen. Dies gilt nicht nur für die als Beispiel genannten Klassifikationsprobleme, sondern auch für Regressionsprobleme wie die in dieser Arbeit untersuchte Korrektur von Jetimpulsen. Hierbei sind mehrere Daten von manchmal mehr als 100 Konstituenten zu verarbeiten[4].

Genau für derart komplexe Probleme gibt es aus dem Bereich des Machine Learnings künstliche neuronale Netze (KNN). Diese vom biologischen Vorbild des Gehirns inspirierten Systeme besitzen je nach Typ eine unterschiedliche Basisstruktur, deren Eigenschaften im Folgenden anhand eines einfachen Feedforward-Netzes bzw. Perzeptrons und des Convolutional Neural Network (ConvNet oder CNN) beschrieben werden[5], da diese beiden Netztypen in der Arbeit genutzt werden.

[2] Vgl. [11, S. 3ff.].
[3] Man denke hier z.B. an die unterschiedlichen Handschriften und Stimmprofile von Menschen.
[4] Effektiv wird hier eine feste Anzahl
[5] Auch das CNN ist ein Feedforward-Netzwerk mit Perzeptronen, allerdings beinhaltet es weitere Aggregate. Daher der Zusatz "einfach" zur Unterscheidung.

2.2.1 Grundlagen

Die Grundlage eines künstlichen neuronalen Netzes sind künstliche Neuronen.
Ein künstliches Neuron[6] besteht aus mehreren Komponenten, wobei analog zu einem biologischen Neuron eine Ausgabe auf Basis einer Eingabe erfolgt. Zunächst werden in einem Neuron die Outputs anderer Neuronen, die eine Verbindung zu diesem Neuron haben, über die Propagierungsfunktion miteinander zu einem einzelnen Input für das Neuron verrechnet. Dabei wird sehr oft[7] eine Funktion verwendet, welche die Linearkombination aller Inputs mit zugehörigen Gewichten als Koeffizienten berechnet. Das dabei erhaltene Ergebnis wird als Netzinput bezeichnet.

Nun wird dieser Netzinput an die Aktivitätsfunktion weitergeleitet, welche ein zugehöriges Aktivitätslevel ausgibt. Beispiele für Aktivitätsfunktionen sind die Identitätsfunktion, Binärfunktion (Stufenfunktion), Normalverteilung oder der Tangens Hyperbolicus. Die Nutzung eines Schwellenwertes wie bei der Binärfunktion kann u.a. Vorteile für eine Rauschunterdrückung haben, weil hierbei Netzinputs unterhalb der Schwelle keinen Beitrag zu den folgenden Neuronen leisten[8].

Auf die Aktivitätsfunktion folgt abschließend die Berechnung des Outputs aus dem erhaltenen Aktivitätslevel, sofern zwischen Output und Aktivitätslevel überhaupt unterschieden wird[9]. Dies erfolgt meist über die Verwendung der Identitätsfunktion.

Der finale Output eines Neurons wird nun mit vollständiger *ausgehender* Gewichtung an die verbundenen Neuronen weitergeleitet und der beschriebene Rechenprozess setzt sich in diesen Neuronen mit allen Inputs fort.

Der Lernprozess des Netzes besteht nun in der stetigen Anpassung der Gewichtungen zwischen den einzelnen Neuronen. Dabei gibt es grundsätzlich verschiedene Ansätze, wie die Gewichtungen optimiert werden und in welcher Form das Netz mit den Inputs und Outputs konfrontiert werden soll.
Soll das Netz über die konkrete Vorgabe von Input-/Output-Paaren die Beziehung zwischen beiden Größen erlernen, handelt es sich um ein Problem des überwachten Lernens (supervised learning). Darüber hinaus gibt es das bestärkende Lernen (reinforcement learning), bei welchem nicht der konkrete Output angegeben wird, sondern eine Information, ob das generierte Ergebnis korrekt ist oder nicht. Man setzt hierbei also auf ein künstliches Belohnungssystem. Beim unüberwachten Lernen (unsupervised learning) hingegen wird der Ansatz verfolgt, dass das Netz selbstständig Strukturen innerhalb der Inputs erkennt und dabei kein Output vorgegeben wird[10].
Die in dieser Arbeit verwendete Methode ist das überwachte Lernen, da insbesondere die für das Lernen benötigten Daten mit konkret zugeordneten Transversalimpulsen generiert werden können und das gewünschte Ergebnis somit stets exakt vorgegeben werden kann.

Die Optimierung der Gewichtungen erfolgt hierbei über Lernregeln, die u.a. davon abhängig ist, ob neben der Input-/Output-Schicht sog. Hidden-Schichten vorhanden sind.

[6] Sofern nicht anders angegeben bezieht sich "Neuron" ab jetzt auf ein künstliches Neuron.
[7] Vgl. [1, S. 17].
[8] Vgl. [1, S. 22].
[9] Vgl. [1, S. 24]
[10] Dieses Vorgehen kann z.B. dazu verwendet werden, um Strukturen in Bildern ohne vorgegebene Zuweisungen selbstständig zu segmentieren.

2.2.2 Perzeptron

Ein einschichtiges Perzeptron besteht aus nur einer einzigen trainierbaren Schicht mit Neuronen, die die Elemente des Inputvektors voll vernetzt, d.h. an alle Neuronen gerichtet, erhalten. Diese Schicht verarbeitet die Daten und stellt zugleich die Output-Schicht dar. Das Lernen kann hierbei über die sog. Delta-Regel erfolgen, bei welcher die Gewichte direkt über die Differenz zwischen ermittelten und wahren Ausgaben optimiert werden.
Bei der Einführung weiterer Hidden-Schichten werden die Outputs der Neuronen ebenso voll vernetzt an die darauffolgende Schicht weitergegeben. Man spricht dann von einem mehrschichtigen Perzeptron. Damit ist der Lernvorgang komplexer als für eine einzelne Schicht. Der Durchbruch zum effizienten Trainieren derartiger Netzstrukturen gelang erst mit der Entwicklung der sog. Backpropagation, die auf einer Erweiterung der Delta-Regel basiert. Das Grundprinzip ist hierbei, dass nach der Ermittlung des Netzfehlers (Differenz zwischen Output und zu erzielenden Output) beginnend bei der Output-Schicht über die Hidden-Schichten zurückpropagiert wird, wobei eine Anpassung der jeweiligen Gewichte erfolgt[11].
Je mehr Schichten ein solches Netz aufweist, desto "tiefer" ist es, was den Begriff des "Deep Learnings" im Bereich des Machine Learnings geprägt hat.

In dieser Arbeit wird das mehr- oder einschichtige Perzeptron in Form eines einfachen Feedforward-Modells als FF deklariert.

2.2.3 Convolutional Neural Network (CNN)

In manchen Fällen ist es hilfreich, die räumliche Struktur der Inputdaten in die Auswertung derselben miteinzubeziehen. Man bedenke hierbei z.B., dass verrauschte Strukturen oft eine hohe Raumfrequenz aufweisen. Dafür bieten sich Convolutional Neural Networks an, da dieser Netztyp eben diese räumlichen Informationen berücksichtigt und bei entsprechender Modellierung für das Ziel irrelevante Informationen herausfiltern kann.
Prinzipiell handelt es sich hierbei zunächst um ein Feedforward-Modell, welches meist aus einer Folge von Konvolutions- (bzw. Faltungs-) und Pooling-Schichten besteht.

Für die Konvolutionsschicht durchläuft zunächst ein Filterkernel vorgegebener Größe den gegebenen Input, der z.B. bei einem Grauwert-Bild zweidimensional sein kann. Dieser für die Schicht gleiche Filter bildet das (einfache) Skalarprodukt der darin eingetragenen Gewichtungen und gibt die erhaltenen Werte je an ein Neuron weiter. Dies ist dann die Konvolutionsschicht.
Bisher beinhaltet diese Schicht jedoch womöglich irrelevante Informationen. So könnten beispielsweise beim Versuch der Identifikation eines Objektes in einem Bild alle Informationen in einem bestimmten Bildbereich unbedeutend sein, da sich das Objekt an einer räumlich begrenzten Stelle befindet. Bei der Anwendung auf das Thema dieser Arbeit *könnten* z.B. auch die Beiträge von Konstituenten mit einem bestimmten Transversalimpulsbereich nach einem bestimmten erlernten Muster vernachlässigbar sein.

Zum Filtern eben dieser unbedeutenden Bereiche bzw. Signale bietet sich eine Pooling-Schicht an, die einen Filter definierter Größe durch die Konvolutionsschicht in vorgegebenen Schritten durchlaufen lässt und dabei z.B. nur den maximalen Beitrag der im Pooling-Kernel durchlaufenen Aktivitäten in der Konvolutionsschicht zurückgibt.
Nach Wiederholung dieser Schichtreihenfolge wird das Endergebnis über ein Perzeptron weitergegeben, wobei eine Flattening-Schicht zur Dimensionsreduktion zur Anwendung kommen kann. Sog. Dropout-Schichten, die auch im normalen Perzeptron vorkommen können, deaktivieren mit einer bestimmten Wahrscheinlichkeit einen Teil der Neuronen, um eine Optimierung auf die

[11] Für mehr Informationen sei auf [1, S. 50 ff.] verwiesen.

Trainingsdaten (Overfitting) zu verhindern.

Alle Modelle diesen Typs werden in dieser Arbeit als CNN bezeichnet.

3 Simulation und Analyse der genutzten Daten

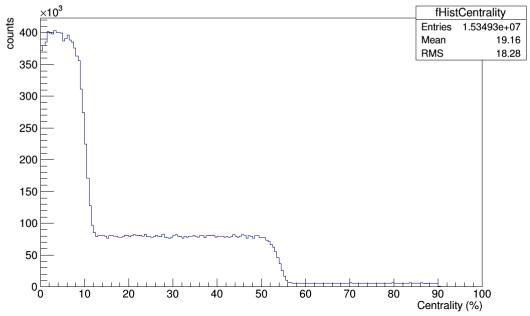
Die den künstlichen neuronalen Netzen zur Verfügung gestellten Daten bestehen aus 80000 Trainings-, 20000 Validierungs- und 15000 Testdatensätzen[12].
Als ein Datensatz wird hierbei die Gesamtheit an (verarbeiteten) Daten eines Jets bezeichnet. Diese Jets entstammen einer von Florian Jonas in Zusammenarbeit mit Dr. Rüdiger Haake entwickelten Simulation, die zunächst beschrieben wird.

3.1 Simulation

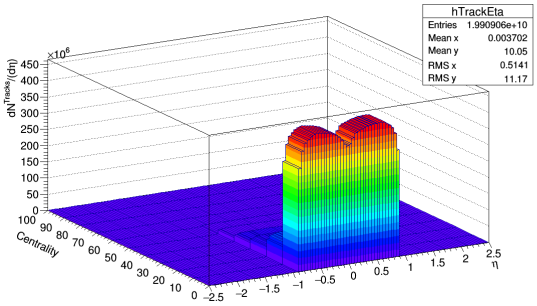
Die Grundidee[13] der genutzten Simulation einer PbPb-Schwerionenkollision ist die Kombination einer Proton-Proton-Kollision mit einem Hintergrund aus Pionen (π^+, π^-).
Hierfür werden zunächst mit dem Eventgenerator von PYTHIA 8 pp-Kollisionsevents generiert, wofür eine Schwerpunktsenergie von $\sqrt{s} = 2{,}76\,\text{TeV}$ angesetzt wird. Diese Energie ist (u.a.) in Experimenten mit Blei-Blei-Schwerionenkollisionen im ALICE-Experiment am LHC möglich.
Nun muss noch der Pionenhintergrund generiert werden. Dafür wird im Rahmen eines Toy-Modells zunächst das aus der PYTHIA-Simulation erhaltene pp-Event mit π^+- und π^--Teilchen (50%/50%) ergänzt. Die Anzahl an hinzugefügten Teilchen richtet sich nach Daten aus der Verteilung eines *realen* PbPb-Kollisionsevents im ALICE-Experiment. Dabei wird die Anzahl an Teilchen anhand eines normierten und als Wahrscheinlichkeit interpretierten Histogramms im relevanten Zentralitätsbereich (vgl. Abb. 1(a)) von insbesondere 0% bis 10% gewürfelt.

Analog wird auch für η und p_T der Teilchen verfahren (vgl. Abb. 1(b) und 1(c)). Für ϕ wird allerdings ein manuelles Vorgehen gewählt[14]. Hier muss berücksichtigt werden, dass die Verteilung dieser Variablen aus den experimentellen Daten nicht genutzt werden kann, da dort nicht auf die Verteilung für ein einzelnes Event zugegriffen werden kann, sondern für mehrere Millionen Events.
Die Teilchen ergänzen nun mit allen hinzugefügten Eigenschaften unsere bisher aus PYTHIA erhaltenen Teilchen.
Der Radius des Jetfinders beträgt 0,3.
Wichtig anzumerken ist, dass der Jetfinder zunächst nur im pp-Event von PYTHIA Jets identifiziert und danach nochmal die zusammengeführten Events von dem Toy-Modell *und* PYTHIA durchläuft, um auf Basis dieser neuen Daten Jets zu erkennen.

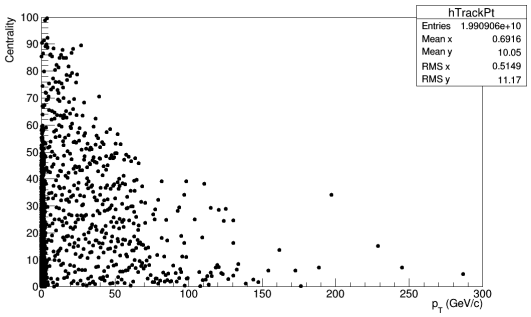
[12] Testdatensätze werden nur für die Angabe einer finalen Performance verwendet und nicht für Auswahl der Modellparameter. Damit wird ausgeschlossen, dass die am Ende bestimmte Vorhersagequalität durch die Modellauswahl auf die Datensätze zur Validierung optimiert wurde.
[13] Das hier dargelegte Vorgehen orientiert sich an Beschreibungen von Florian Jonas ([14]).
[14] Hierfür sei für weitere Informationen auf [13] verwiesen.



(a) Verteilung der Zentralität.



(b) Verteilung von η in Abhängigkeit der Zentralität.

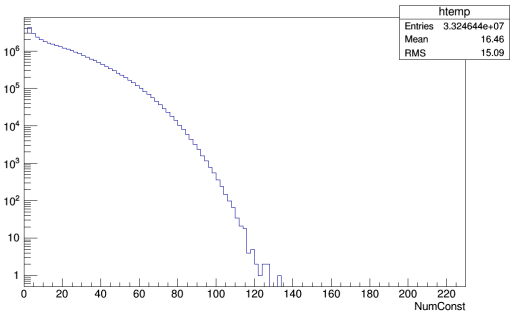


(c) Verteilung der Zentralität in Abhängigkeit von p_T.

Abbildung 1: Darstellung der Verteilungen von η und p_T in Abhängigkeit der Zentralität und die Häufigkeitsverteilung der Zentralität aus ROOT.

3.2 Darstellung der Input-Daten

Die dem jeweils genutzten Netz zunächst zur Verfügung gestellten Daten von Konstituenten pro Jet ($p_{T;Konst}$, η_{Konst}, ϕ_{Konst}) sind auf die ersten nach $p_{T;Konst}$ absteigend sortierten 140 Konstituenten beschränkt. Dies ist für die genutzte p_T-Verteilung ausreichend, wie in Abb. 2(a) zu erkennen ist.

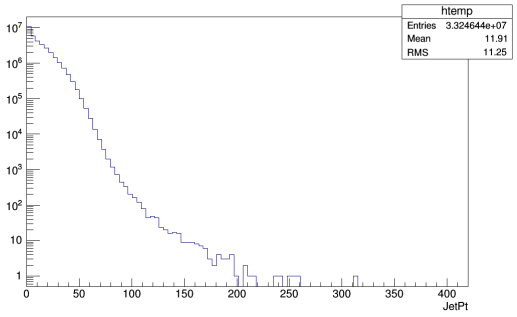


(a) Verteilung der Konstituenten-Anzahl NumConst. Mit 140 Konstituenten wird der gesamte Bereich der Daten abgedeckt.

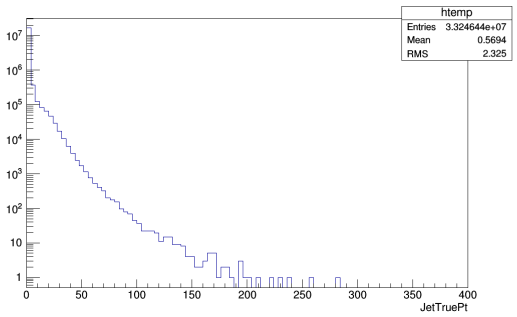
Abbildung 2: Darstellung der Verteilungen der Daten für die Anzahl an Konstituenten aus ROOT.

Des Weiteren sind die genutzten Datensätze auf ein $p_{T;Real}$ von $20\,\text{GeV} \leq p_{T;Real} \leq 150\,\text{GeV}$ eingeschränkt, um die Modelle für einen Impulsbereich mit vertretbar vielen Datensätzen zu

trainieren[15]. Die zugehörigen Häufigkeiten für das messbare p_T und das aus der Simulation bekannte $p_{T;True}$ der Jets sind Abb. 3 zu entnehmen.



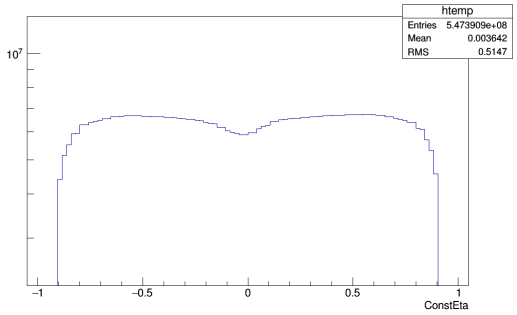
(a) Verteilung für das durch Addition der $p_{T\,Konst.}$ erhaltene $p_{T\,Jet}$.



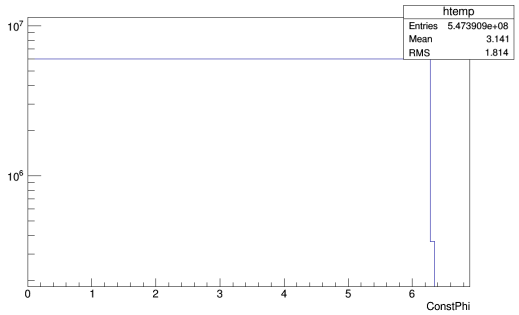
(b) Verteilung für das durch Addition der Transversalimpulse aus dem reinen pp-Event erhaltene $p_{T;True}$ des Jets.

Abbildung 3: Darstellung der Häufigkeitsverteilungen der Daten für das zu berechnende $p_{T;True}$ und das einfach errechnete p_T der Jets aus ROOT.

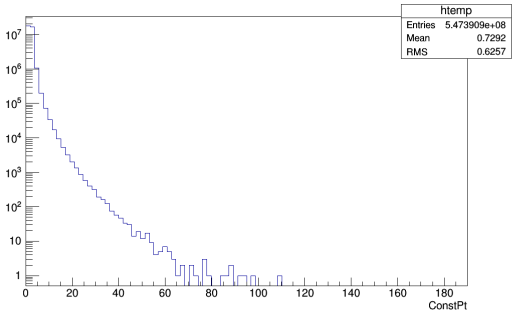
Für die Konstituentendaten ergeben sich die Abb. 4 dargestellten Verteilungen.



(a) Verteilung von $\eta_{Konst.}$.



(b) Verteilung von $\phi_{Konst.}$.



(c) Verteilung von $p_{T_{Konst.}}$.

Abbildung 4: Darstellung der Häufigkeitsverteilungen der Daten für η, ϕ und p_T der Konstituenten aus ROOT.

[15] Für eine Anwendung auf realen Daten kann diese Einschränkung auch auf einen entsprechenden p_T-Bereich der Messungen ohne Wissen des $p_{T;True}$ angepasst werden.

4 Suche nach dem besten Modell

Zur Ermittlung des im Rahmen der untersuchten Parameterräume bestmöglichen Modells werden im Folgenden die durchgeführten Untersuchungen zu einfachen Feedforward-Modellen im Rahmen eines Perzeptrons (FF) und eindimensionalen CNNs (CNN) dargelegt.
Nach einer ersten Rastersuche mit Parametern, deren Einfluss auf den Netztypen als grundlegend eingeschätzt wurden, folgte eine weitere Rastersuche mit anderen Parameterräumen vor dem Hintergrund der bisher erkannten Abhängigkeit der Modellperformance von bestimmten Hyperparametern. Zuletzt wurde das bis dahin je beste Modell über eine dritte Rastersuche u.a. mit zuvor fixierten Parametern feinjustiert.

Wichtig ist hierbei, dass die zu den einzelnen Rastersuchen gezeigten Abhängigkeiten der Performance von Parametern stets im Kontext der untersuchten Parameterräume bewertet werden müssen, da eine generelle Aussage zum Verhalten der Performance aufgrund nichtlinearer Verhaltensweisen (z.B. kritische Punkte) und im Hinblick der beschränkten Rechenzeit nicht möglich ist. Damit sind die identifizierten und formulierten Tendenzen *stets* an die Parameterräume der Rastersuchen gebunden, werden aber als Anhaltspunkt für Verbesserungen der Modelle verwendet.

Für die Erstellung, das Training und die Analyse aller Modelle wurde eine von mir modifizierte Version des AliMLManager-Frameworks von Rüdiger Haake benutzt, welches auf dem Deep Learning Framework "Keras" für Python basiert. Als Backend wurde Googles TensorFlow verwendet. Das genutzte Trainings- und Testsystem hat einen Intel Core i7-7700K @ 4,20Ghz als CPU und eine Nvidia Quadro M2000 als GPU, wobei die Netze auf der GPU trainiert und getestet wurden.

4.1 Fixierte Hyperparameter für beide Netztypen

Da die Suche nach einem besten Modell durch Rastersuche insbesondere zeitlichen Beschränkungen unterliegt, ist die Fixierung von einigen Parametern mit Werten (vgl. Tab 1), die auf eigener oder fremder Erfahrung beruhen und deren Einfluss relativ z.B. zur Anzahl von Neuronen oder der Größe von Konvolutionsfiltern als gering eingeschätzt wird, zwingend erforderlich.
Außerdem bedarf es der Vergleichbarkeit wegen für beide Netztypen eines gleichen Qualitätsmaßes für die erhaltenen Modelle. In der gesamten Arbeit ist hierfür die Loss-Funktion, welche den Fehler der Vorhersagen eines Modells angibt nach dessen Minimierung auf den Trainingsdatensätzen das Netz optimiert wird, die mittlere quadratische Abweichung (MSE). Dieses Qualitätsmaß ist v.a. im Hinblick auf der in den genutzten Daten geringeren Anzahl an Jets mit hohen p_T sinnvoll, weil damit große absolute Abweichungen, wie sie v.a. bei hohen Werten der zu bestimmenden Größe zu erwarten sind, höher gewichtet werden als kleine absolute Abweichungen.
Zur besseren Interpretation der erhaltenen Modellperformance als p_T-Abweichung wird im Folgenden bei der Bewertung der Modelle jedoch die Wurzel der mittleren quadratischen Abweichung (RMSE) genutzt.

Hyperparameter	**Wert**	**Anmerkungen**
Gewicht-Initialisierungsfkt.	"he_uniform"	Vgl. [3].
Optimierungsalgorithmus	"Adam"	Vgl. [4].
Aktivierungsfunktion	tanh	Dies bezieht sich nicht auf das finale Netz, welches das Ergebnis ausgibt.
Loss-Funktion	"mean_squared_error"	Das neuronale Netz wird über den mittleren quadratischen Fehler bewertet und über seine Minimierung optimiert.
Stapelgröße (Batch Size)	512	Neben einer Verringerung der Rechenzeit gegenüber Ansätzen mit sehr geringen Stapelgrößen bietet allg. die Nutzung von Datenstapeln die Möglichkeit, die Hyperparameter des Netzes höher frequent zu optimieren, was mit der effizienten Nutzung der Datenbasis einhergeht. Zu große Stapelgrößen können die Fähigkeit zur Generalisierung auf unbekannte Daten verringern (vgl. [5]).
Lernrate	$r_l = 0{,}0001$	Mit dieser relativ geringen Lernrate[a] soll ein genaue Optimierung der Gewichte ermöglicht werden. Dieser Parameter ist unmittelbar mit der genutzten Optimierungsfunktion verknüpft[b].
Anzahl Epochen	$N = 100$	Für die Ermittlung einer sinnvollen Anzahl an Epochen wurde die Entwicklung der Modellperformance mit steigender Epoche betrachtet. Die Wahl von N muss i. Allg. v.a. mit der Wahl der Lernrate abgestimmt werden, um insbesondere einen zu frühen Abbruch der Optimierung zu verhindern. $N = 100$ stellt einen Kompromiss zwischen benötigter Rechenzeit für mehr Epochen und der Investigation eines größeren Parameterraumes dar.

Tabelle 1: Überblick über alle in den Rastersuchen fixierten Hyperparameter.

[a] Nach [12][S. 94f.] befinden sich nach "Erfahrung" gute Lernraten sogar zwischen 0,01 und 0,9.
[b] Für eine genaue Definition vgl. [4].

Bei jedem Fit der Modelle auf die Trainingsdaten werden die Inputdaten je Epoche zufällig durchmischt, um eine Spezialisierung des Modells auf die ursprüngliche Verteilung der Daten zu verhindern.

4.2 Auftretende Unsicherheiten und Entscheidungskriterien

Da sich die erhaltenen RMSE-Werte teilweise erst auf der dritten Nachkommastelle unterscheiden und die Wahl des vorerst besten Modells für weitere Rastersuchen sehr entscheidend ist, ist eine angemessen gewählte Unsicherheit für die erhaltene Modellperformance von höchster Relevanz. Um die auftretenden Schwankungen der Modellperformance[16] einschätzen zu können, wurden je Netztyp die Schwankungen des RMSE für ein Beispielmodell untersucht. Hierbei wird für eine angemessene Sicherheit der die Standardabweichung für einen 99,8%igen Signifikanzbereich als Referenz betrachtet.
Ein mehrfaches Berechnen jedes Modells wird dafür in Anbetracht der dafür vervielfachten Rechenzeit, die mit einer signifikanten Verringerung der untersuchten Parameterräume einhergehen würde, nicht durchgeführt. Aus dem gleichen Grund sowie der identischen Verteilung in den Trainings- und Validierungsdatensätzen wird im Übrigen die oft empfohlene k-fache Kreuzvalidierung[17] nicht genutzt.

Die Wahl des besten Modells hängt in dieser Arbeit demnach auch von der in allen Modellen auftretenden Tendenz des RMSE in Abhängigkeit bestimmter Parameter ab. Dies bedeutet *nicht*, dass ein bestimmtes Modell für eine kritische Kombination an einzelnen Parametern, die der mittleren Tendenz des RSME widersprechen, nicht besser sein könnte. Immerhin handelt es sich prinzipiell um eine multidimensionale Suche nach Minima. Jedoch muss der Unterschied der Performance zu Modellen, die der allgemeinen Tendenz aller betrachteten Modelle entsprechen, signifikant sein.

Ein weiterer wichtiger Punkt ist, dass zwar für das RMSE auf den Trainings- und Validierungsdatensätzen mit der Unsicherheit auf Basis der Modellfluktuation angegeben wird, dies jedoch nicht bei dem Vergleich zwischen Trainings- und Validierungs-RMSE im *gleichen* Modell anzuwenden ist. Dies hängt damit zusammen, dass sich die Unsicherheit nur aus der durch den Lernprozess entstandenen Modellfluktuationen ergibt, nicht jedoch für die Ergebnisse auf den Validierungsdatensätzen im gleichen Modell. Beide Werte sind somit für das identische Modell korreliert.

Für die Unsicherheit mittlerer Werte von nach Kriterien zusammengefassten Modellen wird eine Signifikanz von 95% über die Standardabweichung mit den der Stichprobenanzahl angepassten Werten der Studentschen t-Funktion angesetzt.

4.3 Ergebnisse zum einfachen Feedforward-Netzwerk

Nach einer kurzen Beschreibung der Modellstruktur (Kap. 4.3.1) folgt nach der Bestimmung der Modellfluktuation (Kap. 4.3.2) eine erste Rastersuche (Kap. 4.3.3), in welcher das am besten eingestufte Modell für weitere Rastersuchen gefunden und das Verhalten der Modellperformance bei Variation bestimmter Hyperparameter dargestellt wird. Mit der zweiten Rastersuche (Kap. 4.3.4) werden die erhaltenen Abhängigkeiten für die Modifikation oder Umstrukturierung des Netzes genutzt, um darauf aufbauend nach detaillierten Suchen (Kap. 4.3.5) das final beste (einfache) Feedforward-Modell (Kap. 4.3.6) ermitteln.

[16] Trotz Fixierung des Anfangswertes für die in Tensorflow und Numpy genutzten Zufallsgeneratoren konnte die Reproduzierbarkeit auch bei mehrmaliger Ausführung des gleichen Programms (und damit je gleichen Reihenfolgen z.B. für die Generierung der Anfangsgewichte) nicht gewährleistet werden.

[17] Die Datensätze werden in einem über k definierten Verhältnis in Validierung- und Trainingsdatensätze geteilt und das Netz k Mal auf den unterschiedlichen Datensätzen trainiert.

4.3.1 Allgemeine Modellstruktur

Die grundlegende Modellstruktur besteht aus drei "Zweigen", d.h. drei Perzeptron-Netze, die jeweils nur $\eta_{Konst.}$, $\phi_{Konst.}$ oder $p_{TKonst.}$ von maximal 140 Konstituenten als Input zur Verfügung gestellt bekommen. Sollte die Anzahl an Konstituenten geringer als 140 sein, werden die restlichen Inputwerte mit Null besetzt (zero-padding). Diese einzelnen Netze haben eine Anzahl von Neuronen $n_{\eta,i}, n_{\phi,i}$ und $n_{p_T,i}$ (bzgl. i-ter Schicht) sowie eine Anzahl von Schichten l_η, l_ϕ und l_{p_T}.

Zur Zusammenfassung aller Ergebnisse aus den drei Zweigen wird ein weiteres finales Feedforward-Netz verwendet, welches l_F Schichten und $n_{F,j}$ (bzgl. j-ter Schicht) Neuronen aufweist und das zu ermittelnde $p_{T;True}$ ausgeben soll.
Ein Überblick der Modellstruktur mit allen explizit gesetzten Parametern in Keras ist zur Reproduzierbarkeit in Tab. 2 dargestellt.

Bezeichnung	**Gesetzte Parameter (Keras)**
Dense	• units = n_{F_i} • kernel_initializer = "he_uniform" • activation = "tanh" • kernel_regularizer= $l2(0{,}001)$
Dropout	• rate = r_{d_1}
Dense	• units = n_{F_j} • kernel_initializer = "he_uniform" • activation = "relu" • kernel_regularizer= $l2(0\ 001)$
Dropout	• rate = r_{d_2}
Dense	• units = 1 • activation = "linear"

Tabelle 2: Darstellung der genutzten Schichten mit zugehörigen Parametern für die untersuchten FF-Modelle. Für die Schichten der Zweignetze wurde der Index i, für die finalen Schichten der Index j verwendet.

4.3.2 Bestimmung der Modellfluktuation

Als Beispielmodell zur Bestimmung der Modellfluktuation wurde ein FF-Modell mit $l = l_\eta = l_\phi = l_{p_T} = l_F = 1$, $n = 32$, $n_F = 64$, $r_d = 0{,}2$ verwendet, wobei die Basisstruktur nach Kap. 4.3.1 gegeben ist und die sonstigen Hyperparameter nach Kap. 1 gesetzt wurden.
Die dabei erhaltenen Ergebnisse sind in Tab. 3 zusammengetragen. Das mittlere RMSE liegt für die Validierungsdatensätze bei $3{,}66 \pm 0{,}07$ (99,8% Signifikanz) und hat damit eine relative Unsicherheit von etwa 2,0%, die als Richtwert für folgende Modelle genutzt wird.

Durchgang	**RMSE (Val.)**	***RMSE (Train.)***
1	3,5897	*3,5892*
2	3,5873	*3,5893*
3	3,7196	*3,7104*
4	3,7130	*3,7094*
5	3,7152	*3,7061*
6	3,6177	*3,6186*
7	3,6417	*3,6331*
8	3,6104	*3,6019*
9	3,6776	*3,6671*
10	3,7214	*3,7090*

Tabelle 3: Darstellung der Modellfluktuation anhand des gewählten Beispielmodells mit $l = 1$, $n = 32$, $n_F = 64$ und $r_d = 0{,}2$. Es zeigen sich deutliche Unterschiede zwischen den einzelnen Durchläufen, womit keinem Modell eine Performance ohne Unsicherheit zugewiesen werden kann.

4.3.3 Erste Rastersuche

4.3.3.1 Betrachteter Hyperparameterraum Zwecks Verkleinerung des Parameterraumes wird die Anzahl aller Neuronen in den Zweignetzen als $n = n_\eta = n_\phi = n_{p_T}$ gleichgesetzt. Dafür wird die Anzahl aller Schichten in den Zweignetzen variiert (vgl. Tab. 4).

Hyperparameter	**Symbol**	**Untersuchte Werte**
Anzahl Neuronen (Zweignetze)	n	$8; 16; 32$
Anzahl Neuronen (finales Netz)	n_F	$32; 64; 128$
Anzahl Schichten (Zweignetze)	l_η, l_ϕ, l_{p_T}	$1; 2$
Anzahl Schichten (finales Netz)	l_F	$1; 2$
Dropout-Rate	r_d	0,2

Tabelle 4: Überblick über alle in der ersten Rastersuche variierten Hyperparameter für das Feedforward-Netz.

Es sollte bei der Betrachtung von n und n_F stets bedacht werden, dass an das finale Feedforward-Netz $3 \cdot n$ Inputs weitergeleitet werden. Daher wurde n_F auch etwas größer als n gewählt, wobei über die Rastersuche auch Modelle mit $n_F < 3 \cdot n$ berücksichtigt werden konnten.
Als Dropout-Rate zur Verhinderung von Overfitting wurde $r_d = r_{d_1} = r_{d_2} = 0{,}2$ gewählt. Dies ist relativ niedrig[18], wird jedoch zunächst angesetzt, da die anfänglich untersuchten Modelle keine große Anzahl an Neuronen und Schichten besitzen, womit ihre Anfälligkeit für Overfitting verringert sein sollte.

[18] Nach [6] sollte die Dropout-Rate sogar bei 50% liegen, in [7] werden 20% und 50% genannt.

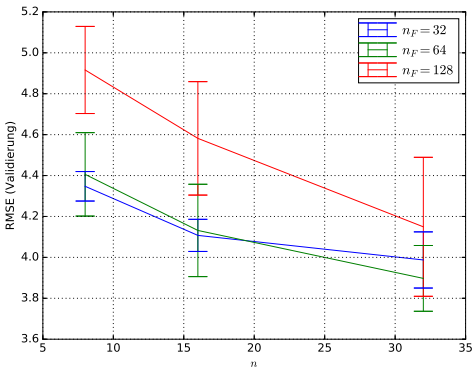
4.3.3.2 Ergebnisse Die bei der ersten Rastersuche erhaltenen Ergebnisse sind in Tab. 5 einzusehen. Das beste Modell befindet sich im Rahmen der abgeschätzten Modellunsicherheit demnach bei Rang 1 bis Rang 17.
Die in der ersten Rastersuche (A) erhaltenen Modelle werden über die in der Ergebnistabelle dargestellte Parameterreihenfolge als $FF_{A[Rang]}(n, l, l_\eta, l_\phi, l_{p_T}, n_F, l_F)$ oder nur über ihren Rang als $FF_{A[Rang]}$ bezeichnet.
Positiv anzumerken ist zunächst, dass keines der untersuchten Modelle signifikantes Overfitting zeigt, da die RMSE auf den Validierungs- und den Trainingsdatensätzen nahezu identisch sind.

Rang	n	l_η	l_ϕ	l_{p_T}	n_F	l_F	**RMSE Val.**	***RMSE Train.***
1	32	1	2	1	128	1	$3{,}53 \pm 0{,}07$	$3{,}54 \pm 0{,}07$
2	32	1	1	1	128	1	$3{,}55 \pm 0{,}07$	$3{,}55 \pm 0{,}07$
3	32	2	1	1	128	1	$3{,}56 \pm 0{,}07$	$3{,}56 \pm 0{,}07$
4	32	2	1	2	128	1	$3{,}56 \pm 0{,}07$	$3{,}57 \pm 0{,}07$
5	32	2	2	2	128	1	$3{,}56 \pm 0{,}07$	$3{,}56 \pm 0{,}07$
6	32	2	1	2	64	1	$3{,}57 \pm 0{,}07$	$3{,}57 \pm 0{,}07$
7	32	1	2	2	128	1	$3{,}58 \pm 0{,}07$	$3{,}57 \pm 0{,}07$
8	32	2	1	1	32	2	$3{,}58 \pm 0{,}07$	$3{,}58 \pm 0{,}07$
9	32	1	1	2	64	1	$3{,}59 \pm 0{,}07$	$3{,}58 \pm 0{,}07$
10	32	1	1	1	32	2	$3{,}60 \pm 0{,}07$	$3{,}59 \pm 0{,}07$
11	32	2	2	2	32	2	$3{,}60 \pm 0{,}07$	$3{,}58 \pm 0{,}07$
12	32	1	1	1	64	1	$3{,}60 \pm 0{,}07$	$3{,}59 \pm 0{,}07$
13	32	2	1	1	64	1	$3{,}60 \pm 0{,}07$	$3{,}60 \pm 0{,}07$
14	32	2	2	1	128	1	$3{,}61 \pm 0{,}07$	$3{,}61 \pm 0{,}07$
15	32	1	1	2	128	1	$3{,}61 \pm 0{,}07$	$3{,}60 \pm 0{,}07$
16	16	1	1	1	64	1	$3{,}63 \pm 0{,}07$	$3{,}63 \pm 0{,}07$
17	16	2	1	1	64	1	$3{,}65 \pm 0{,}07$	$3{,}64 \pm 0{,}07$
18	32	1	2	1	64	1	$3{,}67 \pm 0{,}07$	$3{,}66 \pm 0{,}07$
19	32	1	2	2	64	1	$3{,}69 \pm 0{,}07$	$3{,}67 \pm 0{,}07$
20	16	2	2	1	64	1	$3{,}70 \pm 0{,}07$	$3{,}69 \pm 0{,}07$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
140	8	2	2	2	128	2	$5{,}26 \pm 0{,}11$	$5{,}22 \pm 0{,}10$
141	16	2	1	2	128	2	$5{,}30 \pm 0{,}11$	$5{,}28 \pm 0{,}11$
142	8	2	1	2	128	2	$5{,}43 \pm 0{,}11$	$5{,}38 \pm 0{,}11$
143	8	1	1	2	128	2	$5{,}43 \pm 0{,}11$	$5{,}39 \pm 0{,}11$
144	8	1	2	2	128	2	$5{,}71 \pm 0{,}11$	$5{,}67 \pm 0{,}11$

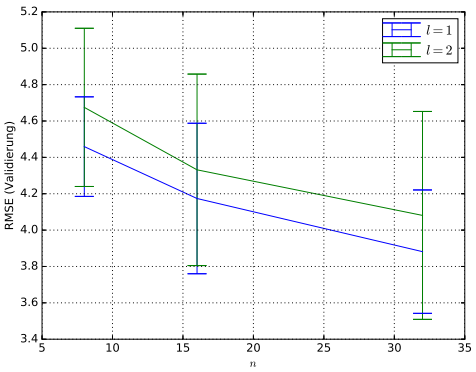
Tabelle 5: Ergebnisse der besten 20 und schlechtesten 5 Modelle für die erste Rastersuche (A) nach dem RMSE auf den Validierungsdatensätzen aufsteigend sortiert. Das mittlere RMSE (Val.) liegt bei $4{,}28 \pm 0{,}08$.

Für die Entwicklung einer Strategie zur Parameterwahl in folgenden Rastersuchen wird nun das Verhalten der Modellperformance in Abhängigkeit mehrerer Parameter untersucht.
Wie in Abb. 5(a) zu erkennen ist, führt eine Erhöhung der Neuronenanzahl in den Zweignetzen für alle genutzten n_F tendenziell zu einer Verbesserung des RMSE. Auffällig ist, dass die Netze mit $n_F = 128$ die größten RMSE (signifikant für $n \in \{8{,}16\}$) aufweisen, wohingegen die Netze mit $n_F \in \{32{,}64\}$ eine etwa gleich bessere Performance zeigen. Dies könnte den Schluss zulassen, dass n für eine Verbesserung des Modells nur erhöht und n_F verringert werden muss. Hierbei sei jedoch angemerkt, dass der durchschnittliche RMSE aller Modelle für $n_F \geq 3n$ (Anzahl aller Outputs der Zweignetze) mit $4{,}42 \pm 0{,}10$ höher ist als für $n_F < 3n$ mit $4{,}00 \pm 0{,}07$. Demnach

scheint auch das Verhältnis zwischen diesen Größen hier einen Einfluss auf die Modellperformance zu haben.



(a) Ergebnisse für $n_F = 32{,}64{,}128$.

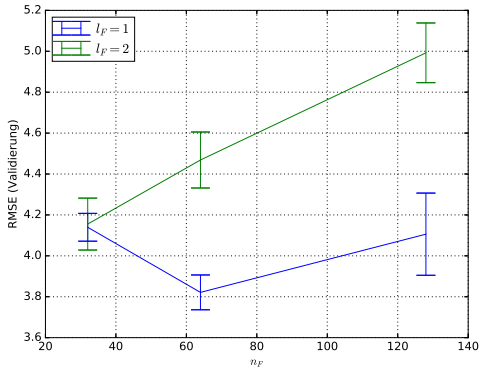


(b) Ergebnisse für $l = 1{,}2$.

Abbildung 5: Abhängigkeit des RMSE für die Validierungsdatensätze von n für unterschiedliche n_F und $l = l_\eta = l_\phi = l_{p_T}$ auf Basis aller 144 Modelle der ersten Rastersuche.

Für eine gleiche Anzahl von Schichten ($l = l_\eta = l_\phi = l_{p_T}$) verbessert sich der durchschnittliche RMSE auch mit steigendem n (vgl. Abb. 5(b)). Zwar sind die Mittelwerte der RMSE für $l = 1$ geringer als für $l = 2$, jedoch sind beide im Unsicherheitsbereich identisch.

Eine sehr deutliche Entwicklung der Modellperformance mit der Anzahl an Schichten im finalen Netz zeigt sich in Abb. 6(a). Das RMSE steigt hierbei mit steigendem n_F für $l_F = 2$, wohingegen sich für $l_F = 1$ bei $n_F = 64$ ein Minimum im untersuchten Parameterraum ergibt. Dies zeigt, dass die Ergebnisse mit einer höheren Anzahl an Schichten im finalen Netz nicht zwingend verbessert werden[19]. Die mittleren RMSE sind hierbei $4{,}02 \pm 0{,}08$ ($l_F = 1$) und $4{,}54 \pm 0{,}11$ ($l_F = 2$).



(a) Ergebnisse für $l_F = 1{,}2$.

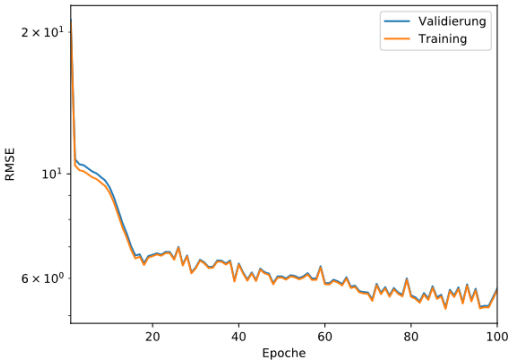
Anders hingegen verhält sich die Modellperformance in Abhängigkeit einzelner Anzahlen von Schichten in den Zweignetzen, welche in Tab. 6 eingetragen sind. Hierbei ergeben sich im Rahmen der Unsicherheit nahezu identische mittlere RMSE, sodass die Anzahl einzelner Schichten von Zweignetzen für den untersuchten Parameterraum keine allgemeine Verbesserung oder Verschlechterung der Performance zur Folge hat.

[19] Diese Aussagen sind stets vor dem Hintergrund des untersuchten Parameterraumes zu verstehen. D.h. anders ausgewählte Parameterräume können möglicherweise andere Tendenzen ergeben.

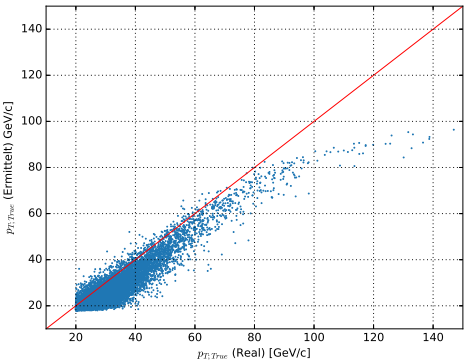
Schichten	1	2
l_η	$4{,}28 \pm 0{,}11$	$4{,}28 \pm 0{,}12$
l_ϕ	$4{,}25 \pm 0{,}11$	$4{,}31 \pm 0{,}12$
l_{p_T}	$4{,}20 \pm 0{,}10$	$4{,}36 \pm 0{,}13$

Tabelle 6: Mittlere RMSE auf den Validierungsdatensätzen in Abhängigkeit der Anzahl an Schichten in den Zweignetzen.

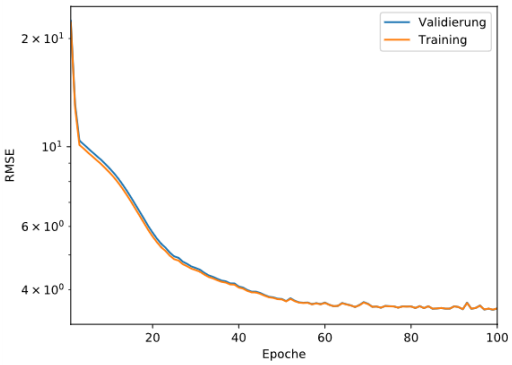
Zur Darlegung, was ein relativ hoher oder niedriger Wert des RMSE für die ermittelten $p_{T;True}$-Werte und damit die Qualität des Modells bedeutet, werden als Beispiel ein Modell mit sehr guter Performance und ein Modell mit schlechter Performance miteinander verglichen.
Ein Vergleich der Ergebnisse der Modelle $FF_{A2}(32; 1; 1; 1; 128; 1)$ und des nach Tab. 5 am schlechtesten vorhersagenden Modells $FF_{A144}(8; 1; 2; 2; 128; 2)$ (vgl. Abb. 6) zeigt deutliche Unterschiede untereinander. Zum Einen weist FF_{A2} nach Abb. 6(d) geringere Fluktuationen des RMSE mit fortlaufender Epoche auf als FF_{A144} in Abb. 6(b). Zum Anderen zeigt sich bei den Streudiagrammen in Abb. 6(c) und 6(e), dass v.a. der Bereich ab einem $p_{T;True}$ von etwa $100\,\frac{\mathrm{GeV}}{\mathrm{c}}$ und insbesondere bei $140\,\frac{\mathrm{GeV}}{\mathrm{c}}$ im schlechteren Modell mit $(90 \pm 10)\,\frac{\mathrm{GeV}}{\mathrm{c}}$ wesentlich zu gering eingeschätzt wird und die Hintergrundkorrektur damit für diesen Impulsbereich fehlschlägt. Bei beiden Modellen werden die Impulswerte bei steigendem Realwert tendenziell zu gering errechnet, was möglicherweise auch auf den Einfluss der in den Datensätzen überproportionalen Anzahl an Jets mit geringem $p_{T;True}$ zurückgeführt werden kann oder mit einer zu geringen Lernzeit zusammenhängt.



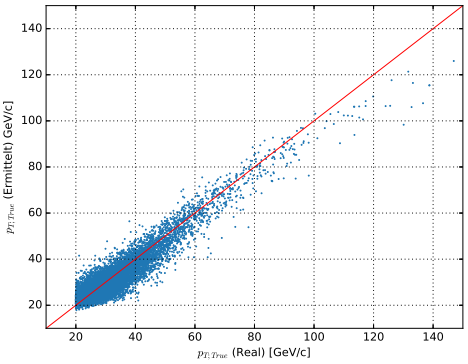
(b) $FF_{A144}(8;1;2;2;128;2)$



(c) $FF_{A144}(8;1;2;2;128;2)$



(d) $FF_{A2}(32;1;1;1;128;1)$



(e) $FF_{A2}(32;1;1;1;128;1)$

Abbildung 6: Logarithmische Lernkurven und Streudiagramme der FF-Modelle auf Rang 2 und 144 nach 100 Epochen Training.

4.3.4 Zweite Rastersuche

Da der Vorsprung des Modells auf Rang 1 ($FF_{A1}(32;1;2;1;128;1)$) bzgl. seiner Performance nicht signifikant höher ist als das Modell auf Rang 2 ($FF_{A2}(32;1;1;1;128;1)$) und nach Tab. 6 die Erhöhung von l_η, l_ϕ oder l_{p_T} keine Verbesserung des RMSE hervorruft, wird u.a. das Modell $FF_{A2}(32;1;1;1;128;1)$ (Variante 1) als Grundlage für weitere Rastersuchen verwendet.
Des Weiteren soll genauer untersucht werden, ob die Tendenz, dass 2 Schichten in den Netzen keine Verbesserung herbeiführen, mit der Beschränkung auf gleiche Neuronenanzahlen je Schicht zusammenhängt. Dafür wird eine Modellstruktur (Variante 2) betrachtet, deren Neuronenanzahl in der je folgenden Schicht meist verschieden und z.T. geringer ist. Diese Modellierung liegt auch der Intuition zu Grunde, dass für die Ermittlung eines einzelnen Outputs auf Basis von dreifach 140 Inputs eine Struktur von Vorteil sein könnte, die durch stufenweise Zusammenfassung funktioniert.

4.3.4.1 Betrachteter Hyperparameterraum Die von beiden Modellstrukturen einfachere Variante 1 sowie Variante 2 weisen in der zweiten Rastersuche die in Tab. 7 gezeigten Parameterräume auf. Die Anzahl an Schichten ist als $l = l_F = l_\eta = l_\phi = l_{p_T}$ festgesetzt.

Hyperparameter	**Wertebereich**	
	Variante 1	Variante 2
n_1	$64; 128; 256; 512$	$64; 128; 256$
n_2		$32; 64; 128$
l	1	2
n_{F_1}	$64; 128; 256$	$32; 64; 128$
n_{F_2}		$16; 32; 64$

Tabelle 7: Überblick über alle in der zweiten Rastersuche variierten Hyperparameter für die beiden Varianten des Feedforward-Netzes.

Hierbei wurde explizit darauf geachtet, dass das finale Netz meist weniger Neuronen hat als die Anzahl aller in dieses Netz weitergeleiteten Outputs, d.h. $n_F < 3n$ gilt.
Für Variante 1 soll gezeigt werden, ob sich die in der ersten Rastersuche erkannten Tendenzen (d.h. die Verbesserung der Performance für $n_F < 3n$ und v.a. für die Erhöhung von n) fortsetzen.

4.3.4.2 Ergebnisse Die Ergebnisse der zweiten Rastersuche sind für beide Modellvarianten in Tab. 8 einzusehen. Unter den nach Rang besten 10 Modellen befinden sich 6 der 12 Modelle der Variante 1. Da es sich für diese Modellvariante jedoch schon um eine Rastersuche auf Basis der erkannten Tendenzen handelt, kann nicht darauf geschlossen werden, dass diese Modell*struktur* i. Allg. bessere Ergebnisse liefert, für die betrachteten Parameterräume aber schon.

Im Vergleich zur besten Performance des besten ausgewählten Modells der ersten Rastersuche ($FF_{A2}(32; 1; 1; 1; 128; 1)$) mit $3{,}55 \pm 0{,}12$ zeigt sich hier eine signifikante Verbesserung sogar für alle Modelle der zweiten Rastersuche bis über Rang 20. Demnach haben die Untersuchungen der neuen Parameterräume für beide betrachteten Modellvarianten tatsächlich eine Verbesserung der Performance zur Folge.

Rang	n_1	n_2	n_{F_1}	n_{F_2}	l	**RMSE Val.**	***RMSE Train.***
1	512		256		1	$3{,}27 \pm 0{,}07$	*3,21 ± 0,06*
2	256		256		1	$3{,}30 \pm 0{,}07$	*3,28 ± 0,07*
3	512		128		1	$3{,}34 \pm 0{,}07$	*3,30 ± 0,07*
4	128		128		1	$3{,}35 \pm 0{,}07$	*3,34 ± 0,07*
5	256	128	128	64	2	$3{,}35 \pm 0{,}07$	*3,33 ± 0,07*
6	128	128	64	32	2	$3{,}36 \pm 0{,}07$	*3,36 ± 0,07*
7	128		256		1	$3{,}36 \pm 0{,}07$	*3,35 ± 0,07*
8	256		128		1	$3{,}36 \pm 0{,}07$	*3,35 ± 0,07*
9	256	128	64	32	2	$3{,}37 \pm 0{,}07$	*3,36 ± 0,07*
10	256	128	128	16	2	$3{,}37 \pm 0{,}07$	*3,36 ± 0,07*
11	64	128	128	16	2	$3{,}38 \pm 0{,}07$	*3,38 ± 0,07*
12	128	128	64	16	2	$3{,}40 \pm 0{,}07$	*3,41 ± 0,07*
13	256	64	32	32	2	$3{,}40 \pm 0{,}07$	*3,39 ± 0,07*
14	64	128	128	32	2	$3{,}40 \pm 0{,}07$	*3,40 ± 0,07*
15	256	128	128	32	2	$3{,}40 \pm 0{,}07$	*3,38 ± 0,07*
16	128	128	128	16	2	$3{,}41 \pm 0{,}07$	*3,41 ± 0,07*
17	256	64	64	32	2	$3{,}41 \pm 0{,}07$	*3,40 ± 0,07*
18	64		128		1	$3{,}41 \pm 0{,}07$	*3,41 ± 0,07*
19	256		64		1	$3{,}42 \pm 0{,}07$	*3,40 ± 0,07*
20	128	128	128	32	2	$3{,}43 \pm 0{,}07$	*3,42 ± 0,07*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
89	128	64	32	64	2	$5{,}78 \pm 0{,}12$	*5,73 ± 0,11*
90	256	64	32	64	2	$6{,}04 \pm 0{,}12$	*5,99 ± 0,12*
91	64	64	32	64	2	$6{,}15 \pm 0{,}12$	*6,11 ± 0,12*
92	64	128	32	64	2	$6{,}17 \pm 0{,}12$	*6,12 ± 0,12*
93	256	128	32	64	2	$6{,}27 \pm 0{,}13$	*6,21 ± 0,12*

Tabelle 8: Ergebnisse der Modellperformance für beide Varianten der Modellstruktur für die zweite Rastersuche (B). Das mittlere RMSE (Val.) liegt bei $4{,}12 \pm 0{,}17$. Dargestellt sind die 20 besten und 5 schlechtesten Modelle nach RMSE (Val.).

Die Modellvariante 1 weist mit $n_{F_1} \in \{128, 256\}$ bis einschließlich Rang 18 stets bessere Ergebnisse auf als mit $n_{F_1} = 64$.

Zu den Ergebnissen von Modellvariante 2 ist anzumerken, dass die Verringerung der Neuronenanzahl von der ersten zur zweiten Schicht für die Zweignetze und das finale Netz ein unterschiedliches Verhalten zeigt. So liegt der RMSE-Durchschnitt für Modelle mit $n_{F_2} < n_{F_1}$ bei $3{,}89 \pm 0{,}15$ und mit $n_{F_2} \geq n_{F_1}$ bei $4{,}88 \pm 0{,}35$, zeigt also eine deutliche Verbesserung für die erstgenannte Bedingung. Anders hingegen verhalten sich n_1 und n_2 im untersuchten Parameterraum, denn hier ist der Durchschnitt für $n_2 < n_1$ mit $4{,}28 \pm 0{,}22$ im Rahmen der Unsicherheit $4{,}10 \pm 0{,}35$ für $n_2 \geq n_1$ identisch.

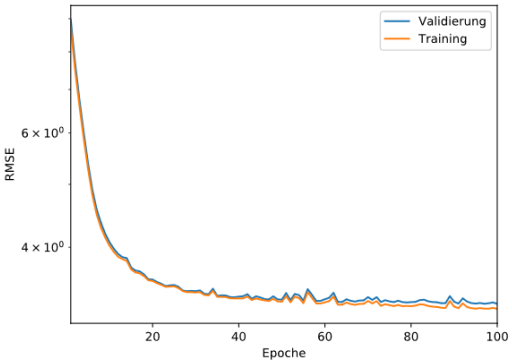
In allen Modellen ist anhand des RMSE für die Trainings- und Validierungsdatensätze nach 100 Epochen noch kein ausgeprägtes Overfitting zu erkennen, jedoch bedarf es für eine genaue Untersuchung der Betrachtung der Entwicklung des RMSE für die Trainings- und Validierungsdatensätze. Es ist in Abb. 7(a) zu erkennen, dass die beiden Kurven für beide Datensätze immer weiter voneinander entfernt sind mit steigender Epoche (sehr gut erkennbar etwa ab Epoche 50). Da die Trainingsperformance hierbei immer besser wird als die Validierungsperformance, ist

davon auszugehen, dass leichtes Overfitting vorliegt.

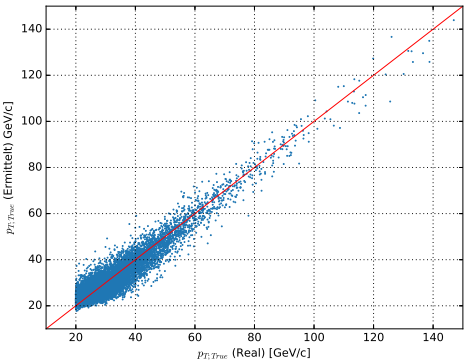
Ein Vergleich der Lernkurven und der Streudiagramme für das vormals ausgewählte Modell $FF_{A2}(32;1;1;1;128;1)$ mit dem neuen besten Modell $FF_{B1_1}(512;256;1)$ zeigt in Abb. 7 die deutlichen Verbesserungen durch die Änderungen. So ist in Abb. 7(f) die Berechnung von Transversalimpulsen vor allem ab 100 $\frac{\text{GeV}}{\text{c}}$ stets mit zu geringen Werten verbunden, wohingegen in Abb. 7(b) eine derart deutliche Tendenz nicht vorliegt.

Außerdem zeigen die beiden Lernkurven in Abb. 7(a) und 7(e) ebenso signifikante Unterschiede im Lernverhalten beider Modelle. Während in FF_{A2} der Lernvorgang in den ersten 20 Epochen nur eine langsame Verbesserung des RMSE herbeiführt, ist für FF_{B1_1} eine anfänglich schnell abfallende exponentielle Lernkurve zu erkennen.
Des Weiteren ist der Lernvorgang bei Epoche 100 in etwa konstant, womit die Anzahl an untersuchten Epochen akzeptabel erscheint[20].

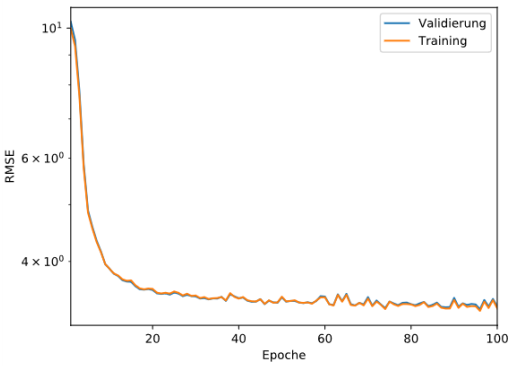
[20]Dies bedeutet nicht, dass sich die Modellperformance ab einer kritischen Epoche nicht noch verbessern könnte. Tatsächlich wurden im Verlauf der Untersuchungen auch Modelle gefunden, die nach längerer näherungsweiser Konstanz bzgl. des RMSE ab einer bestimmten Epoche ihr Lernverhalten bedeutend verbessert haben.



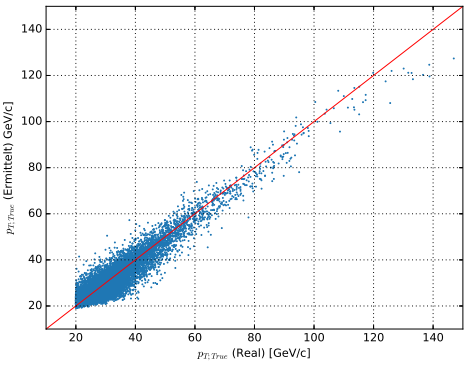
(a) $FF_{B1_1}(512; 1; 1; 1; 256; 1)$



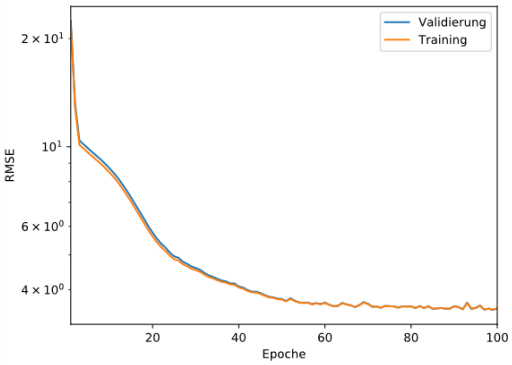
(b) $FF_{B1_1}(512; 1; 1; 1; 256; 1)$



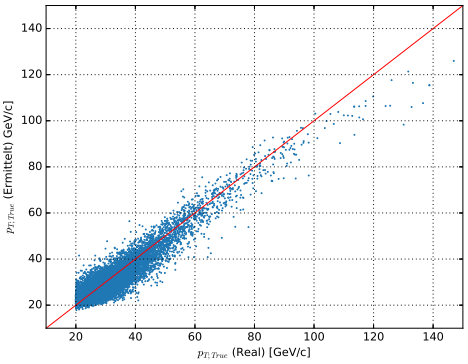
(c) $FF_{B5_2}(256; 128; 128; 64; 2)$



(d) $FF_{B5_2}(256; 128; 128; 64; 2)$

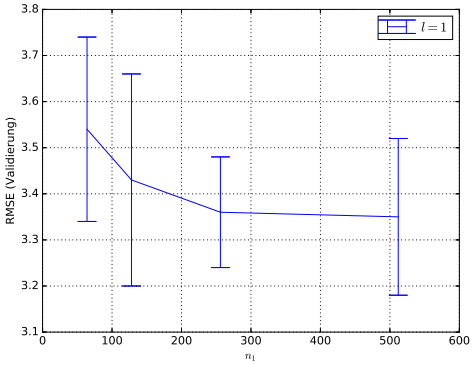


(e) $FF_{A2}(32; 1; 1; 1; 128; 1)$

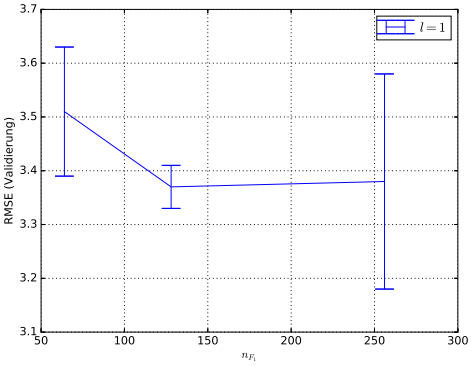


(f) $FF_{A2}(32; 1; 1; 1; 128; 1)$

Abbildung 7: Logarithmische Lernkurven und Streudiagramme der Feedforward-Modelle FF_{B1_1} (oben), FF_{B5_2} (mittig) und FF_{A2} (unten) nach 100 Epochen Training.



(a) Ergebnisse für Variation von n_1.



(b) Ergebnisse für Variation von n_{F_1}.

Abbildung 8: Mittlere Modellperformance für Modellvariante 1 in Abhängigkeit von n_1 und n_{F_1}.

Aufgrund der geringen Anzahl der für Modellvariante 1 trainierten Modelle erweisen sich die in Abb. 8 dargestellten zusammengefassten Performances als unzureichend aussagekräftig. Da sich die Erhöhung der Neuronenanzahlen in den Zweignetzen seit der ersten Rastersuche positiv auf die Performance ausgewirkt hat, wird das erstplatzierte Modell FF_{B1_1} als Grundlage für weitere Optimierungen genutzt.

4.3.5 Detailsuche nach dem besten Modell

Das zuletzt ausgewählte Modell FF_{B1_1} wird nun in einem kleineren Parameterraum für eine Detailsuche variiert.

4.3.5.1 Betrachtete Hyperparameter Um die benötigte Rechenzeit einzuschränken, wird dieses Mal zunächst n_F konstant gehalten. Da nun erstmals Overfitting aufgetreten ist und n erhöht wird, wird die Dropout-Rate zunächst auch nur in den Zweignetzen variiert. Damit kann erstmals der Einfluss der Dropout-Rate auf die Modellperformance und das Overfitting erschlossen werden.

Hyperparameter	**Symbol**	**Untersuchte Werte**
Anzahl Neuronen (Zweignetze)	n	$1024; 2048; 4096$
Anzahl Neuronen (finales Netz)	n_F	256
Dropout-Rate (Zweignetze)	r_{d_1}	$0{,}2; 0{,}3; 0{,}5$
Dropout-Rate (finales Netz)	r_{d_2}	0,2

Tabelle 9: Überblick über alle in der dritten Rastersuche variierten Hyperparameter für das Feedforward-Netz auf Grundlage von FF_{B1_1}.

Zur Überprüfung, ob sich die tendenziell bessere Performance mit höherem n fortsetzt, wurden (vgl. Tab. 9) nun noch größere Werte von n in größeren Abständen gewählt. Außerdem wurden für die Dropout-Raten Werte bis 50% betrachtet, weil diese Rate nach [6] eine sinnvolle Wahl sein soll.

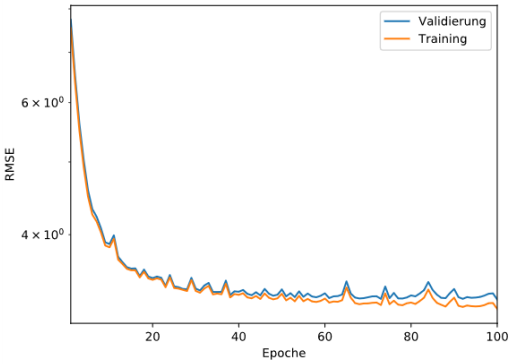
4.3.5.2 Ergebnisse In Tab. 10 zeigt sich für alle Modelle keine im Rahmen der Unsicherheit signifikante Verbesserung der RMSE-Validierungsperformance untereinander sowie im Vergleich mit FF_{B1_1} $(3{,}27 \pm 0{,}07)$. Sofort auffällig ist jedoch, dass abgesehen von Modell FF_{C4} trotz einer auf 0,5 erhöhten Dropout-Rate auch die Modelle FF_{C1} und FF_{C9} eine relativ hohe Abweichung zwischen dem RMSE auf Trainings- und Validierungsdatensätzen zeigen, was auf Overfitting

hindeutet. Das bedeutet, dass eine weitere Anpassung der Dropout-Rate z.B. im finalen Netz erforderlich zu sein scheint.

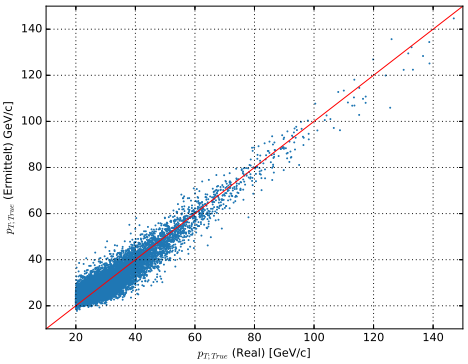
Rang	n	r_{d_1}	n_F	r_{d_2}	**RMSE (Val.)**	***RMSE (Train.)***
1	2048	0,5	256	0,2	$3{,}28 \pm 0{,}07$	*3,19 ± 0,06*
2	1024	0,2	256	0,2	$3{,}28 \pm 0{,}07$	*3,17 ± 0,06*
3	2048	0,3	256	0,2	$3{,}28 \pm 0{,}07$	*3,14 ± 0,06*
4	1024	0,5	256	0,2	$3{,}30 \pm 0{,}07$	*3,26 ± 0,07*
5	2048	0,2	256	0,2	$3{,}32 \pm 0{,}07$	*3,17 ± 0,06*
6	4096	0,2	256	0,2	$3{,}32 \pm 0{,}07$	*3,13 ± 0,06*
7	4096	0,3	256	0,2	$3{,}33 \pm 0{,}07$	*3,12 ± 0,06*
8	1024	0,3	256	0,2	$3{,}34 \pm 0{,}07$	*3,25 ± 0,07*
9	4096	0,5	256	0,2	$3{,}34 \pm 0{,}07$	*3,19 ± 0,06*

Tabelle 10: Alle Ergebnisse der Modellperformances für die dritte Rastersuche (C).

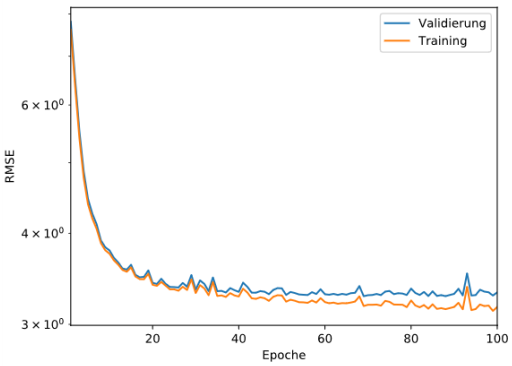
Das Problem des Overfittings zeigt sich auch in der RMSE-Kurve für die Validierungs- und Trainingsdatensätze mit steigender Epochenanzahl zum Training in Abb. 9(a) und 9(c). Ein Vergleich zum bis jetzt besten Modell FF_{B1_1} zeigt in Abb. 9(e) das mit der erhöhten Modellkomplexität bei gleichbleibender Dropout-Rate auftretende Overfitting deutlich.
In den Streudiagrammen sind in Abb. 9(f) und 9(b) keine unmittelbar deutlichen Unterschiede bzgl. der Vorhersagequalität erkennbar, was sich in den ähnlichen RMSE-Werten beider Modelle ebenso ausdrückt.



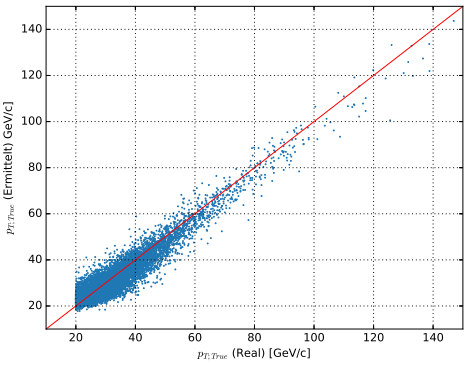
(a) $FF_{C1}(2048; 0{,}5; 256; 0{,}2)$



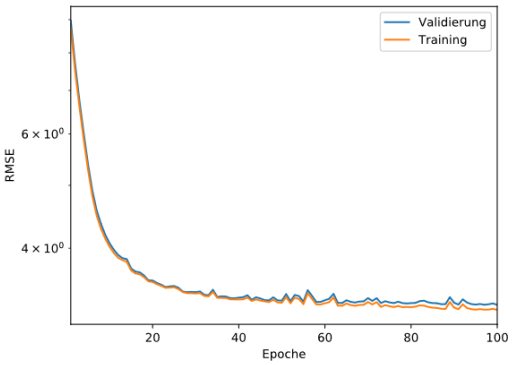
(b) $FF_{C1}(2048; 0{,}5; 256; 0{,}2)$



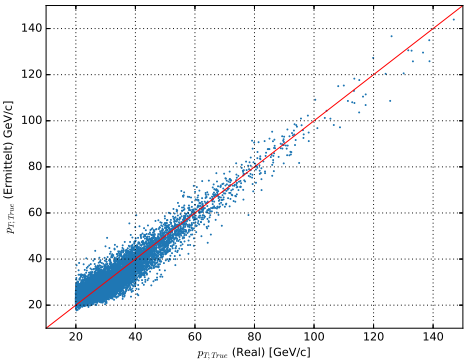
(c) $FF_{C5}(2048; 0{,}2; 256; 0{,}2)$



(d) $FF_{C5}(2048; 0{,}2; 256; 0{,}2)$



(e) $FF_{B1_1}(512; 1; 1; 1; 256; 1)$



(f) $FF_{B1_1}(512; 1; 1; 1; 256; 1)$

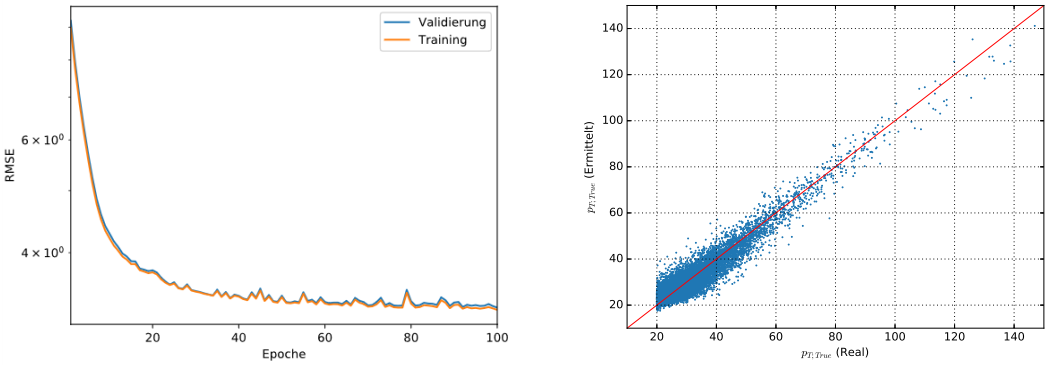
Abbildung 9: Logarithmische Lernkurven und Streudiagramme der Feedforward-Modelle FF_{C1} (oben), FF_{C5} (mittig) und FF_{B1_1} (unten) nach 100 Epochen Training.

Nun wurde versucht, über eine Erhöhung von r_{d_2} auf 0,5 für FF_{C1} das Overfitting zu reduzieren und über Variation von n_F mit $n_F \in \{256{,}512{,}1024\}$ eine bessere Performance zu erzielen. Die dabei aufgetretenen Ergebnisse (vgl. Tab 11) zeigen zwar eine Verbesserung bzgl. des Overfittings, sind jedoch nicht besser als die Modellperformance von FF_{B1_1} mit $3{,}27 \pm 0{,}07$.

Rang	n	r_{d_1}	n_F	r_{d_2}	RMSE (Val.)	*RMSE (Train.)*
1	2048	0,5	512	0,5	$3{,}30 \pm 0{,}07$	*3,24 ± 0,06*
2	2048	0,5	256	0,5	$3{,}37 \pm 0{,}07$	*3,33 ± 0,07*
3	2048	0,5	1024	0,5	$3{,}38 \pm 0{,}07$	*3,27 ± 0,07*

Tabelle 11: Ergebnisse der detaillierten Suche (D) auf Basis von FF_{C1} durch Variation von n_F.

Für eine Verringerung des Overfittings bei FF_{B1_1} wurde zudem geprüft, inwiefern sich eine Erhöhung der Dropout-Raten auf $r_d = r_{d_1} = r_{d_2} = 0{,}3$ unter Beibehaltung der sonstigen Modellparameter hierbei auswirkt.
Die erhaltene Performance für das als F_{E1} bezeichnete Modell beträgt $3{,}28 \pm 0{,}07$ (Validierung) bzw. $3{,}26 \pm 0{,}07$ (Training). Das bedeutet, dass das Overfitting bei im Rahmen der Unsicherheit gleicher Performance reduziert werden kann. Ein Vergleich zwischen Abb. 9(e) und Abb. 10(a) zeigt, dass das Overfitting deutlich reduziert werden konnte, während die Streudiagramme in Abb. 9(f) und 10(b) keine besonderen Unterschiede aufweisen.
Im Rahmen der Unsicherheit sind die RMSE-Werte von FF_{B1_1} und FF_{E1} identisch, da jedoch das Overfitting von FF_{E1} bei keinen signifikanten Unterschieden der Vorhersagequalität geringer ausgeprägt ist, wird dieses Modell als besser angesehen.



(a) FF_{E1} (b) FF_{E1}

Abbildung 10: Logarithmische Lernkurven und Streudiagramme der Feedforward-Modelle FF_{E1} nach 100 Epochen Training.

4.3.6 Wahl des besten Feedforward-Modells

Da sich die zuletzt besten Modelle $FF_{C1}(2048; 0{,}5; 256; 0{,}2)$ mit einem RMSE von $3{,}28 \pm 0{,}07$ und $FF_{E1}(512; 0{,}3; 256; 0{,}3)$ mit einem RMSE von $3{,}28 \pm 0{,}07$ auf den Validierungsdatensätzen nicht voneinander unterscheiden bzgl. ihrer Modellperformance, bedarf es der Anwendung weiterer Kriterien für die Auswahl des nun besten Feedforward-Modells im Rahmen der betrachteten Parameterräume.
Auch die Streudiagramme weisen nach Kap. 4.3.5.2 keine deutlichen Unterschiede auf[21].

Das Overfitting ist in FF_{E1} mit der in Abb. 10(a) zu erkennenden Nähe beider RMSE-Kurven als gering einzustufen, wohingegen dieses Problem bei FF_{C1} verstärkt (vgl. Abb. 9) auftritt.

[21] Natürlich dient das RMSE der Einschätzung der Vorhersagequalität über den gesamten untersuchten Bereich, allerdings könnte es aufgrund der Datenverteilung dazu kommen, dass in bestimmten Impulsbereiche nur eine sehr schlechte Übereinstimmung vorliegt, was bedeuten würde, dass das Netz den Zusammenhang nicht ausreichend erkannt hat.

Ein Vergleich der beiden RMSE-Diagramme in Abb. 10(a) und 9(a) zeigt auch, dass ein Abbruch bei einer früheren Epoche wegen des abseits geringer Fluktuationen (nahezu) konstanten Verlaufs der Validierungs-RMSE-Kurve dieses Problem lösen könnte[22].

Insgesamt ergibt sich demnach als finales Feedforward-Modell FF_{E1} mit $n = 512$, $l = l_\eta = l_\phi = l_{p_T} = 1$, $r_{d_1} = 0{,}2$, $n_F = 256$, $l_F = 1$ und $r_{d_2} = 0{,}3$ und einer Validierungs-Performance von $3{,}28 \pm 0{,}07$ sowie einer Trainings-Performance von $3{,}26 \pm 0{,}07$, weil die erhöhte Modellkomplexität bei FF_{C1} keine bessere Vorhersagequalität bietet und das Overfitting mit FF_{E1} (nahezu) eliminiert werden konnte.

4.4 Ergebnisse zum Convolutional Neural Network

4.4.1 Allgemeine Modellstruktur

Die Basisstruktur des genutzten eindimensionalen CNN-Modells ist in Tab. 12 zu sehen. Bei der Modellierung werden mehrere Lagen von Conv1D-, MaxPooling1D- und Dropout-Schichten verwendet. Am Ende wird nach einer Dimensionsreduktion über Flatten ein normales Perzeptron verwendet. Auch hier wiederholen sich die Schichten.

Als Input dienen die auf drei Kanäle aufgeteilten Konstituentendaten $\eta_{Konst.}$, $\phi_{Konst.}$ und $p_{T_{Konst.}}$ wieder mit bis zu 140 Konstituenten (Zero-Padding) absteigend nach dem Transversalimpuls sortiert.

[22] Oft geht eine solche Veränderung nach eigener Erfahrung v.a. mit schlechteren Ergebnissen für Inputbereiche mit einer geringeren Anzahl an Datensätzen einher, da die Details mit einer geringen Datenbasis meist erst später erlernt werden.

Bezeichnung	**Gesetzte Parameter (Keras)**
Conv1D	• filters = f_i • kernel_size = 2 • kernel_initializer = ”he_uniform” • activation = ”tanh” • strides = 1 • padding = ”same”
MaxPooling1D	• pool_size = p_i
Dropout	• rate = r_{d_1}
Flatten	ohne Parameter
Dense	• units = n_{F_j} • kernel_initializer = ”he_uniform” • activation = ”relu” • kernel_regularizer= $l2(0{,}001)$
Dropout	• rate = r_{d_2}
Dense	• units = 1 • activation = ”linear”

Tabelle 12: Darstellung der genutzten Schichten mit zugehörigen Parametern für die untersuchten CNN-Modelle. Für die Konvolutionsschichten wurde der Index i, für die finalen Schichten der Index j verwendet.

4.4.2 Bestimmung der Modellfluktuation

Da sich die Struktur des CNN von der Struktur eines normalen Perzeptrons aus den FF-Modellen deutlich unterscheidet, wird für das CNN eine eigene Ermittlung der Modellfluktuation durchgeführt.
Als Testmodell wurde ein eindimensionales CNN-Modell nach Tab. 12 mit gleichen Filtergrößen $f = f_1 = f_2 = f_3 = 32$, $n_F = 128$, $l_F = 2$ und $r_{d_{1,2}} = 0{,}2$ verwendet. Dabei (vgl. 13) zeigt sich für 10 Durchläufe ein durchschnittliches RMSE auf den Validierungsdatensätzen von etwa $3{,}563 \pm 0{,}093$ (99,8% Signifikanz) bzw. 2,6%. Die hierbei auftretende Unsicherheit wird damit höher als bei den FF-Modellen eingeschätzt.

Durchlauf	RMSE (Val.)	*RMSE (Train.)*
1	3,47	*3,47*
2	3,55	*3,54*
3	3,59	*3,58*
4	3,63	*3,61*
5	3,47	*3,47*
6	3,54	*3,52*
7	3,63	*3,61*
8	3,55	*3,55*
9	3,69	*3,66*
10	3,51	*3,50*

Tabelle 13: Darstellung der Modellfluktuationen anhand des gewählten CNN-Beispielmodells mit $f = f_1 = f_2 = f_3 = 32$, $n_F = 128$, $l_F = 2$ und $r_{d_{1,2}} = 0{,}2$. Wie auch beim FF-Modell sind deutliche Unterschiede zwischen den Modellperformances erkennbar, was die Modellauswahl signifikant beeinflussen könnte.

4.4.3 Erste Rastersuche

In der ersten Rastersuche soll ein erster Überblick über das Verhalten des Netzes bei Variation der Filtergrößen und des finalen Netzes gewonnen werden.

4.4.3.1 Betrachtete Hyperparameter Ein CNN weist ähnlich wie das Feedforward Netzwerk viele Hyperparameter auf, deren Auswahl für Rastersuchen in Anbetracht der Rechenzeit wohl überlegt werden muss. Dafür wurden auch hier *zunächst* mehrere Hyperparameter fixiert (vgl. Tab. 1), deren Wahl auf Standard- und Erfahrungswerten basiert.

Neben diesen für die erste Rastersuche fixierten Hyperparametern sind die dabei variierten Parameter in Tab. 14 eingetragen. Die hierbei auftretenden Filtergrößen beziehen sich auf die jeweilige Konvolutionsschicht. Die Anzahl an Konvolutionsschichten wurde hierbei zunächst als 3 festgelegt, da diese Anzahl zum Einen eine akzeptable Rechenzeit auf dem Arbeitssystem erfordert und zum Anderen die Möglichkeit bietet, den Einfluss der Vergrößerung oder Verringerung aufeinanderfolgender Konvolutionsschichten auf die Modellperformance zu untersuchen.

Die Wahl der MaxPooling1D-Schichten ergab sich aus dem Grundgedanken, dass zunächst über eine Pooling-Schicht p_1 die größten Rauschbeiträge dezimiert werden und die darauffolgenden Schichten p_2 und p_3 ein größeres Gesamtbild extrahierter Informationen verarbeiten können.

Hyperparameter	Symbol	Untersuchte Werte
Filtergrößen	f_1, f_2, f_3	$8; 16; 32$
Anzahl Neuronen (finales Netz)	n_F	$64; 128$
Anzahl Schichten (finales Netz)	l_F	$1; 2$
Dropout-Rate (CNN und finales Netz)	$r_{d_{1,2}}$	0,2
Pooling	$[p_1, p_2, p_3]$	[2,0,0]

Tabelle 14: Überblick über alle in der ersten Rastersuche für das CNN-Modell variierten Hyperparameter für das Feedforward-Netz.

Um das vorerst beste Modell zu finden wurden alle Permutationen aus f_1, f_2, f_3, n_F und l_F bzgl. ihrer Performance auf Basis des RMSE mit dem Validierungsdatensatz untersucht.

4.4.3.2 Ergebnisse Die Ergebnisse für die erste Rastersuche zeigen nach Tab. 15, dass nur die zwei besten erhaltenen Modelle zur Auswahl stehen für weitere Optimierungen der Modellparameter.

Rang	f_1	f_2	f_3	n_F	l_F	**RMSE Val.**	***RMSE Train.***
1	32	32	32	128	2	$3{,}47 \pm 0{,}09$	$3{,}47 \pm 0{,}09$
2	32	8	32	128	2	$3{,}55 \pm 0{,}09$	$3{,}53 \pm 0{,}09$
3	32	16	32	128	2	$3{,}57 \pm 0{,}09$	$3{,}57 \pm 0{,}09$
4	32	32	8	128	2	$3{,}61 \pm 0{,}09$	$3{,}59 \pm 0{,}09$
5	8	32	32	64	2	$3{,}61 \pm 0{,}09$	$3{,}60 \pm 0{,}09$
6	8	32	32	128	2	$3{,}62 \pm 0{,}09$	$3{,}60 \pm 0{,}09$
7	16	32	32	64	2	$3{,}67 \pm 0{,}10$	$3{,}65 \pm 0{,}10$
8	32	32	32	64	2	$3{,}67 \pm 0{,}10$	$3{,}65 \pm 0{,}10$
9	32	32	16	128	2	$3{,}70 \pm 0{,}10$	$3{,}69 \pm 0{,}10$
10	16	32	32	128	2	$3{,}72 \pm 0{,}10$	$3{,}70 \pm 0{,}10$
11	32	32	16	64	2	$3{,}73 \pm 0{,}10$	$3{,}71 \pm 0{,}10$
12	16	32	16	128	2	$3{,}77 \pm 0{,}10$	$3{,}75 \pm 0{,}10$
13	32	16	8	128	2	$3{,}77 \pm 0{,}10$	$3{,}76 \pm 0{,}10$
14	32	16	16	64	2	$3{,}78 \pm 0{,}10$	$3{,}75 \pm 0{,}10$
15	32	16	32	64	2	$3{,}78 \pm 0{,}10$	$3{,}75 \pm 0{,}10$
16	16	16	32	128	2	$3{,}86 \pm 0{,}10$	$3{,}84 \pm 0{,}10$
17	8	16	32	128	2	$3{,}86 \pm 0{,}10$	$3{,}82 \pm 0{,}10$
18	16	8	32	64	2	$3{,}86 \pm 0{,}10$	$3{,}83 \pm 0{,}10$
19	16	16	16	64	2	$3{,}86 \pm 0{,}10$	$3{,}84 \pm 0{,}10$
20	16	8	16	128	2	$3{,}89 \pm 0{,}10$	$3{,}86 \pm 0{,}10$
21	32	32	8	64	2	$3{,}90 \pm 0{,}10$	$3{,}86 \pm 0{,}10$
22	32	8	16	128	2	$3{,}91 \pm 0{,}10$	$3{,}88 \pm 0{,}10$
23	32	32	32	128	1	$3{,}91 \pm 0{,}10$	$3{,}89 \pm 0{,}10$
24	8	16	32	128	1	$3{,}91 \pm 0{,}10$	$3{,}90 \pm 0{,}10$
25	8	8	32	128	2	$3{,}92 \pm 0{,}10$	$3{,}91 \pm 0{,}10$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
99	8	32	8	64	1	$4{,}83 \pm 0{,}13$	$4{,}76 \pm 0{,}12$
100	8	8	8	64	1	$4{,}85 \pm 0{,}13$	$4{,}78 \pm 0{,}13$
101	8	16	8	64	1	$4{,}96 \pm 0{,}13$	$4{,}89 \pm 0{,}13$
102	16	8	8	128	1	$4{,}98 \pm 0{,}13$	$4{,}89 \pm 0{,}13$
103	16	16	8	64	1	$4{,}99 \pm 0{,}13$	$4{,}93 \pm 0{,}13$
104	16	8	8	64	1	$4{,}99 \pm 0{,}13$	$4{,}91 \pm 0{,}13$
105	8	16	32	64	2	$5{,}00 \pm 0{,}13$	$4{,}97 \pm 0{,}13$
106	8	8	8	128	1	$5{,}02 \pm 0{,}13$	$4{,}94 \pm 0{,}13$
107	8	16	16	64	1	$5{,}18 \pm 0{,}14$	$5{,}13 \pm 0{,}13$
108	8	8	32	64	2	$5{,}18 \pm 0{,}14$	$5{,}15 \pm 0{,}13$

Tabelle 15: Erhaltene Modellperformances für das eindimensionale CNN-Modell in der ersten Rastersuche (A). Gut zu erkennen ist, dass die besten 22 Modelle 2 Schichten im finalen Netz aufweisen.

Um neben der Entwicklung des RMSE mit bestimmten variierten Parametern auch mögliche Abhängigkeiten von Bedingungen an die Modellstruktur zu untersuchen, wurden die in Tab. 16

gezeigten Bedingungen auf alle Modelle angewandt[23]. Im Rahmen der auftretenden Unsicherheiten sind dabei alle auftretenden RMSE-Mittelwerte identisch. Auch ein Blick auf die besten 10 Ränge in Tab. 15 zeigt, dass die Größenrelation zwischen den Filtern z.B. beim Vergleich von CNN_{A2} ($3{,}55 \pm 0{,}09$) mit CNN_{A4} ($3{,}61 \pm 0{,}09$) und CNN_{A6} ($3{,}62 \pm 0{,}09$) keinen signifikanten Unterschied bzgl. der Modellperformance ergibt.

Bedingung	**RMSE (Val.) Wahr**	**RMSE (Val.) Falsch**
$f_1 > f_2 > f_3$	$4{,}19 \pm 0{,}59$	$4{,}25 \pm 0{,}08$
$f_1 < f_2 < f_3$	$4{,}23 \pm 0{,}73$	$4{,}25 \pm 0{,}08$
$f_1 = f_2 = f_3$	$4{,}18 \pm 0{,}28$	$4{,}26 \pm 0{,}08$

Tabelle 16: Ergebnisse für das RMSE auf den Validierungsdatensätzen unter verschiedenen Bedingungen, die für alle Ergebnisse gemittelt wurden.

Die gemittelten Modellperformances in Abhängigkeit der Filtergrößen mit $f = f_1 = f_2 = f_3$ sowie der Anzahl an Neuronen in der finalen Schicht n_F zeigen in Abb. 11 ebenso keine ausreichend signifikant fallende Tendenz, jedoch ist bis Rang 18 jedes Modell mit mindestens einer Konvolutionsschicht, die einen Filter von 32 aufweist. Nur 7 Modelle weisen bis zu diesem Rang einen Filter der Größe 8 auf.

Bei der Betrachtung der Abhängigkeit des RMSE von n_F und l_F zeigt sich eine signifikante Tendenz, wie sie in Tab 11 für die Erhöhung beider Größen zu erkennen ist. So verringert sich der RMSE von $l_F = 1$ auf $l_F = 2$ gemittelt um[24] $0{,}46 \pm 0{,}12$. Die gleiche Entwicklung ist von $n_F = 64$ zu $n_F = 128$ mit einer Verringerung um $0{,}22 \pm 0{,}14$ zu sehen. Eine Erhöhung der Schichtanzahl bei gleichzeitiger Erhöhung der Neuronenanzahl im finalen Netz verbessert die Performance sogar um $0{,}69 \pm 0{,}15$.

Eine Erhöhung der Filtergrößen zeigt nicht signifikant verbessertes RMSE (Abb. 11), legt aber nahe, dass größere Filtergrößen untersucht werden sollten.

[23] Da stets auf die Nutzung der Studentschen t-Funktion für einen 95%igen Signifikanzbereich geachtet wird und manche Bedingungen die Anzahl der zu mittelnden RMSE-Werte erheblich verringern, ist die Unsicherheit entsprechend hoch.

[24] Für berechnete Werte (hier die Differenz) wird die Gaußsche Fehlerfortpflanzung angewandt.

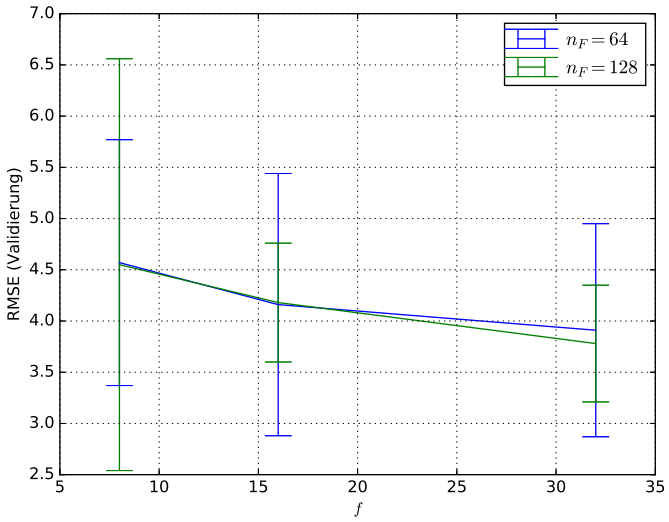


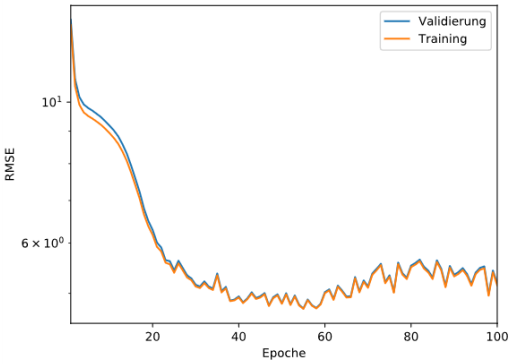
Abbildung 11: Abhängigkeit der Modellperformance von n_F und f.

RMSE (Val.	$\boldsymbol{n_F = 64}$	$\boldsymbol{n_F = 128}$	$\boldsymbol{n_F \in \{64{,}128\}}$
$l_F = 1$	$4{,}62 \pm 0{,}10$	$4{,}34 \pm 0{,}11$	$4{,}48 \pm 0{,}08$
$l_F = 2$	$4{,}10 \pm 0{,}15$	$3{,}93 \pm 0{,}11$	$4{,}02 \pm 0{,}09$
$l_F \in \{1{,}2\}$	$4{,}36 \pm 0{,}11$	$4{,}14 \pm 0{,}09$	$4{,}25 \pm 0{,}07$

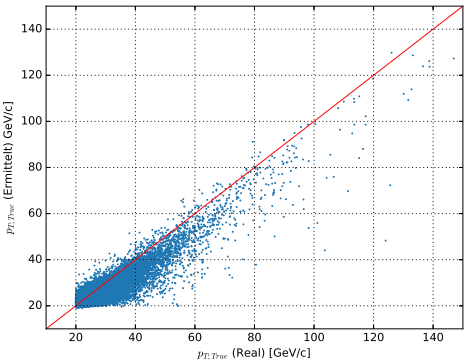
Tabelle 17: Darstellung der gemittelten RMSE auf den Validierungsdatensätzen in Abhängigkeit der auftretenden Parameterkombinationen von n_F und l_F.

Ein Vergleich zwischen den Ergebnissen von CNN_{A1} und CNN_{A108} (Abb. 12) zeigt auch hier, dass eine starke Verbesserung des RMSE (hier über 30%) zu einer wesentlich besseren Vorhersagequalität führt. Das Modell CNN_{A108} schätzt alle Werte eher als zu gering ein, wohingegen CNN_{A1} eine um die (ideale) Diagonale zu großen Anteilen gleichmäßige Verteilung aufweist. Dennoch sind in Abb. 12(d) mehrere stark vom Realwert abweichende Punkte zu erkennen. So wird z.B. ein Jet mit einem $p_{T;True}$ von etwa 125 $\frac{\text{GeV}}{\text{c}}$ bei etwas über 70 $\frac{\text{GeV}}{\text{c}}$ eingeschätzt. Diese Abweichungen zeigen, dass es weiterer Optimierungen bedarf, um ein für den vorgesehenen Bereich des $p_{T;True}$ akzeptables Ergebnis zu erhalten.

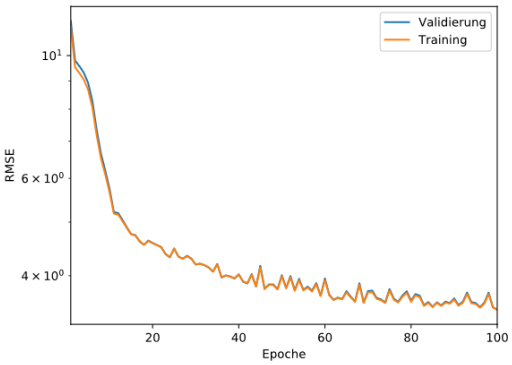
Sehr auffällig ist, dass das RMSE für CNN_{A108} nach etwa 50 Epochen anfängt zu steigen, bei CNN_{A1} aber noch ein leichtes Absinken zu erkennen ist. Möglicherweise ist die Lernrate der Modelle (evtl. nur für höhere Epochen) zu hoch angesetzt, was auch durch die gut erkennbaren Fluktuationen des RMSE in Abb. 12(a) und 12(c) naheliegt.
Trotz eine früheren Abbruchs des Lernprozesses bei CNN_{A108} wäre die Performance immer noch signifikant schlechter als bei CNN_{A1} und das bessere Modell zeigt, dass auch mit dieser Lernrate ein erwünschter (fallender) Verlauf der RMSE-Kurve gewährleistet werden kann.



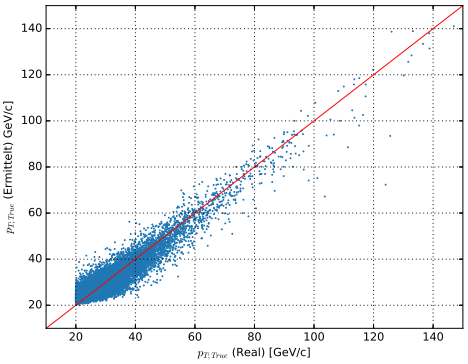
(a) $CNN_{A108}(8; 8; 32; 64; 2)$



(b) $CNN_{A108}(8; 8; 32; 64; 2)$



(c) $CNN_{A1}(32; 32; 32; 128; 2)$



(d) $CNN_{A1}(32; 32; 32; 128; 2)$

Abbildung 12: Logarithmische Lernkurven und Streudiagramme der CNN-Modelle auf Rang 1 und 108 nach 100 Epochen Training. Es ist deutlich zu sehen, dass keine der beiden Modelle Overfitting zeigen. Auffällig ist jedoch, dass ein früherer Abbruch des Lernvorgangs bei CNN_{A108} zu einer besseren RMSE-Performance geführt hätte.

4.4.4 Zweite Rastersuche

Für die zweite Rastersuche wird nun das Modell CNN_{A1} aus der ersten Rastersuche als Basis genommen. Es gilt nun, die in der ersten Rastersuche erkannten Tendenzen detaillierter zu untersuchen.
Da die Trainingszeit der folgenden CNN-Modelle erheblich größer war als für die getesteten FF-Modelle, ist die Größe des untersuchten Parameterraumes hier wesentlich geringer.

4.4.4.1 Betrachtete Hyperparameter Da es nach den bisherigen Erkenntnissen keine Verbesserungen für ungleiche Filtergrößen gibt, wird CNN_{A1} entsprechend $f = f_1 = f_2 = f_3$ gesetzt. Außerdem sollen die bisher deutlichen Verbesserungen bei Erhöhung von n_F und l_F weiterhin geprüft werden, weshalb (vgl. Tab. 18) beide Parameter mit z.T. höheren Werten angesetzt werden. Außerdem werden die Filtergrößen erhöht, um die Tendenz des RMSE für erhöhte Filtergrößen bei den besten Modellen aus der ersten Rastersuche weiter zu prüfen.
Um zu erkennen, ob eine geringere Lernrate wegen einer möglichen Verringerung der RMSE-Fluktuationen bei fortlaufender Epoche erforderlich ist, wird auch die halbierte Lernrate betrachtet.

Hyperparameter	Symbol	Untersuchte Werte
Filtergrößen	f	$64; 128$
Anzahl Neuronen (finales Netz)	n_F	$128; 256$
Anzahl Schichten (finales Netz)	l_F	$2; 3$
Dropout-Rate (CNN und finales Netz)	$r_{d_{1,2}}$	0,2
Pooling	$[p_1, p_2, p_3]$	[2,0,0]
Lernrate	r_l	$0{,}0001; 0{,}00005$

Tabelle 18: Überblick über alle in der zweiten Rastersuche für das CNN-Modell variierten Hyperparameter. Dieses Mal wird auch die Lernrate angepasst.

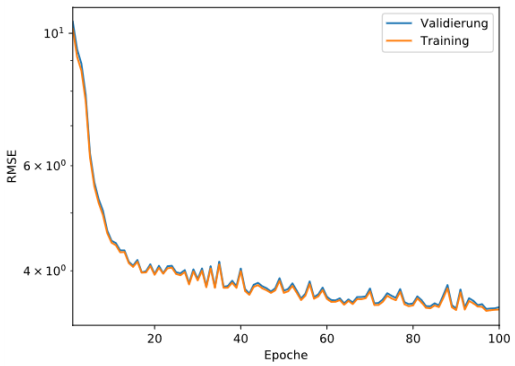
4.4.4.2 Ergebnisse Die erhaltenen Ergebnisse[25] (vgl. Tab. 19) zeigen entgegen der Erwartung, dass die Modelle mit drei l_F-Schichten im betrachteten Parameteraum meist schlechter sind als die Modelle mit zwei Schichten. So befinden sich von Rang 1 bis 8 nur zwei Modelle mit drei Schichten, wohingegen sechs Modelle mit drei Schichten von Rang 9 bis 16 platziert sind.
Auch die Verringerung der Lernrate scheint nicht erforderlich zu sein, denn die besten drei zur Auswahl stehenden Modelle weisen alle eine Lernrate von 0,0001 wie ursprünglich angesetzt auf.

Rang	f	n_F	l_F	r_l	RMSE (Val.)	*RMSE (Train.)*
1	128	256	2	0,0001	$3{,}37 \pm 0{,}09$	*3,34 ± 0,09*
2	128	128	2	0,0001	$3{,}38 \pm 0{,}09$	*3,37 ± 0,09*
3	64	256	2	0,0001	$3{,}40 \pm 0{,}09$	*3,39 ± 0,09*
4	64	128	3	0,0001	$3{,}47 \pm 0{,}09$	*3,44 ± 0,09*
5	128	256	2	0,00005	$3{,}48 \pm 0{,}09$	*3,45 ± 0,09*
6	128	128	2	0,00005	$3{,}56 \pm 0{,}09$	*3,54 ± 0,09*
7	64	256	3	0,0001	$3{,}58 \pm 0{,}09$	*3,55 ± 0,09*
8	64	256	2	0,00005	$3{,}61 \pm 0{,}09$	*3,59 ± 0,09*
9	64	128	2	0,0001	$3{,}68 \pm 0{,}10$	*3,65 ± 0,10*
10	64	128	2	0,00005	$3{,}72 \pm 0{,}10$	*3,69 ± 0,10*
11	64	256	3	0,00005	$4{,}00 \pm 0{,}10$	*3,97 ± 0,10*
12	64	128	3	0,00005	$4{,}48 \pm 0{,}12$	*4,43 ± 0,12*
13	128	128	3	0,0001	$4{,}76 \pm 0{,}12$	*4,69 ± 0,12*
14	128	256	3	0,00005	$5{,}05 \pm 0{,}13$	*4,99 ± 0,13*
15	128	128	3	0,00005	$5{,}46 \pm 0{,}14$	*5,40 ± 0,14*
16	128	256	3	0,0001	$5{,}55 \pm 0{,}15$	*5,48 ± 0,14*

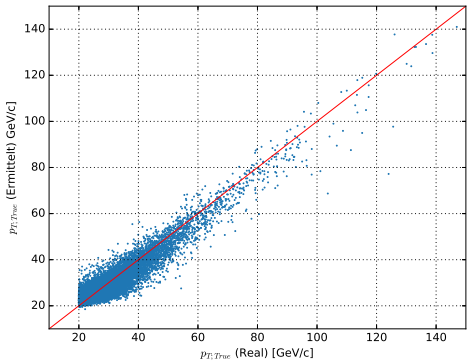
Tabelle 19: Ergebnisse der zweiten Rastersuche (B) für das CNN-Modell auf Basis von CNN_{A1}.

Ein Vergleich des zuvor gewählten Modells CNN_{A1} mit dem Modell CNN_{B1} zeigt in Abb. 13, dass keine direkt offensichtlichen Verbesserungen bzgl. der Vorhersage zu erkennen sind. Insbesondere ist das Problem starker Abweichungen für einzelne Werte des $p_{T;True}$ immer noch nicht ausreichend optimiert.

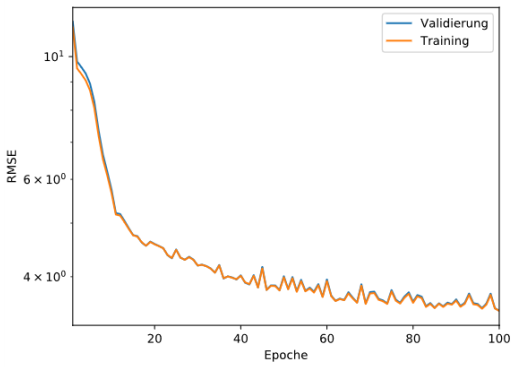
[25] Wegen der geringen Anzahl an Modellen wird hier auf Gesamtüberblicke mit Mittelwerten verzichtet.



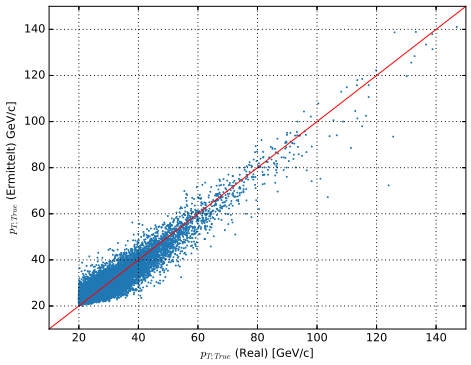
(a) CNN_{B1}



(b) CNN_{B1}



(c) CNN_{A1}



(d) CNN_{A1}

Abbildung 13: Logarithmische Lernkurven und Streudiagramme der CNN-Modelle CNN_{B1} und CNN_{A1} nach 100 Epochen Training.

Insgesamt konnte unter Beibehaltung der Lernrate und der Anzahl an Schichten im finalen Netz das RMSE mit der erfolgten Erhöhung von n_F und f signifikant von $3{,}47 \pm 0{,}09$ (CNN_{A1}) auf $3{,}37 \pm 0{,}09$ (CNN_{B1}) verringert werden. Weil dies eine Tendenz hin zu besseren RMSE-Werten mit steigendem f und n_F nahelegt, wird CNN_{B1} für die folgende Detailsuche verwendet.

4.4.5 Detailsuche

Für die Detailsuche sollen einige Parameter variiert werden, die bis jetzt nicht variiert wurden, und das beste CNN-Modell gefunden werden.

4.4.5.1 Betrachtete Hyperparameter Bis jetzt wurde die Anzahl an Konvolutionsschichten konstant gehalten. Um den Einfluss dieses Struktur-Parameters zu untersuchen, wird das Netz mit einer kleinere und größere Anzahl an Konvolutionsschichten verändert (vgl. Tab. 20).
Des Weiteren werden die MaxPooling-Schichten neben dem bisherigen Ansatz ($p_1 = 2$, alle anderen Schichten 0) um eine konstante Anzahl an Pooling-Größen ($p_1 = p_2 = ... = p_N = 2$ für N Schichten) erweitert, um auch den Einfluss dieser Größe zu prüfen.

Hyperparameter	**Symbol**	**Untersuchte Werte**
Anzahl Konvolutionsschichten	N	2,3,4
Pooling	$[p_1, p_2, ..., p_N]$	[2,0, ...,0]; [2,2, ...,2]

Tabelle 20: Überblick über alle in der dritten Rastersuche für das CNN-Modell variierten Hyperparameter. Erstmals werden die Anzahl der Konvolutionsschichten und die Pooling-Schichten verändert.

Ausgenommen aus der Rastersuche ist die in ihr enthaltene Modellstruktur von CNN_{B1}.

4.4.5.2 Ergebnisse Die in Tab. 21 gezeigten Ergebnisse legen nahe, dass eine weitere Erhöhung der Anzahl an Konvolutionsschichten in dem untersuchten Parameterraum keine Verbesserung, sondern sogar eine Verschlechterung der Performance zur Folge hat.

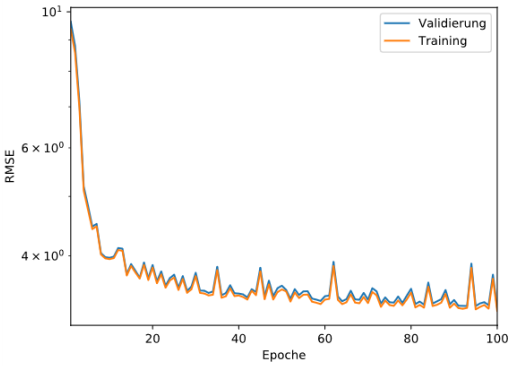
Rang	N	f	p	n_F	**RMSE (Val.)**	***RMSE (Train.)***
1	2	128	[2,2]	256	$3{,}29 \pm 0{,}09$	$3{,}25 \pm 0{,}09$
2	3	128	[2,2,2]	256	$3{,}35 \pm 0{,}09$	$3{,}33 \pm 0{,}09$
3	2	128	[2,0]	256	$3{,}52 \pm 0{,}09$	$3{,}44 \pm 0{,}09$
4	4	128	[2,0,0,0]	256	$3{,}53 \pm 0{,}09$	$3{,}52 \pm 0{,}09$
5	4	128	[2,2,2,2]	256	$3{,}63 \pm 0{,}10$	$3{,}57 \pm 0{,}09$

Tabelle 21: Ergebnisse für das RSME der CNN-Modelle in der dritten Rastersuche (C).

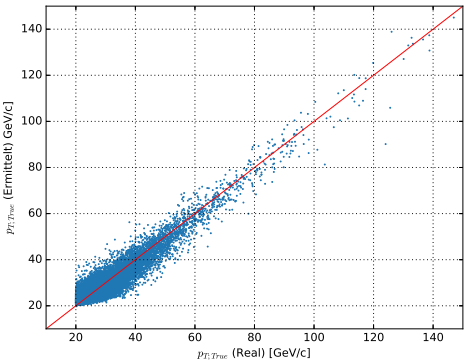
Für die Wahl der Pooling-Schichten scheint [2,2] bzw. [2,2,2] zunächst vorteilhaft zu sein, allerdings sind die für CNN_{C1} und CNN_{C2} auftretenden Validierungs-RMSE nicht signifikant verschieden von dem bisherigen Spitzenwert von CNN_{B1} mit $3{,}37 \pm 0{,}09$. Aus diesem Grund bedarf es des detaillierten Vergleichs beider Modelle über die zugehörigen Streudiagramme.

4.4.6 Wahl des besten CNN-Modells

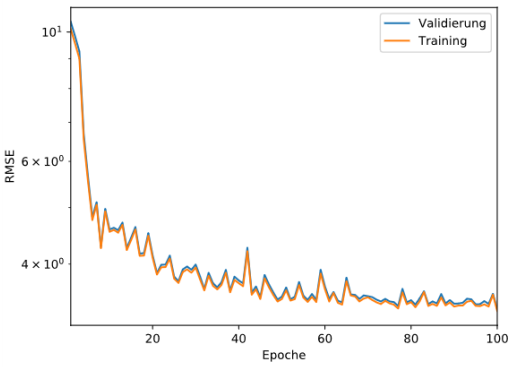
Es zeigt sich in Abb. 14, dass die zuvor stark von den realen Werten abweichenden einzelnen Datenpunkte nun näher am realen Wert liegen. Bei CNN_{C1} wird kein $p_{T;True}$, welches real über 100 $\frac{\text{GeV}}{\text{c}}$ liegt, unter 80 $\frac{\text{GeV}}{\text{c}}$ eingeschätzt, wie es in den Streudiagrammen für CNN_{C2} und CNN_{B1} deutlich sogar bei über 120 $\frac{\text{GeV}}{\text{c}}$ (Realwert) zu erkennen ist.



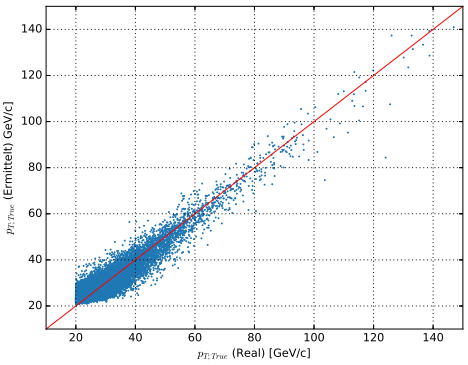
(a) CNN_{C1}



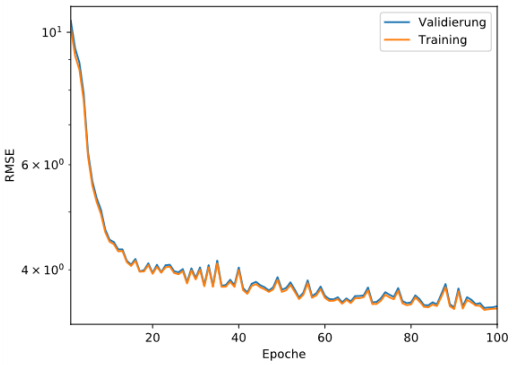
(b) CNN_{C1}



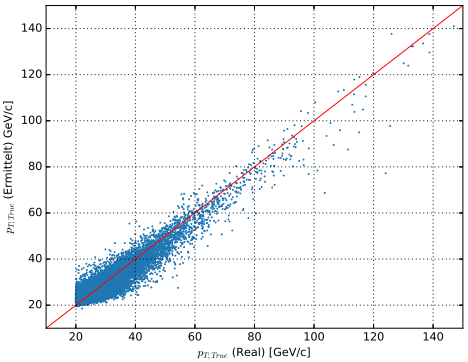
(c) CNN_{C2}



(d) CNN_{C2}



(e) CNN_{B1}



(f) CNN_{B1}

Abbildung 14: Logarithmische Lernkurven und Streudiagramme der CNN-Modelle CNN_{C1} und CNN_{B1} und CNN_{A1} nach 100 Epochen Training.

Aufgrund der beschriebenen Verbesserung des RMSE sowie der über die Streudiagramme geprüften Adäquanz der Ergebnisse ist das beste CNN-Modell im betrachteten Parameterraum CNN_{C1} mit $f = f_1 = f_2 = 128$, $p = [2,2]$, $r_{d_1} = 0{,}2$, $n_F = 256$, $l_F = 2$, $r_{d_2} = 0{,}2$, $r_l = 0{,}0001$ und einem RMSE von $3{,}29 \pm 0{,}09$ (Validierung) bzw. $3{,}25 \pm 0{,}09$ (Training).

Dies ist insofern irritierend, als dass der Sinn eines CNN eigentlich daraus bestehen sollte, einen Mehrwert aus der Lokalität der Eigenschaften der Inputdaten zu erhalten. Mit einem Filter der Größe 128 ist kaum Lokalität bei max. 140 Konstituenten vorhanden, zumal viele Modelle

unter 100 Konstituenten aufweisen.

4.5 Vergleich der Performance des besten CNN- und einfachen Feedforward-Modells

Für einen Vergleich der Vorhersagequalität der ermittelten besten Modelle mit einer einfachen Feedforward- sowie einer eindimensionalen CNN-Struktur sollen nun 15000 Testdatensätze verwendet werden, die bei der Modellauswahl und dem Modelltraining noch nie verwendet wurden. Damit wird berücksichtigt, dass durch die Modellselektion anhand der Validierungsperformance eine Optimierung auf die Validierungsdatensätze vorliegen könnte[26].

Zur Einschätzung der Modellfluktuation und zur Angabe einer finalen Performance werden die Modelle FF_{E1} und CNN_{C1} je fünf Mal mit den bisherigen Trainingsdatensätzen trainiert und das RMSE auf den Testdatensätzen berechnet. Die Ergebnisse für sind in Tab. 22 zu sehen.

Typ (RMSE)	$\boldsymbol{CNN_{C1}}$	$\boldsymbol{FF_{E1}}$
Maximum (Test)	$3{,}42 \pm 0{,}09$	$3{,}40 \pm 0{,}07$
Minimum (Test)	$3{,}37 \pm 0{,}09$	$3{,}35 \pm 0{,}07$
Mittel (Test)	$3{,}39 \pm 0{,}06$	$3{,}37 \pm 0{,}05$
Bisheriger Bestwert (Validierung)	$3{,}29 \pm 0{,}09$	$3{,}28 \pm 0{,}07$

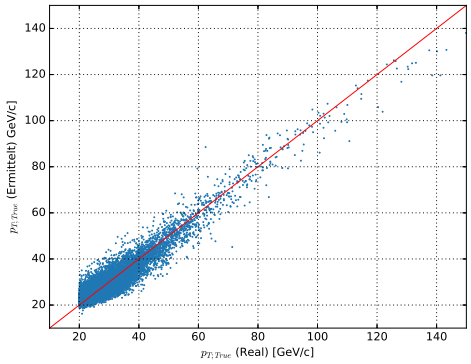
Tabelle 22: Erhaltene RMSE-Werte für die Testdatensätze mit den Modellen CNN_{C1} und FF_{E1} im Vergleich mit dem bisherigen Bestwerten auf den Validierungsdatensätzen. Die Unsicherheiten wurden mit einem Signifikanzbereich von 99,8% der Standardabweichung berechnet.

Es zeigt sich, dass die RMSE-Werte für beide Modelle sowohl auf den Test- als auch auf den Validierungsdatensätzen im Vergleich keinen signifkanten Unterschied aufweisen. Aus diesem Grund ist eine genauere Untersuchung der Vorhersagen beider Modelle erforderlich.

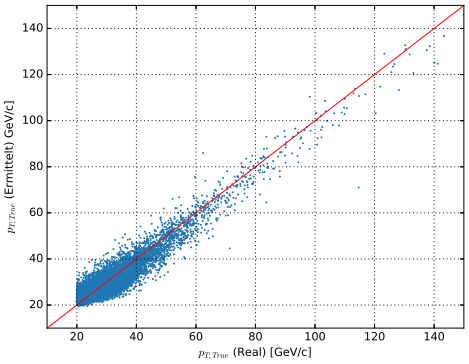
In Abb. 15 zeigt sich, dass beide Modelle ähnliche Verteilungen bzgl. ihrer Vorhersage aufweisen. Dennoch gibt es Unterschiede bzgl. ihrer Details.
So ist beim CNN_{C1}-Modell (Abb. 15(b)) ein einzelner Punkt sehr schlecht ermittelt worden. Dieser Punkt (Jet) hat ein reales $p_{T;True}$ von etwa 115 $\frac{\text{GeV}}{\text{c}}$, wird aber bei ca. 71 $\frac{\text{GeV}}{\text{c}}$ eingeordnet. Das Feedforward-Modell gibt hierfür 109,48 $\frac{\text{GeV}}{\text{c}}$ zurück und weist auch sonst keine derart extremen Fehleinschätzungen auf. Interessant ist, dass beide Modelle im Bereich 60 $\frac{\text{GeV}}{\text{c}}$ bis 90 $\frac{\text{GeV}}{\text{c}}$ teils ähnliche Punkte mit großer Abweichung vom Realwert aufweisen.
Für den mit wenigen Trainingsdatensätzen ausgestatteten Bereich hoher $p_{T;True}$ ab etwa 120 $\frac{\text{GeV}}{\text{c}}$ schätzt das FF_{E1}-Modell die Werte als geringer ein als das CNN_{C1}-Modell.

Beide Modelle sind somit in der Lage, die Hintergrundkorrektur durchzuführen.

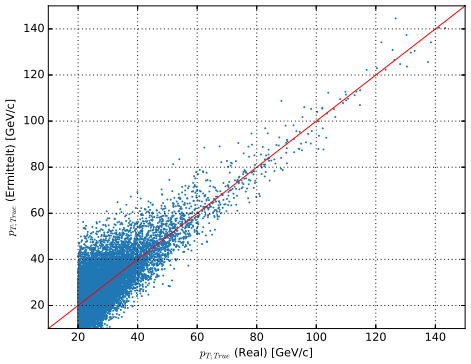
[26] Auch wenn die Verteilung der Daten für alle Datensätze homogen ist, soll dieser Faktor vollkommen ausgeschlossen werden, zumal leicht differierende Werte des $p_{T;True}$ womöglich unterschiedlich gut zu optimieren sein *könnten*.



(a) FF_{E1}



(b) CNN_{C1}



(c) Standardmethode über Korrektur durch $p_{T;Gemessen}$ - Hintergrunddichte·Jetfläche

Abbildung 15: Streudiagramme der je besten ermittelten Modelle mit einer einfachen Feedforward- und CNN1D-Struktur. Zum Vergleich ist auch eine herkömmliche Methode dargestellt.

Ein Vergleich der Ergebnisse mit einer herkömmlichen Korrekturmethode[27] für den Hintergrund über die Differenz auf gemessenem Transversalimpuls und dem Produkt aus Hintergrunddichte·Fläche zeigt deutliche Verbessserungen der Vorhersage v.a. für Bereiche geringerer Transversalimpulse.

5 Zusammenfassung

Die mehrstufige Optimierung von Modellparametern durch die Ergebnisse von Rastersuchen im Parameterraum konnte sowohl für die CNN- als auch die FF-Modelle die Qualität der Regressionen verbessern. Insbesondere die Berechnung von hohen $p_{T;True}$-Werten mit einer geringen Anzahl an Datensätzen ab etwa 100 $\frac{\text{GeV}}{\text{c}}$ konnte damit wesentlich verbessert werden und die Vorhersagequalität beider Netze ist der Korrekturmethode über die Subtraktion der Hintergrunddichte·Jetfläche überlegen.

Für die simplen Feedforward-Modelle wurde die Performance auf den Validierungsdatensätzen mit dieser Strategie von $3{,}55 \pm 0{,}07$ (FF_{A2}) auf $3{,}28 \pm 0{,}07$ (FF_{E1}) signifikant verbessert. Dabei wurde im betrachteten Parameterraum erkannt, dass eine stetige Erhöhung der Anzahl an Neuronen und Schichten in den Zweinetzen und dem finalen Netz keine konsequente Verbesserung der Regressionsqualität herbeiführen muss. Overfitting konnte durch eine der Modellkomplexität

[27] Für weitere Informationen siehe [8].

angepasste Dropout-Rate verringert werden.

Für die eindimensionalen CNN-Modelle wurde die Validierungs-Performance mit allen Raster- und Detailsuchen von 3,47±0,09 (CNN_{A1}) auf 3,29±0,09 (CNN_{C1}) ebenso signifikant verbessert. Die Verbesserungen konnten durch eine Erhöhung der Filtergröße und der Anzahl an Neuronen im finalen Netz erreicht werden. Eine Erhöhung der Anzahl an Konvolutionsschichten bedeutete für die betrachteten Parameterräume keine Optimierung der Modellperformance, sondern eine Reduktion derselben.

Insgesamt konnte gezeigt werden, dass beide Modellstrukturen in der Lage sind, das $p_{T;True}$ der Jets auf Basis von $\eta_{Konst.}$, $\phi_{Konst.}$ und $p_{T_{Konst.}}$ von bis zu 140 Konstituenten mit den größten Transversalimpulsen im Rahmen der simulierten Daten zu ermitteln, was sich auch in der Test-performance von $3{,}39 \pm 0{,}06$ (CNN_{C1}) und $3{,}37 \pm 0{,}05$ (FF_{E1}) zeigt.

Allgemein zur Optimierung von neuronalen Netzen konnte in der Arbeit erkannt werden, dass eine Erhöhung der Komplexität eines Modells durch mehr Neuronen oder Schichten bei fixierten anderen Parametern nicht automatisch die Ergebnisqualität verbessern muss und auch die Anzahl der Epochen kritische Werte erreichen kann, ab der sich das Lernverhalten drastisch verändert, was das nichtlineare Trainings-Verhalten verdeutlicht.

Literatur

[1] Günter Daniel Rey, Karl F. Wender: *Neuronale Netze - Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung.* 2. Auflage, Verlag Hans Huber, 2011.

[2] Sebastian Raschka: *Python Machine Learning.* 1. Auflage, Packt Publishing Ltd., 2015.

[3] Kaiming He, Xiangyu Zhang, Shaoqing, Jian Sun: *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.* [https://arxiv.org/abs/1502.01852], 2015, abgerufen am 07.09.17.

[4] Diederik Kingma, Jimmy Ba: *Adam: A Method for Stochastic Optimization.* [https://arxiv.org/abs/1412.6980v8], 2014, abgerufen am 07.09.17.

[5] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, Ping Tak Peter Tang: *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.* [https://arxiv.org/abs/1609.04836], 2017, abgerufen am 07.09.17.

[6] Pierre Baldi, Peter Sadowski: *Understanding Dropout.* Advances in Neural Information Processing Systems 26, [http://papers.nips.cc/paper/4878-understanding-dropout.pdf], 2013, abgerufen am 15.09.17.

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Journal of Machine Learning Research 15, [https://www.cs.toronto.edu/ hinton/absps/JMLRdropout.pdf], 2014, abgerufen am 16.09.17.

[8] ALICE Collaboration *Measurement of Event Background Fluctuations for Charged Particle Jet Reconstruction in Pb-Pb collisions at* $\sqrt{s_{NN}}$ *=2.76TeV.* [https://arxiv.org/pdf/1201.2423.pdf],

[9] Christian Klein-Bösing: *Study of the Quark-Gluon Plasma with Hard and Electromagnetic Probes.* Habilitation, [https://www.uni-muenster.de/imperia/md/content/physik_kp/agwessels /thesis_db/ag_wessels/klein-boesing_c_2013_habilitation.pdf], 2013, abgerufen am 17.09.17

[10] Philipp Kähler: *Untersuchung des Antwortverhaltens der Jet-Rekonstruktion in Schwerionenkollisionen.* Bachelorarbeit, [https://www.uni-muenster.de/imperia/md/ content/physik_kp/agwessels/thesis_db/ag_wessels/kaehler_2011_bachelor.pdf], 2011, abgerufen am 17.09.17.

[11] Hendrik Poggenburg: *Characterization of Heavy-Ion-Background in Jet-Reconstruction.* Bachelorarbeit, [https://www.uni-muenster.de/imperia/md/content/physik_kp /agwessels/thesis_db/ag_wessels/poppenborg_h_2011_bachelor.pdf], 2011, abgerufen am 18.09.17

[12] D. Kriesel: *Ein kleiner Überblick über neuronale Netze.*[http://www.dkriesel.com/_media/ science/neuronalenetze-de-zeta2-2col-dkrieselcom.pdf], abgerufen am 15.09.2017.

[13] Jan Fiete Grosse-Oetringhaus: *Introduction to Heavy-Ion Physics - Part II.* Vortrag am CERN, [https://cds.cern.ch/record/2275545], 24.07.2017, abgerufen am 18.09.17.

Sonstige Quellen

[14] Persönliche Kommunikation mit Florian Jonas. 2017.

[15] Persönliche Kommunikation mit Dr. Rüdiger Haake. 2017.

6 Danksagungen

Ich bedanke mich bei Dr. Klein-Bösing und Dr. Rüdiger Haake für die Bereitstellung und engagierte Betreuung dieses spannenden Themas. Außerdem möchte ich mich bei Florian Jonas für das Generieren der Simulationsdaten bedanken. Für die schnellen Antworten auf Fragen möchte ich Dr. Rüdiger Haake und Florian Jonas einen besonderen Dank aussprechen.