Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Josef Matthias Tremmel

born in Öhringen (Germany)

2022

Studien zur Verbesserung der Signalverarbeitung des ATLAS Level-1 Calorimeter Triggers durch künstliche neuronale Netze

This Bachelor Thesis has been carried out by Josef Matthias Tremmel at the
Kirchhoff-Institute for Physics Heidelberg
under the supervision of
Prof. Dr. Hans-Christian Schultz-Coulon

Abstract

Due to the planned Phase-II upgrade of the Large Hadron Collider, the number of occurring proton-proton collisions will increase to 140-200. The resulting stronger background processes are challenging the ATLAS Level-1 Calorimeter Trigger. Its tasks include the detection and energy reconstruction of measured particles in the calorimeters of the ATLAS detector.
In this thesis, it was investigated whether machine learning algorithms can improve the performance of the trigger. For this purpose, artificial neural networks were trained with simulated data from the hadronic calorimeter. For a first feasibility study, data with an average number of proton-proton collisions of 40 were used.
The investigation of the performance of the networks compared to the current system has shown that for particles a more accurate reconstruction of their transversal energy is possible using a neural network, especially in the low energy range. In addition, the networks achieve better efficiency and purity when detecting events.

Zusammenfassung

Durch das geplante Phase-II Upgrade des Large Hadron Colliders wird die Anzahl der auftretenden Proton-Proton-Kollisionen auf 140-200 steigen. Die dadurch stärkeren Untergrundprozesse stellen eine Herausforderung für den ATLAS Level-1 Calorimeter Trigger dar. Zu seinen Aufgaben zählt die Detektion und Energierekonstruktion gemessener Teilchen in den Kalorimetern des ATLAS-Detektors.
In dieser Arbeit wurde untersucht, ob Machine Learning Algorithmen die Leistungsfähigkeit des Triggers verbessern können. Dafür wurden künstliche neuronale Netze mit simulierten Daten aus dem hadronischen Kalorimeter trainiert. Für eine erste Machbarkeitsstudie kamen Daten mit einer mittleren Anzahl Proton-Proton-Kollisionen von 40 zum Einsatz.
Die Untersuchung der Leistungsfähigkeit der Netzwerke im Vergleich zu dem aktuellen System hat gezeigt, dass insbesondere für niedrigenergetische Pulse eine genauere Rekonstruktion der transversalen Energie durch ein neuronales Netz möglich ist. Außerdem erreichen die Netzwerke bei der Detektion von Ereignissen eine bessere Effizienz und Reinheit.

Inhaltsverzeichnis

Kapitel 1: Einleitung

Das A Toroidal LHC ApparatuS (ATLAS)-Experiment ist ein Teilchendetektor am Large Hadron Collider (LHC), dem weltweit größten Teilchenbeschleuniger. In diesem werden Protonen auf eine Geschwindigkeit von bis zu 99.9999991 % der Lichtgeschwindigkeit beschleunigt und zur Kollision gebracht. Durch die Energie der Kollision entstehen Teilchen, deren Eigenschaften von ATLAS gemessen werden. So konnte im Jahre 2012 das Higgs Boson durch die Experimente ATLAS[1] und Compact Muon Solenoid (CMS)[2] nachgewiesen werden.
Eine wichtige Aufgabe im ATLAS-Experiment ist die Detektion und Energierekonstruktion von Teilchen in den verschiedenen Detektorbereichen. Durch geplante Upgrades des LHCs zu höheren Luminositäten steht der ATLAS-Detektor vor der Herausforderung mit höheren Produktionsraten umgehen zu können. In [3] konnte für das elektromagnetische Kalorimeter gezeigt werden, dass bei diesen Bedingungen die Energierekonstruktion von Ereignissen durch neuronale Netze dem bisherigen Algorithmus überlegen ist. Diese Arbeit stellt den ersten Schritt dar, um zu überprüfen, ob dies auch für das hadronische Kalorimeter der Fall ist.
Im ersten Kapitel dieser Arbeit wird auf das aktuelle Triggersystem im ATLAS eingegangen, gefolgt von einem Kapitel über die Grundlagen von künstlichen neuronalen Netzen. Kapitel 3 geht genauer auf die Datenstruktur ein und behandelt die verschiedenen Rauschquellen. Die Beschreibung der verwendeten Netzwerke und die Anpassung an die verwendeten Daten erfolgt in Kapitel 4. Danach folgt die Untersuchung der Leistungsfähigkeit der Netzwerke und der Vergleich mit dem aktuellen Trigger. Abschließend werden in Kapitel 6 die Ergebnisse zusammengefasst.

1.1 Der Large Hadron Collider

Der LHC ist ein Proton-Proton-Teilchenbeschleuniger der Europäischen Organisation für Kernforschung (CERN). Er besteht aus einem Hauptring, der einen Umfang von 26.7 km aufweist und bis zu 170 m unter der Oberfläche liegt. In einem System aus Vorbeschleunigern werden Protonen auf 450 GeV vorbeschleunigt und in den Hauptring injiziert. Dort werden sie auf eine Energie von 6.5 TeV beschleunigt. Dies entspricht einer Schwerpunktsenergie von 13 TeV. An den insgesamt vier Kreuzungs-

punkten der beiden Strahlen befinden sich die Experimente: ATLAS, A Large Ion Collider Experiment (ALICE), CMS und Large Hadron Collider beauty (LHCb).[4]
Die Protonen sind im LHC in Protonen-Pakete, sogenannten Bunches, mit je

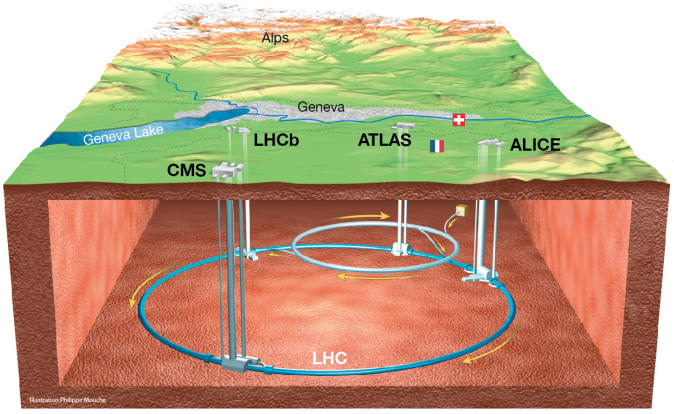


Abbildung 1.1: Übersicht des Large Hadron Collider mit dem Beschleunigungsring und den 4 verschiedenen Experimenten[5]

1.2×10^{11} Protonen aufgeteilt. Das Füllschema legt dabei die Anordnung der Bunches während eines Umlauf im Hauptring fest. Die Bunches der entgegenlaufenden Protonenstrahlen kollidieren in sogenannten Bunch Crossings (BCs) zu festgelegten Zeiten mit einer Frequenz von 40 MHz. Zwei wichtige Größen bei Kollisionen ist der Pile-Up $\langle\mu\rangle$ und die Luminosität $\mathcal{L}$. Die Luminosität ist ein Maß für die Strahlintensität. Der Zusammenhang zwischen Luminosität, Wirkungsquerschnitt σ und der Ereignisrate $\frac{dN}{dt}$ ist durch folgende Gleichung gegeben:

$$\frac{dN}{dt} = \sigma \cdot \mathcal{L}. \tag{1.1.1}$$

Die Ereignisrate von Prozessen hängt vom Wirkungsquerschnitt des Prozesses und der Luminosität ab. Seltene Prozesse werden somit bei einer höheren Luminosität öfters beobachtet. Der Pile-Up $\langle\mu\rangle$ ist direkt von der Luminosität abhängig und beschreibt die Anzahl der inelastischen Streuungen pro BC. Während des letzten Runs im Jahr 2018 wurde ein $\langle\mu\rangle$ von 36 erreicht bei einer peak Luminosität von $19 \cdot 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ [6]. Nach dem geplanten Phase-II Upgrade soll die peak Luminosität auf $7.5 \cdot 10^{34}\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ mit einer geschätzten Erhöhung von $\langle\mu\rangle$ auf 140-200 ansteigen [7]. Daraus resultieren deutlich mehr produzierte Teilchen, deren Signal vom Detektor verarbeiten werden muss.

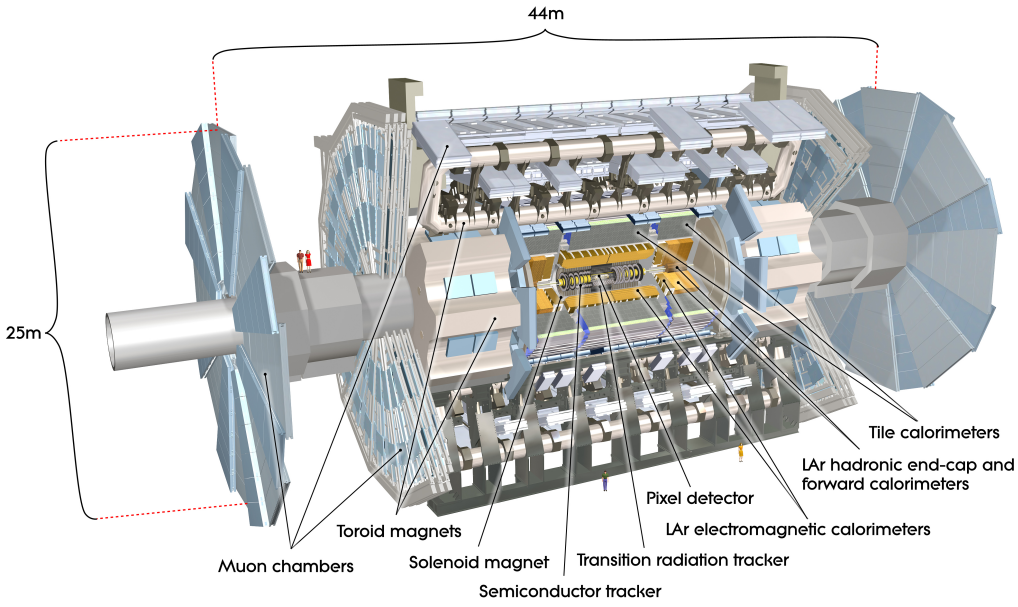


Abbildung 1.2: Das ATLAS Experiment mit den verschiedenen Teilbereichen[8]

1.2 Das ATLAS-Experiment

Der ATLAS-Detektor ist mit einem Durchmesser von 25 m, einer Länge von 46 m und einem Gewicht von 7000 t der weltweit größte Teilchendetektor an einem Beschleuniger. ATLAS und CMS sind Allzweckdetektoren am LHC, deren unterschiedlicher Aufbau unabhängige Messungen erlaubt. Der ATLAS-Detektor ist symmetrisch um die Strahlachse gebaut und hat eine vorwärts-rückwärts Symmetrie bezüglich des Kollisionspunktes der aufeinandertreffenden Strahlen. Der Aufbau besteht aus mehreren Teilsystemen (siehe Abbildung 1.2), um die Position, Energie und den Impuls der entstandenen Teilchen zu bestimmen. Ganz im Inneren um den Interaktionspunkt befindet sich das Tracking System. Ein Magnetfeld lenkt geladene Teilchen auf eine gekrümmte Bahn, aus deren Form die Ladung und der Impuls bestimmt wird. Aufgebaut ist das Tracking System aus einem sehr hochauflösenden Pixeldetekor, der von einem Silizium-Streifen-Detektor umschlossen wird. Die äußerste Schicht des Tracking Systems bildet der Übergangsstrahlungsdetektor.
Zur Energiebestimmung besitzt der ATLAS-Detektor zwei Kalorimeterschichten außerhalb des Tracking Systems. In den Kalorimetern wird die Energie von Elektronen, Photonen und Hadronen in Form von Schauern deponiert. Die verwendeten Kalorimeter haben eine Sandwichstruktur aus abwechselnden Schichten aus einem Absorber und einem aktiven Detektormaterial. Im Absorber bilden sich Schauer, die im aktiven Material nachgewiesen werden. Aus dem Signal des aktiven Detektormaterials wird die deponierte Energie rekonstruiert. Das erste Kalorimeter bildet das elektromagnetische Kalorimeter, welches Blei als Absorber und flüssiges Argon als

aktives Material verwendet. In ihm wird die Energie von Elektronen und Photonen präzise gemessen. Die Hadronen deponieren nur einen Teil ihrer Energie im elektromagnetischen Kalorimeter. Deshalb wird dieses von dem hadronischen Kalorimeter umschlossen.
In dieser Arbeit ist das Tile Long Barrel (TileLB) von besonderer Bedeutung. Es ist ein Teil des hadronischen Kalorimeters und deckt den Bereich $|\eta^1| < 1.0$ ab. Das TileLB ist aufgebaut aus vielen abwechselnden Schichten aus dem passiven Absorber Stahl (3mm) und dem aktiven Detektormaterial in Form von Plastikszintillatoren (14mm). Hadronen deponieren ihre Energie in Form eines hadronischen Schauers, welcher über von ihm angeregte Photonen in den Plastikszintillatoren gemessen wird.
Das Myon-System bildet die äußerste Schicht des ATLAS-Detektors. Da Myonen eine ca. 200-fach größere Masse besitzen als Elektronen, sorgt die Bremsstrahlung erst bei hohen Energien für einen signifikanten Energieverlust. Die zuvor beschriebenen Detektorsysteme werden von den Myonen mit geringem Energieverlust durchdrungen. Erst im Myon-System wird der Impuls gemessen und die Bahn vermessen [4] [9].

1.3 Der ATLAS Level-1 Calorimeter Trigger

Ein abgespeichertes Ereignis des ATLAS-Detektors hat eine Größe von 1.5 MB. Bei einer Kollisionsfrequenz der Bunches von 40 MHz würden innerhalb einer Sekunde mehrere TB an Daten anfallen. Die Aufgabe eines Triggers ist es, lediglich die interessanten Ereignisse zu selektieren, damit diese für eine spätere Analyse abgespeichert werden können. ATLAS verwendet für die Reduktion der Datenrate einen zweistufigen Trigger aus einem hardwarebasierten Level-1 (L1) Trigger und einem softwarebasierten High-Level Trigger (HLT). Der L1-Trigger selektiert aus dem Detektorsignal, mit einer Frequenz von 40 MHz, potentiell interessante Ereignisse mit einer maximalen Design Rate von 100 kHz. Die für die Speicherung vorgesehenen Ereignisse reduziert der HLT auf 1kHz.
Der L1-Trigger muss mit einer Latenz von $2.5\,\mu s$ nach einer Kollision eine Entscheidung treffen, da die Signale während der Prozessierung zwischengespeichert werden müssen und die Zwischenspeicher volllaufen. Um dies zu bewerkstelligen, werden vom L1-System viele Detektorbereiche parallel verarbeitet. Deshalb ist er aufgeteilt in den Level-1 Calorimeter Trigger (L1Calo) und L1 Muon Trigger (L1Muon), deren Ergebisse vom Central Trigger Processor (CTP) für eine finale L1 Entscheidung verwendet werden. Der Input von L1Calo besteht 7168 Trigger Tower (TT), welche aus den $190 \cdot 10^3$ Zellen in dem elektromagnetischen und hadronischen Kalo-

[1] Pseudo-Rapidität $\eta = -\log\left(\tan\left(\frac{\theta}{2}\right)\right)$, mit Winkel θ relative zur Strahlachse, Azimuthwinkel ϕ : Winkel um die Strahlachse

rimeter kombiniert werden. Für einen TT wird das Signal mehrerer Zellen aus den beiden Kalorimetern analog aufsummiert. Die dadurch höhere Granularität liegt im TileLB bei $\Delta\eta \times \Delta\phi = 0.1 \times 0.1$. Das Signal der TT gelangt über 30-70 m lange Twisted-Pair-Kabel zu den Receivern des L1Calo, welcher sich nicht direkt am Detektor befindet. Mithilfe von Gain-Faktoren wird das analoge Signal skaliert und die gemessene Energie E in die transversale Energie E_T umgerechnet[2]. Das kalibrierte Signal erreicht anschließend den PreProcessor (PPr) des L1Calo. Im PPr wird zuerst ein Offset auf das Signal addiert, damit das Signal für den folgenden Analogue-to-Digital Converter (ADC) das korrekte Pedestal[3] von 32 Counts erhält. Im ADC wird das analoge Signal mit einer Frequenz von 80 MHz digitalisiert, sodass mit den gewählten Gain-Faktoren im Reciver 1 ADC Count 250 MeV entsprechen. Die 80 Mhz Digitalisierung wird für den Fall eines saturierten Pulse benötigt. Ist dies nicht der Fall wird auf 40 MHz reduziert. Aus dem digitalisierten Signal werden anschließend die Pulse, welche signifikanten Energiedepositionen entsprechen, ausgewählt. Für diese Pulse findet eine Bestimmung des korrekten BCs[4] sowie der transversalen Energie statt. Diese beiden Aufgaben des PPrs sind für diese Arbeit von besonderer Bedeutung.
Die Informationen über Position und Energie von detektierten Signalen werden von dem JetEnergy Processor (JEP) und Cluster Processor (CP) weiterverarbeitet. Der CP sucht nach Elektronen, Photonen und τ-Lepton-Kandidaten in Form von schmalen Clustern in den Kalorimetern. Er unterscheidet elektromagnetische und hadronische Schauer, während im JEP nach Jet-Kandidaten gesucht, deren Eigenschaften bestimmt und die globale Energiesumme berechnet wird. Der CTP bestimmt mit den Informationen aus dem L1Calo und dem L1Muon, ob das Ereignis verworfen oder behalten wird. Bei letzterem wird ein Level-1 Accept (L1A) erteilt und die Informationen über die Kandidaten, wie z. B. ihre Koordinaten, an den HLT in Form von Regions of Interest (ROIs) weitergegeben. Der HLT benutzt anders als der L1 eine feinere Granularität der Kalorimeter, Präzisionsmessungen des Muon-Systems und Spurinformationen aus dem Tracking System, um die Auswahl der Kandidaten von L1 weiter einzuschränken [10] [11].

Finite-Impulse-Response-Filter

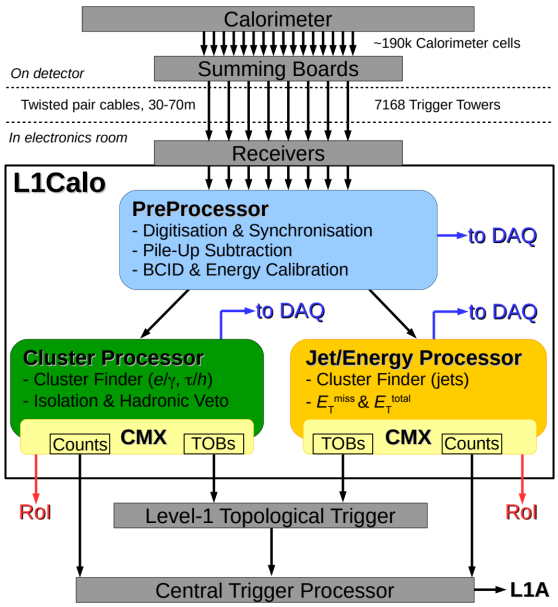
Der Finite-Impulse-Response (FIR)-Filter befindet sich im PPr des L1Calo. In den Kalorimetern gemessene Teilchen werden durch einen charakteristischen Puls im

[2]Die Umrechnung von E zu E_T findet für das hadronische Kalorimeter in den Recivern satt, während dies für das elektromagnetische Kalorimeter schon in der Elektronik am Detektor geschieht.

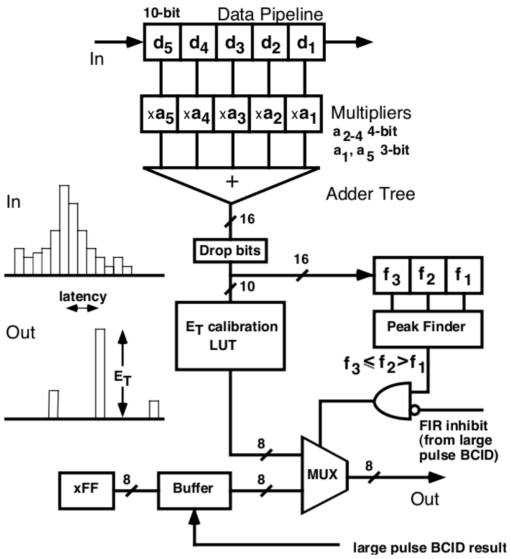
[3]Das Pedestal ist die Grundlinie des ADC-Outputs und hat einen nominellen Wert von 32 ADC Einheiten.

[4]Das korrekte BC ist definiert durch die Position an der der Pulse die maximale Amplitude hat.

digitalen Signal sichtbar. Der FIR-Filter bestimmt das korrektes BC und die transversale Energie (E_T) dieser Pulse in dem digitalen Signal mit einem Optimal Filter, der von Cleland und Stern in [12] beschrieben wird. Der FIR-Filter stellt die erste Stufe dar, um interessante Ereignisse zu filtern. In der folgenden Arbeit werden neuronale Netze traininert, um die Aufgaben des FIR-Filters zu übernehmen. Da dieser dabei als Vergleich dient, wird im Folgenden etwas genauer auf seine Funktionsweise eingegangen:



(a) Architektur des L1Calo [10]



(b) Aufbau des FIR-Filters [13]

Abbildung 1.3: Übersicht über den L1Calo des ATLAS Experiments (a) und des im PreProcessor eingesetzten FIR-Filters (b). Bei der Darstellung des FIR-Filter fehlt die erwähnte Pedestal Korrektur.

Alle 25 ns (Abstand zwischen zwei BCs) wird, gemäß

$$f(t) = \sum_{i}^{5} a_i \cdot d(t+i-2) - f_{pedCorr}(t), \tag{1.3.1}$$

der FIR-Output mit fünf aufeinander folgenden Werten des ADC-Signals $d(i)$ berechnet. Eine gewichtete Summe aus den ADC-Outputs skaliert mit festgelegten Konstanten a_i. Dadurch wird das Signal-to-Noise Verhältnis für die erwartete Pulsform (für TileLB siehe Abbildung 3.1) erhöht. Die Subtraktion von $f_{pedCorr}(t)$ von der gewichteten Summe stellt die Pedestal Korrektur dar. Da es aufgrund des Füllschemas (siehe 3.2) im LHC zu Verschiebungen des Pedestals während eines kompletten Umlaufs kommen kann, müssen diese mit $f_{pedCorr}(t)$ korrigiert werden. Dabei wird der FIR-Output über 2^{16} Umläufe gemittelt und anschließend die Differenz zum mittleren Pedestal berechnet. Energiedepositionen in Form von Pulsen im FIR-Output

werden vom Peak-Finder-Algorithmus detektiert und parallel ihre Energie über eine Look-Up Table (LUT) berechnet. Der Peak-Finder vergleicht 3 aufeinander folgende FIR-Outputs. Befindet sich an der mittleren Position das Maximum der drei Signale, gibt der Peak-Finder eine 1 aus. Ist dies nicht der Fall, wird eine 0 ausgegeben. In diesem Fall wird der Output der LUT auf Null gesetzt. Auf diese Weise können nur detektierten Pulsen Energien ungleich null zugeordnet werden. Das Blockdiagramm des FIR-Filters in Abbildung 1.3b veranschaulicht die Funktionsweise. Die LUT, charakterisiert durch eine Slope, ein Offset und einen Noisecut, berechnet die Energie E_T eines detektierten Pulses:

$$\text{LutOut} = \begin{cases} (\text{LutIn} \cdot \text{Slope} - \text{Offset} + 2048) \gg 12 & \text{falls LutIn} \cdot \text{Slope} - \text{Offset} > \text{Noisecut} \\ 0 & \text{sonst} \end{cases} . \tag{1.3.2}$$

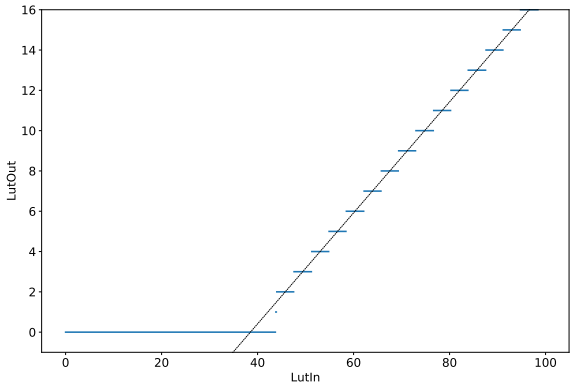


Abbildung 1.4: Die LUT mit LutOut als Funktion von LutIn. Sie ist Teil des FIR-Filters und bestimmt die transversale Energie E_T von detektierten Pulsen. Der Parameter Slope und Offset sind so gewählt, dass 1 LutOut 1 GeV entspricht.

Aufgrund der Hardwareimplementierung hat LutIn eine Größe von 10 Bit und für LutOut stehen 8 Bit zur Verfügung. Dies wird berücksichtig, indem $f(t)$ für LutIn auf 10 Bits reduziert wird und für LutOut 12 Bits abgeschnitten werden. Der Noisecut verhindert fake Hits durch Rauschen im Signal und setzt LutOut auf Null falls die Energie unterhalb einer Schwelle liegen sollte [4][14].
Bei den Untersuchungen in dieser Arbeit wurden die Daten aus einem TT im TileLB mit einer Pseudo-Rapidität $\eta = -0.25$ verwendete. Für die Implementierung des FIR-Filters werden Parameter verwendet, die für diesen Detektorbereich in [4] bei einem $\langle\mu\rangle$ von 40 optimiert wurden.

Der Noisecut für die in Tabelle 1.1 angegebenen Parameter wurde für die verwendeten Daten über die Methode der Integrated Occupancy bestimmt [14]. Die

Parameter	verwendeter Werte
a_1	1
a_2	9
a_3	15
a_4	10
a_5	4
db	5
Slope	1128
Offset	43428

Tabelle 1.1: Parameter aus [4] für die Implementierung des FIR-Filters

Integrated Occupancy einer beliebigen Energie beschreibt den Anteil aller detektierten Ereignisse mit mindestens dieser Energie. Der Noisecut entspricht der Energie, bei der ein Anteil von 0.5 % erreicht wird. Die Abbildung 1.5 zeigt die Integrated Occupancy für den implementierten FIR-Filter und den verwendeten Daten. Der somit bestimmte Noisecut beträgt 5928. Dies entspricht etwa einer transversalen Energie von $E_T = 1.4\,\mathrm{GeV}$.

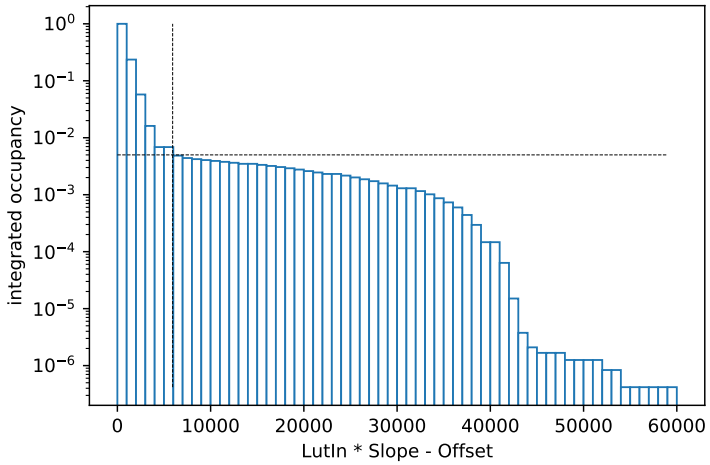


Abbildung 1.5: Die Integrated Occupancy ist in Abhängigkeit von (LutIn*Slope-Offset) dargestellt, da dies für den Fir-Filter die relevante Größe darstellt. Für den Schnittpunkt mit 0.5 % wurde zwischen den Bins linear interpoliert.

Kapitel 2: Künstliche Neuronale Netze

Künstliche Neuronale Netze (engl. : Artifical Neural Networks (ANN)) bilden eine Kategorie von Machine Learning (ML) Algorithmen. Was ML eigentlich bedeutet, beantwortet T. Mitchell 1997 : "Man sagt, dass ein Computerprogramm dann aus Erfahrungen E in Bezug auf eine Aufgabe T und ein Maß für die Leistung P lernt, wenn seine durch P gemessene Leistung bei T mit der Erfahrung E anwächst"[15]. In den letzten Jahren haben ANN stetig mehr Aufmerksamkeit bekommen und es geschafft, immer komplexere Probleme lösen zu können, wie z.B. in der Bild- und Spracherkennung. Im November 2020 ist dem Netzwerk AlphaFold von Deep-Mind ein Durchbruch in der Biologie gelungen, indem genaue Vorhersagen über die 3D-Struktur von Proteinen anhand ihrer Aminosäuresequenz möglich wurden [16]. Die Idee von ANN ist jedoch schon älter. Im Jahr 1943 stellten McCulloch und Pitts das erste Mal die Idee von einem neuronalen Netz vor [17]. Sie ließen sich von Neuronen lebender Organismen und deren Signalverarbeitung inspirieren und es gelang ihnen, jede logische Operation mit künstlichen Neuronen durchzuführen. Die Verbindung zu biologischen Neuronen stellt lediglich die Grundidee dar. Es ist keineswegs das Ziel von ANN, die Funktionsweise des Gehirns nachzubilden [18]. Der Baustein von neuronalen Netzen wurde 15 Jahre später von Rosenblatt mit dem Perceptron veröffentlicht und auch die Verknüpfung mehrerer Perceptrons in einer Schicht und Trainingsalgorithmen wurden entwickelt. Die Forschung kam zum Stillstand als Minsky und Pepert 1969 Nachteile des Perceptron hervorhoben, wie z.B. die nicht-Lösbarbeit des XOR-Problems[1]. In den Folgejahren konzentrierte sich die Forschung auf andere ML-Verfahren. Erst mit der gestiegenen Rechenkapazität zum Trainieren großer Netze, den verbesserten Trainingsalgorithmen und der Verfügbarkeit riesiger Datenmengen sind ANN in den letzten Jahren wieder populär geworden. Im ATLAS-Detektor finden ANN an vielen Stellen Anwendung:

- Bei der besseren Energierekonstruktion von Signalen im Liquid Argon Kalori-

[1]Das XOR-Problem ist eine Klassifikationsaufgabe, bei der die XOR-Operation implementiert werden muss. Dabei handelt es sich um nicht linear trennbare Daten, weshalb ein einzelnes Perceptron die Aufgabe nicht lösen kann, mehrere in Reihe dagegen schon.

meter für die high-luminosity Phase des LHC [3]

- Um mit einer lokalen Pile-Up Korrektur durch ANN die Auswirkungen des Pile-Up auf die Energie von detektieren Jets oder anderen Objekten zu reduzieren und damit die fehlende transversale Energie E_T^{miss} genauer bestimmen zu können [19]

- Die b-Jet Identifikation wird durch einen neuen Algorithmus mit ANN verbessert. [20]

Dies sind nur wenige Beispiele von vielen Anwendungen. Im folgenden Abschnitt werden die Grundlagen, das Training und die in dieser Arbeit benutzten Netzwerkarten genauer beschrieben. Die Informationen dafür wurden, wenn nicht anders gekennzeichnet, von [21] bezogen.

2.1 Grundlagen neuronaler Netze

Ziel eines neuronalen Netzes ist es, eine Funktion $f^*(\vec{x})$ zu lernen. Diese ist abhängig von den Input Daten $\vec{x}$, auch Feature genannt, und kann z. B. im Falle einer Klassifikation ein Bild $\vec{x}$ einer Klasse $y = f^*(\vec{x})$ zuordnen. Das neuronale Netz bestimmt die Kategorie y mit der gelernten Funktion $f(\vec{x};\theta)$. Die Parameter θ werden während des Trainings mit einem Trainingsdatensatz optimiert, sodass die wahre Funktion f^* möglichst genau durch f approximiert wird. $f(\vec{x};\theta)$ ist eine Verschachtelung vieler mathematischer Funktionen, die auch als einzelne Neuronen bezeichnet werden. Ein einzelnes Neuron kann als mathematische Funktion gesehen werden. Es gibt einen Input $\vec{x}$, welcher vektoriell sein kann, und einen von $\vec{x}$ abhängigen Output $y(\vec{x})$. Der Output entsteht aus einer gewichteten Summe aus dem Input, Gewichten $\vec{\theta}$, Bias b und einer Aktivierungsfunktion f:

$$y(\vec{x}) = f(\vec{\theta} \cdot \vec{x} + b). \tag{2.1.1}$$

Die Aktivierungsfunktion bestimmt dabei, welcher Net-Input $(\vec{\theta} \cdot \vec{x} + b)$ zu großen Ausgaben des Neurons führt und steuert so die Reaktion des Neurons auf die Eingabe. Neuronen, welche parallel aufgebaut sind, also den gleichen Input erhalten, aber unterschiedliche Gewichte haben, bilden eine Schicht (engl. : Layer). Damit hat das Netz die Möglichkeit, verschiedene Eigenschaften des Inputs getrennt hervorzuheben. Bei neuronalen Netzen kommt es häufg vor, dass Schichten übereinander platziert werden und der Output einer Schicht den Input der nächsten Schicht darstellt. Die erste Schicht, in der sich die Features befinden, wird Eingabeschicht genannt. Die letzte Schicht mit dem Ausgabeneuron heißt Ausgabeschicht. Alle Schichten

dazwischen sind verborgene Schichten (Hidden Layer). Die Anzahl der verwendeten Schichten wird als Tiefe bezeichnet während die Breite die maximale Anzahl von Neuronen in einer Schicht beschreibt. Die Gewichte und der Bias sind die einzigen Parameter, die das Netzwerk während des Trainings verändern kann. Alle anderen, wie z.B. die Tiefe oder die Breite, sind Hyperparameter und müssen bei der Initialisierung des Netzwerkes festgelegt werden. In dieser Arbeit wurden ausschließlich Feedforward Netze eingesetzt. Das heißt, es kommt an keiner Stelle zu einer Rückführung des Signals.

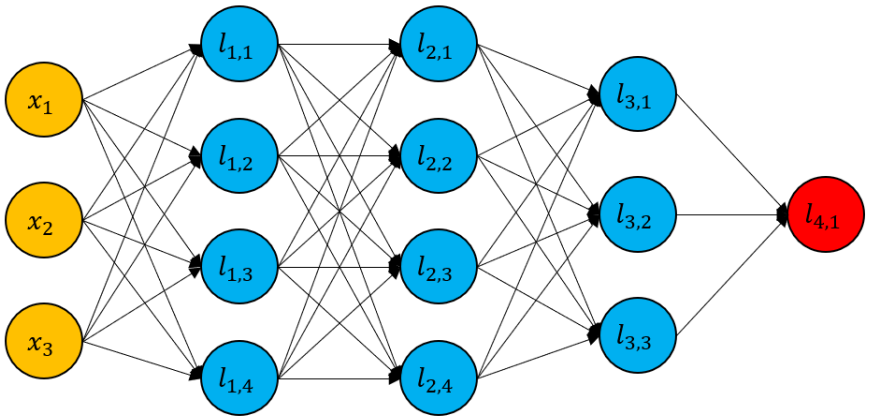


Abbildung 2.1: Darstellung eines neuronalen Netzes mit 3 Hidden Layer (blau) und einer Breite von 4. Die Eingabeschicht (grün) besitzt 3 Neuronen, während es ein Neuron in der Ausgabeschicht (rot) gibt. Ein Kreis entspricht dabei einem Neuron, welches eine mathematische Operation (Gleichung 2.1.1) repräsentiert.

Abbildung 2.1 zeigt die Struktur eines ANN. Die Ausgabe $l_{i,j}$ eines Neurons in der i-ten Schicht und an j-ter Stelle setzt sich zusammen aus: den Gewichten $\vec{\theta}_{i,j}$, dem Bias $b_{i,j}$ und der Aktivierungsfunktion f_i. Der Output des gesamten Netzwerks ergibt sich aus:

$$y(\vec{x}) = l_4(\vec{l_3}(\vec{l_2}(\vec{l_1}(\vec{x})))) \tag{2.1.2}$$

oder in ausgeschriebener Form

$$y(\vec{x}) = f_4(\vec{\theta}_4 \cdot \vec{f}_3(\theta_3^\top \cdot \vec{f}_2(\theta_2^\top \cdot \vec{f}_1(\theta_1^\top \cdot \vec{x} + \vec{b}_1) + \vec{b}_2) + \vec{b}_3) + b_4). \tag{2.1.3}$$

Die Gewichte der verborgenen Schichten l_{1-3} sind als Gewichtsmatrix θ_{1-3} dargestellt, deren j-te Spalte durch $\vec{\theta}_{1-3,j}$ besetzt wird. Da die Ausgabeschicht aus lediglich einem Neuron besteht, reduziert sich die Gewichtsmatrix auf einen Vektor $\vec{\theta}_4$.

Aktivierungsfunktion

Die Aktivierungsfunktion jeder Schicht (f_{1-4} in dem Netz von Abbildung 2.1) ist von zentraler Bedeutung für die Fähigkeit von ANNs, komplexe Aufgaben zu lösen.

Es ist wichtig, dass die Aktivierungsfunktion nicht linear ist, da sonst alle Schichten und Neuronen zu einem linearen Modell zusammengefasst werden können:

$$y(\vec{x}) = \theta_{\text{linear}}^{\top}\vec{x} + \vec{b}_{\text{linear}}. \tag{2.1.4}$$

Hierdurch wären komplexe Aufgaben nicht lösbar. Die Parameter $\theta_{\text{linear}}^{\top}$ und $\vec{b}_{\text{linear}}$ setzen sich in diesem Fall aus den Gewichtsmatrixen und Biasvektoren des Netzwerks zusammen. Die verwendeten Aktivierungsfunktionen sind in Abbildung 2.2 dargestellt.

Die Rectified Linear Unit (ReLU)(Abbildung 2.2a) ist definiert durch:

$$\text{ReLU}(x) = \max\{0, x\}. \tag{2.1.5}$$

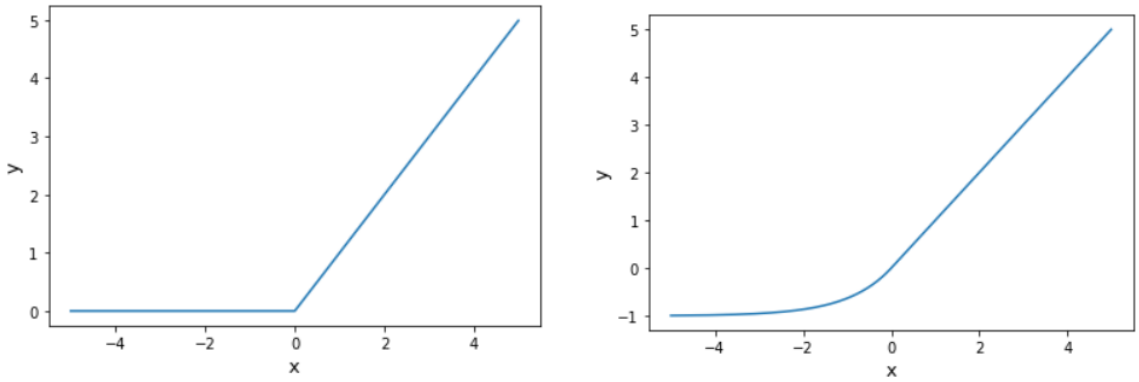
Sie ist ein verbreiteter, gut funktionierender Standard. Durch ihre stückweise Linearität ist die Ableitung im Vergleich zu anderen Aktivierungsfunktionen wie die der später vorgestellten Sigmoid nicht nur groß, sondern auch konstant für den aktiven Bereich der Funktion. Dies hat Vorteile bei dem Training der Netzwerke. Bei $x = 0$ existiert allerdings ein Punkt, an dem sie nicht differenzierbar ist, wodurch es zu Sprüngen während des Trainings kommen kann. Neben der ReLU wurde mit der Exponential Linear Unit (ELU) noch eine weitere Aktivierungsfunktion verwendet:

$$\text{ELU}(x) = \begin{cases} x & \text{wenn } x > 0 \\ \alpha(\exp(x) - 1) & \text{wenn } x \leq 0 \end{cases}. \tag{2.1.6}$$

Sie ist für $x > 0$ identisch mit der ReLU, geht aber für negative x asymptotisch gegen $-\alpha$ und ist an der Stelle $x = 0$ differenzierbar. Diese Eigenschaften führen zu einer höheren Generalisierung des Netzwerkes und zu schnellerem Lernen [22]. Durch die im Vergleich zur ReLU komplizierteren Form dauern die Berechnungen hingegen länger. Neben der ReLU und ELU fand auch noch bei den Ausgabeschichten die Sigmoid Aktivierungsfunktion, definiert als

$$\text{sigmoid(x)} = \frac{\exp x}{1 + \exp x}, \tag{2.1.7}$$

Verwendung.



(a) Rectified Linear Unit (b) Exponential Linear Unit ($\alpha = 1$)

Abbildung 2.2: Verwendete Aktivierungsfunktionen

2.2 Training von ANN

Der größte Unterschied zwischen ANN und klassischen Filtern ist die Berechnung der Gewichte jedes Neurons [23]. Diese werden während des Trainings von dem Netzwerk ohne Eingreifen des Entwicklers selbständig optimiert. Das Training von ANN unterteilt sich in zwei Gebiete: das Supervised und Unsupervised Learning. In dieser Arbeit wurde das Supervised Learning verwendet. Dabei stehen dem Netz neben dem Datensatz der Input Feature X auch die gewünschten Ausgaben Y, welche das Netz mit X berechnen soll, zur Verfügung. Während des Trainings wird der vom Netzwerk berechnete Output $\hat{y}$ mit dem wahren Wert y verglichen und basierend darauf werden die Gewichte und der Bias der einzelnen Neuronen verändert. Beim Unsupervised Learning versucht das Netzwerk die der Daten X zugrundeliegende Verteilung zu erlernen.
Die zur Verfügung stehenden Daten X mit dem gewünschten Output Y müssen in ein Trainingsdatensatz $(X_{\text{train}}, Y_{\text{train}})$ und ein Testdatensatz $(X_{\text{test}}, Y_{\text{test}})$ aufgeteilt werden. Das Training findet ausschließlich mit X_{train} statt und X_{test} dient zur Evaluierung. Das Training des Netzes hat zwei Ziele:

1. Reduktion des Trainingfehlers
2. Reduktion des Unterschieds zwischen Trainings- und Testfehler

Trainingsfehler und Testfehler sind dabei folgendermaßen definiert:

$$\text{Trainingsfehler} = \frac{1}{m_{\text{train}}} \sum_{i=1}^{m_{\text{train}}} (y_i - \hat{y}_i)^2 \quad \text{mit } y_i \in Y_{\text{train}} \tag{2.2.1}$$

und

$$\text{Testfehler} = \frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} (y_i - \hat{y}_i)^2 \quad \text{mit } y_i \in Y_{\text{test}}. \tag{2.2.2}$$

Die Größen m_{test} und m_{train} stellen die Anzahl der verfügbaren Trainings-/Testdaten (x_i, y_i) dar. Der Trainingsfehler wird auch Loss genannt und mittels einer Loss Funktion J (in 2.2.1 der mittlere quadratische Fehler) berechnet. Andere mathematische Funktionen, wie z. B. die mittlere absolute Differenz oder der mittlere prozentuale Fehler, können für J ebenso verwendet werden.
Das erste Ziel beschreibt die Optimierung der Gewichte an die gesehenen Daten in X_{train}, während das zweite Ziel für die Generalisierung (Performance für nicht trainierte Daten) des Netzes steht. Das Netzwerk darf sich, um beide Ziele zu erfüllen, nicht zu stark an X_{train} anpassen, da sonst der Fehler für X_{test} steigt. Dies entspricht einem Overfit. Beim Underfit hingegen ist der Trainingsfehler hoch, da das Netzwerk nicht die Möglichkeiten hat, sich X_{train} anzupassen. Die Netzwerkarchitekur und Länge des Trainings bestimmen maßgeblich die Tendenz für den Over- oder Underfit und müssen deshalb an die Aufgabe angepasst werden.

Der grundlegende Algorithmus hinter dem Training ist der Backpropagation Algorithmus. In einem Vorwärtsdurchlauf wird für einen Teil der Trainingsdaten die Ausgabe aller Neuronen, sowie der Output des gesamten Netzes $\hat{y}(\vec{\theta})$ in Abhängigkeit aller Gewichte $\vec{\theta}$ berechnet. Der Trainingsfehler $J(y, \hat{y}(\vec{\theta}))$ setzt sich dann aus dem gewünschten Output y, der Loss Funktion J und dem berechneten Output $\hat{y}(\vec{\theta})$ zusammen. Der folgende Rückwärtsdurchlauf dient zur Anpassung der Parameter. Zuerst berechnet der Algorithmus die Beiträge der Gewichte an der Loss Funktion mittels der Gradienten $\nabla_{\vec{\theta}} J(\vec{\theta})$. Dabei beginnt er bei der Ausgabeschicht und arbeitet sich, durch Anwenden der Kettenregel, stückweise zur Eingabeschicht durch das gesamte Netzwerk. Der letzte Schritt dient zur Aktualisierung der Gewichte mit den berechneten Gradienten mit dem Ziel, das globale Minimum der Loss Funktion zu finden. Die einfachste Art, die Gewichte anzupassen, stellt das Gradientenverfahren dar:

$$\vec{\theta} \longleftarrow \vec{\theta} - \eta \cdot \nabla_{\vec{\theta}} J(\vec{\theta}). \tag{2.2.3}$$

Der Hyperparameter η ist die Lernrate und bestimmt die Schrittlänge, mit welcher der Algorithmus die Gewichte in entgegengesetzter Richtung der Gradienten verändert. Der Rückwärtsdurchlauf ist somit abschlossen und neue Datenpunkte beginnen mit dem Forwärtsdurchlauf. Nach der Verwendung aller Daten im Trainingsdatensatz ist eine Epoche abschlossen. Für eine gute Anpassung der Parameter sind immer mehrere Epochen pro Training notwendig.

Das Gradientenverfahren hat den Nachteil, dass bei kleinen Gradienten die Optimierung der Gewichte sehr langsam stattfindet. Deshalb fand der Adam Optimizer [24] in dieser Arbeit Anwendung. Dieser benutzt nicht nur die Gradienten des aktuellen Schrittes, sondern berücksichtigt auch die vorangegangenen Schritte, um so

schneller und direkter das Minimum zu finden [18].

2.3 Netzwerkarten

Es gibt viele verschiedene Arten von neuronalen Netzen. Sie unterscheiden sich dabei nicht nur in ihrer unterschiedlichen Anzahl an Schichten oder benutzen Aktivierungsfunktionen, sondern auch in der Struktur, wie die einzelnen Neuronen ihr Signal weitergeben. Die in dieser Bachelorarbeit genutzten Netzwerkarten sind Multilayer Perceptron (MLP)-Netze, Convolutional Neural Networks (CNNs) oder Kombinationen aus Beiden. Für die Realisierung der Netze sowie dem Training wurden die Bibliotheken von Tensorflow [25] und Keras [26] in Python verwendet.

Multi-Layer Perceptron Netzwerk

Ein MLP ist von der Struktur aufgebaut wie das Netzwerk in Abbildung 2.1 mit lediglich mehr und breiteren Schichten. Es stellt die einfachste Struktur dar, in der die Neuronen verbunden werden. Die Ausgabe eines Neurons wird an alle Neuronen der nachfolgenden Schicht weitergegeben, sodass man von einem vollständig verbundenen Netzwerk spricht. Die entscheidenden Parameter für MLP sind die Anzahl der Schichten, die Neuronen pro Schicht und die Aktivierungsfunktionen jeder Schicht.

Convolutional Neural Networks

CNNs sind besonders erfolgreich bei der Erkennung von Strukturen, wie z.B. der Objekterkennung in Bildern oder von Pulsen in einer Zeitserie, weshalb sie für diese Arbeit ausgewählt wurden. Sie besitzen mindestens eine Convolutional Layer, dessen Funktion an der mathematischen Operation der Faltung angelehnt ist.
Eine Serie von Inputdaten, deren Länge dem Receptive Field entspricht, wird von einem Kernel mit einem Fenster, realisiert durch ein Neuron, abgefahren. Die Anzahl an Inputs, die das Neuron verarbeitet, bestimmt das Fenster des Kernels, welches kleiner als das Receptive Field sein muss. Während des stückweisen Abfahrens der Inputdaten verarbeitet das Neuron gemäß Gleichung 2.1.1, mit gelernten Gewichten und einem Bias, die im Fenster liegenden Daten zu einem eindimensionalen Output. Abbildung 2.3 visualisiert diese Verfahren. Neben der Fenstergröße des Kernels ist ein weiterer wichtiger Parameter die Anzahl der Feature Maps in dem Convolutional Layer. Die Feature Maps geben an, wie viele Kernel die Input Serie parallel durchgehen. Diese besitzen dabei unterschiedliche Gewichte und können sich so auf unterschiedliche Muster spezialisieren. Die nachfolgende Schicht erhält den Output der verschiedenen Kernels aus den Feature Maps.
Nach oder zwischen den Convolutional Layern folgen meistens noch Schichten zur

Reduktion der Datenmenge, den Pooling Layern, gefolgt von Schichten mit vollständig verbunden Neuronen, die den Output der Convolutional Layer verarbeiten und die Ausgabe des gesamten Netzwerks berechnen. Der Vorteil von CNNs liegt darin, insgesamt weniger Parameter zu benötigen und viele Parameter mehrmals verwenden zu können. Der Kernel fährt das komplette Receptive Field mit den gleichen Parametern ab, während im Vergleich dazu ein MLP-Netz für jeden Zeitschritt im Receptive Field ein Neuron mit eigenen Gewichten zugeteilt hat.
Entscheidende Hyperparameter bei der Initialisierung sind dabei die Kernelgröße, die Anzahl an Feature Maps, die Aktivierungsfunktionen, die Anzahl der Convolutional Layer, die Art und Häufigkeit von Pooling Layern sowie die Anzahl der vollständig verbundenen Neuronen.

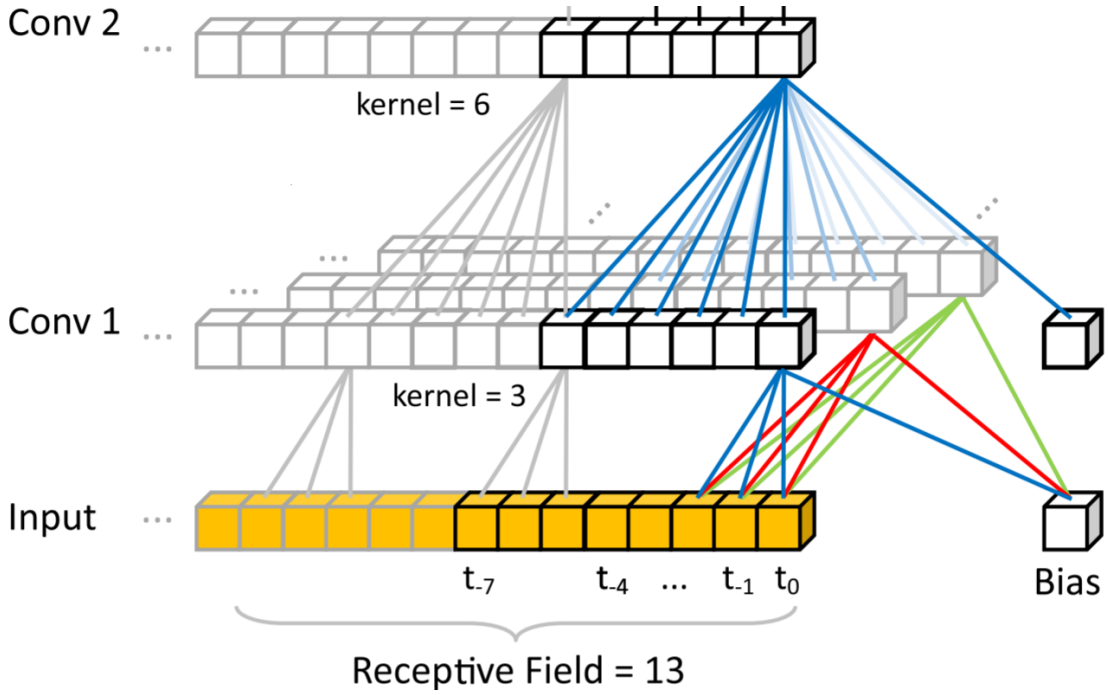


Abbildung 2.3: Die Abbildung illustriert die Funktionsweise zweier hintereinander platzierten Convolutional Layer. Der Kernel von Conv 1 wird dargestellt als eine Reihe von identischen Neuronen (weiße Würfel), bei denen jedes Neuron nur ein kleines Fenster (Fenstergröße des Kernels) des Receptive Fields sieht. Drei Feature Maps in Conv 1 sind durch die parallelen Reihen der Neuronen dargestellt, deren unterschiedliche Gewichte mit 3 Farben gekennzeichnet sind. Durch die Feature Maps in Conv 1 erhält Conv 2 einen zweidimensionalen Input und der Kernel wird zu einem 6×3 Fenster, welches, die Inputsequenz abfährt. Bildausschnitt aus [3]

Kapitel 3: Simulation und Datenstruktur

Für das Supervised Learning wird ein umfangreicher Datensatz zu dem ADC-Output des Tile-Kalorimeters mit der wirklichen Energie der detektieren Teilchen benötigt. Echte Daten aus älteren Läufen des LHC haben das Problem, dass die den Hits zugewiesene Energie aus der Berechnung mit dem FIR-Filter stammt. Das trainierte Netz wäre somit von der Zuverlässigkeit des FIR-Filters abhängig. Auch werden bei detektierten Hits nur 5 oder 7 BC abgespeichert, wodurch die Netzwerkarchitektur eingeschränkt wäre. Eine Alternative stellen Zero Bias Daten mit hinzugefügten Pulsen bekannter Energie dar.
Die Daten wurden mithilfe der Simulation ToyMC generiert. Diese entwickelte A. C. Daniels im Rahmen seiner Doktorarbeit[4], um die Parameter des FIR-Filters zu optimieren. ToyMC simuliert den vollständigen ADC-Output der TT in den Kalorimetern über einen kompletten Umlauf im LHC für beliebige $|\eta|$. Berücksichtigt wird dabei das Füllschema des LHCs, damit Effekte wie in-time und out-of-time Pile-Up sowie verschiedene Rauschquellen genau rekonstruiert werden. Der Vorteil einer Simulation des vollständigen ADC-Outputs besteht darin, durch längere Eingangssequenzen flexibler in der Architektur der Netzwerke zu sein. Neben dem digitalisierten Signal eines TT kann auf die Energie der im Output platzierten Pulse zurückgegriffen werden, womit sich das Supervised Learning für die neuronalen Netze realisieren lässt. Dieses Kapitel behandelt die Struktur der verwendeten Daten und geht auf die Implementierung in der Simulation ein.

3.1 Rauschquellen

Die größten Rauschquellen im Tile-Kalorimeter sind das thermische Rauschen und der Pile-Up. Die Auswirkung auf die Leistungsfähigkeit des Triggers liegen in der schlechteren Effizienz bei der Detektion von niedrigenergetischen Hits und der reduzierten Auflösung für die Energiebestimmung bei getriggerten Hits. Dies ist der Grund für die Einführung eines Noisecuts im FIR-Filter und den trainierten Netz-

werken. Ohne Noisecut steigt die Rate von falsch detektierten Ereignissen mit niedriger Energie. Das thermische Rauschen hat ein weißes Spektrum. Es hängt von der Detektor Charakteristik und der elektronischen Verarbeitung des Signals, wie z. B. den Verstärkern oder Kabeln, ab. Für den untersuchten TT in dieser Arbeit bei $\eta = -0.25$ wird in den ToyMC Tool ein gaussförmiges thermisches Rauschen mit einer Standardabweichung von $\sigma_{\text{Thermal}} = 0.357\,\text{GeV}$ verwendet. Im Vergleich zu TT bei höheren $|\eta|$ ist das thermische Rauschen, für den betrachteten TT, deutlich höher. Das Rauschen ist abhängig von der deponierten Energie. Da jedoch im Trigger die transversale Energie bestimmt wird, skaliert das thermische Rauschen mit $\sin(\theta)$ [4]. Die Simulation berücksichtigt das thermische Rauschen, indem für jedes BC ein zufälliger Wert gemäß der Normalverteilung für den simulierten TT zum Pedestal addiert wird.

Eine weitere Rauschquelle entsteht durch das Digitalisieren des analogen Signals auf 10 Bit. Für den ADC-Output mit einem least significant Bit (LSB) von 0.25 GeV beträgt der mittlere Fehler gemäß der Regel $\frac{\text{LSB}}{\sqrt{12}}$ [27] ungefähr 72 MeV.

Der Pile-Up entsteht dadurch, dass es in jedem BC zu Ereignissen mit hohem Wirkungsquerschnitt kommt. Die Größe $\langle\mu\rangle$ beschreibt, wie viele Proton-Proton-Interaktionen es im Mittel pro BC gibt und stellt somit ein Maß für die Höhe des Pile-Ups dar. Da die meisten dieser Interaktionen uninteressant sind, es aber trotzdem zu einer Energiedeposition in den Kalorimentern kommt, tragen sie zum Rauschen bei. Es wird unterschieden zwischen dem in-time Pile-Up und out-of-time Pile-Up.

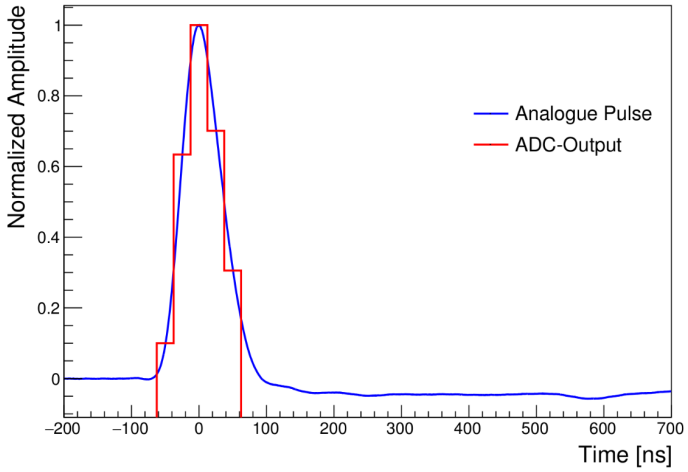


Abbildung 3.1: Puls in analoger und digitalisierter Form in dem Kalorimeterbereich TileLB aus Oszilloskop-Messungen[14]. Nach dem positiven Ausschlag folgt der Undershoot mit leicht negativer Amplitude. Bei Pulsen im Tile-Kalorimeter ist der Undershoot deutlich geringer im Vergleich zum elektromagnetischen Kalorimeter.

Out-Of-Time Pile-Up

Die Untergrundprozesse erzeugen genauso wie hochenergetische Hits einen Puls im analogen Output des Kalorimeters, wie er in Abbildung 3.1 zu sehen ist. Da die gesamte Pulslänge inklusive Undershoot sich insgesamt über mehrere BCs erstreckt, beeinflusst ein Pile-Up Event bis zu 26 nachfolgende BCs. Der Zeitnullpunkt wird als das Maximum des Pulses definiert. Aufgrund der Pulsbreite kann sich der Einfluss eines Pile-Up Events auch auf bis zu 3 vorherige BCs erstrecken. Die Pile-Up Events erzeugen eine ständige Überlagerung von Untergrund-Pulsen und hochenergetischen Hits. Der out-of-time Pile-Up kann interessante Ereignisse auf verschiedene Weise beeinflussen: Es entsteht eine konstruktive Interferenz zwischen Pile-Up Event und hochenergetischem Hit, wenn der Pile-Up nahe der Pulsamplitude auftritt. Die Energie des Hits wird überschätzt. Durch den Undershoot der Pulse kann aber auch eine destruktive Interferenz entstehen, wodurch es zu einer Unterschätzung der Energie des Hits kommt.

In-Time Pile-Up

Der in-time Pile-Up entsteht, anders als der out-of-time Pile-Up nicht durch Pile-Up Events aus vorherigen BCs, sondern von Ereignissen im selben BC. Die Pulsamplituden des interessanten hochenergetischen Hits und dem in-time Pile-Up überlagern sich und sind nicht zu unterscheiden. Die konstruktive Interferenz führt zu einer Überschätzung der Energie des Hits. Der Pile-Up wird in der Simulation berücksichtigt, indem in jedes BC ein Pulse mit zufälliger Amplitude eingefügt wird. Die Abbildung 3.2 zeigt die auftretenden Pile-Up Ereignisse als Funktion ihrer Amplitude in ADC Counts. Die maximale Pile-Up Amplitude beträgt 1.8 ADC und damit 0.45 GeV. Verglichen mit dem thermischen Rauschen mit $\sigma_{\text{Thermal}} = 0.357\,\text{GeV}$ ist der Pile-Up mit einer mittleren Amplitude von 0.025 GeV klein. Somit ist die dominierende Rauschquelle für die betrachteten Daten rein gauss'sches Rauschen. Diese Werte gelten für ein simuliertes $\langle\mu\rangle$ von 40.

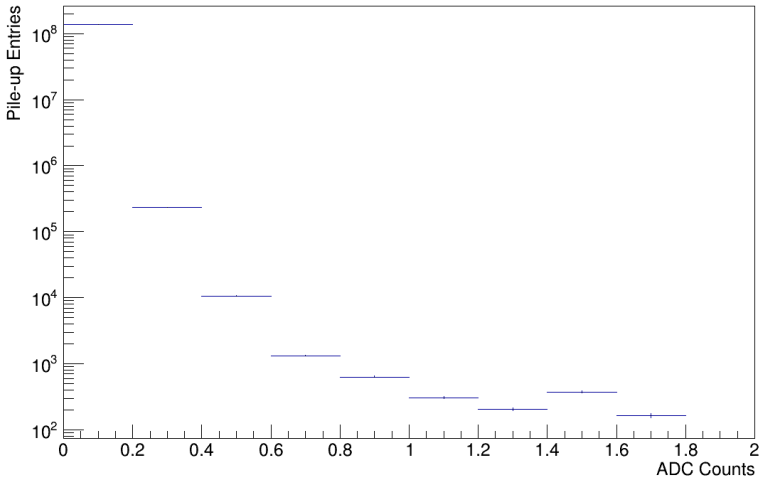


Abbildung 3.2: Amplituden der Pile-Up Ereignisse bei einer Simulation von $1.4 \cdot 10^8$ BCs bei $\langle \mu \rangle = 40$

3.2 Füllschema

Die Bunches im LHC sind nicht gleichmäßg mit Teilchen gefüllt, sondern gemäß eines Füllschemas. Für die Simulation wurde ein 4-4-2 Schema verwendet, wie es in Abbildung 3.3 zu sehen ist. Es werde jeweils zwei oder vier Gruppen aus 72 gefüllten Bunches in sogenannte Batches zusammengefasst. Zwischen den Gruppen liegt die Short Gap mit einer Länge von 8 nicht gefüllten Bunches. Nach einem Batch folgt die Long Gap von 36 nicht gefüllten Bunches bis zum nächsten Batch. Dies wird in dem Muster 4-4-2 wiederholt, bis die Abort Gap am Ende eines LHC-Umlaufs das Füllschema abschließt. Dieses Schema resultiert aus den Eigenschaften des Vorbeschleunigersystems.
Die Lücken im Füllschema, in denen keine Proton-Proton-Interaktionen auftreten, haben Auswirkungen auf den ADC-Output aufgrund des out-of-time Pile-Ups. Es entsteht eine BC-abhängige Verschiebung des Pedestals. Abbildung 3.4 zeigt den gemittelten ADC-Output über $6.5 \cdot 10^4$ LHC-Umläufe. Das Pedestal ist überall dort größer als das mittlere Pedestal von 31.68, wo die Bunches gefüllt sind. Ursache dafür ist der Puls (Abbildung 3.1), welcher nicht auf Null normiert ist, wie es in anderen Detektorbereichen der Fall ist. Jeweils zu Beginn einer Gruppe ist ein deutlicher Anstieg, relativ zum Rest der Gruppe, sichtbar. Der ADC-Output setzt sich aus einer Überlagerung von Hits, Undershoot von vorherigen Hits, in-time Pile-Up, out-of-time Pile-Up und thermischem Rauschen zusammen. Aufgrund der nicht gefüllten BCs in der Short Gap fehlen am Anfang einer Gruppe der Undershoot von Pile-Up Events und Hits aus vorherigen BCs. Daher steigt das Pedestal dort stark an. Je länger die Gap vor einer Gruppe ist, desto höher fällt der Anstieg aus. Innerhalb einer Gruppe führen die Überlagerungen von Hits und Pile-Up Events aus vorherigen

BCs zunächst zu einer Verringerung des Pedestals bis dieser schlussendlich konstant wird.

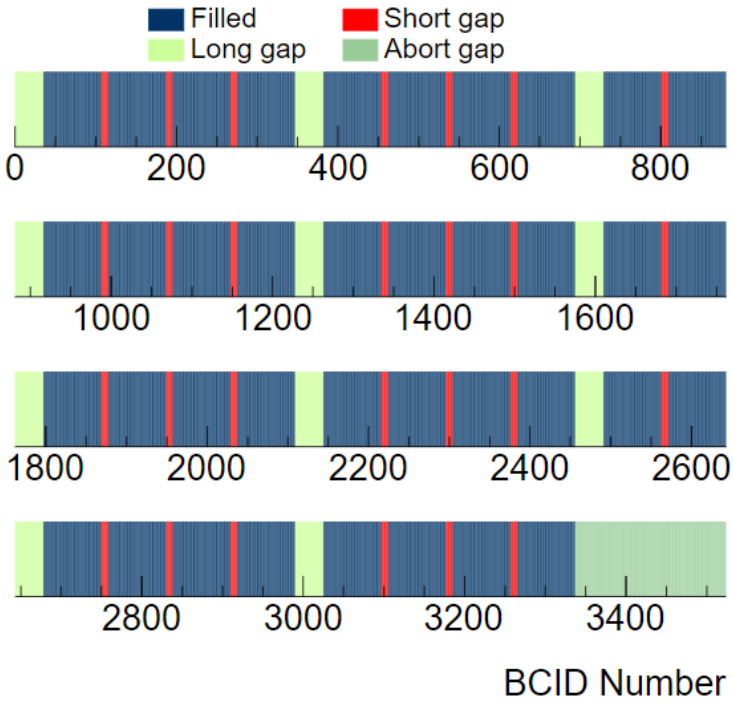


Abbildung 3.3: Verwendetes 4-4-2 Füllschema des LHC[4]

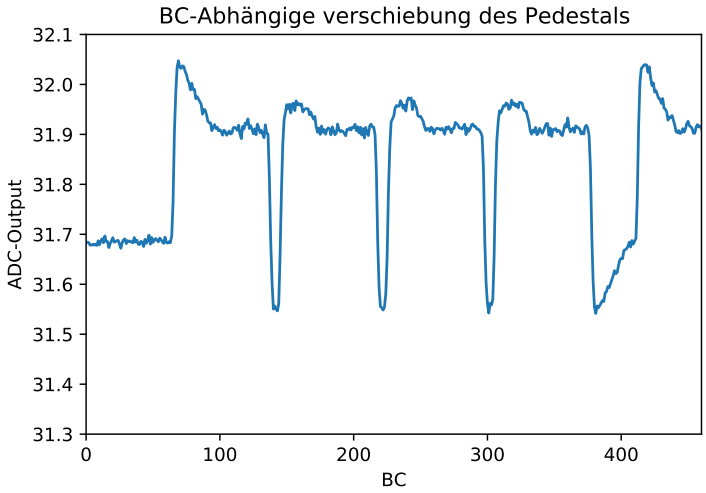


Abbildung 3.4: Auswirkungen des Füllschemas und Pile-Ups auf das mittlere Pedestals

3.3 Implementierung von Hits

Der Pile-Up und die Hits, welche durch den Filter erkannt werden sollen, sind in der ToyMC Simulation getrennt voneinander. Somit werden neben den Pile-Up Pulsen auch Pulse implementiert, deren Amplitude aus einer Verteilung für die Hits gezogen wird. Die Verteilung der Amplituden stammt aus experimentellen Messungen und wurde für die Arbeit mit den Netzen modifiziert, sodass höherenergetische Hits öfters auftreten. Für das Training mit den Netzwerken ist eine hohe Anzahl von Hits mit höheren Energien notwendig. Die Modifikation verkürzt dabei die Rechenzeit. Beide Verteilungen sind im Anhang zu finden (Abbildung A.1).

In Abbildung 3.5 ist der ToyMC Output für einen Ausschnitt von 100 aufeinander folgenden BCs dargestellt. Die blaue Linie zeigt den ADC-Output, welcher als Input für die Modelle dient. Weiter zu sehen sind die verschiedenen, zuvor beschriebenen, Beiträge, aus welchen sich der ADC-Output zusammensetzt. Die orangene Linie beschreibt die implementierten Hits, die rote Linie ist der Pile-Up und das thermische Rauschen wird durch die grünen Balken dargestellt. Der Puls bei dem BC 2403 besteht aus einer Überlagerung von zwei direkt hintereinander auftretenden Pulsen mit Energien von 0.96 GeV und 2.32 GeV. Alle anderen implementierten Ereignisse in dem Ausschnitt liegen in einem Bereich von ca. $0.1 - 0.5$ GeV. Aufgrund des starken thermischen Rauschens sind diese Ereignisse im ADC-Output nicht erkennbar. Der Pile-Up hat im Vergleich zum thermischen Rauschen einen vernachlässigbaren Einfluss auf den ADC-Output.

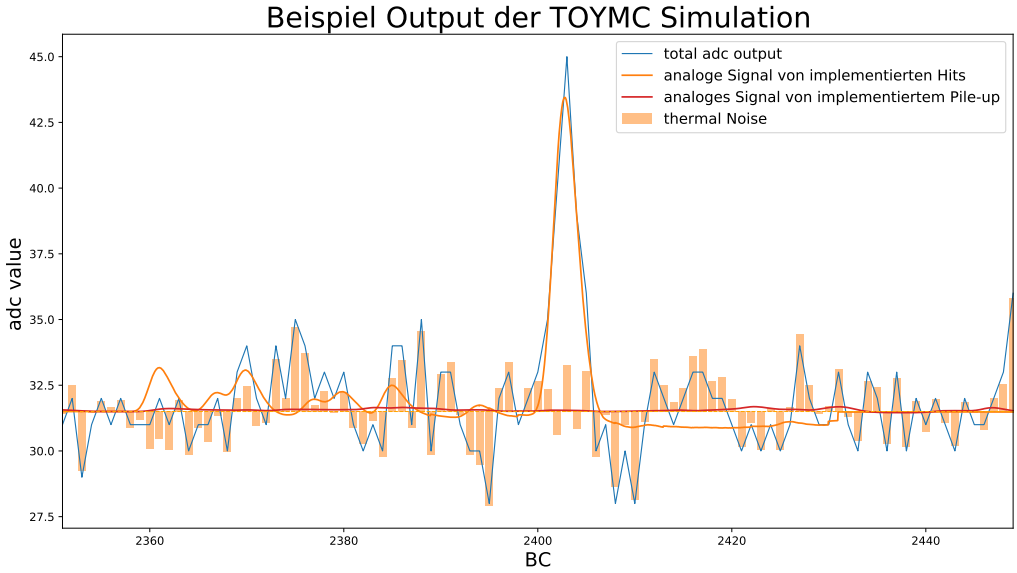


Abbildung 3.5: Output der ToyMC Simulation mit den verschiedenen Rauschquellen und implementierten Hits

Kapitel 4: Initialisierung und Training der Netzwerke

In diesem Kapitel wird die Implementierung der in Kapitel 2 beschriebenen Netzwerkarten mit den in Kapitel 3 eingegangenen Daten diskutiert. Die grundlegende Aufgabe, die durch ein trainiertes Modell gelöst werden soll lautet: Hits im ADC-Output identifizieren und ihre Energie genau berechnen. Ein gutes Netz zeichnet sich dadurch aus, dass Hits auch bei niedrigen Energien dem richtigen BC zugeordnet werden und die Auflösung der Energiebestimmung hoch ist. Das Netz erhält als Input eine Sequenz des ADC-Outputs, den Featuren, und wird darauf trainiert, auf einen bestimmten Eintrag in der Sequenz sensitiv zu sein. Die Ausgabe des Netzwerks entspricht der Energie des Pulses, dessen Amplitude sich an dem sensitiven BC befindet. Gibt das Netzwerk eine Energie von Null aus, bedeutet dies, dass es keinen Puls an der entsprechenden Stelle identifiziert hat. Es ist dadurch möglich, eine ausgegebene Energie eindeutig einem BC zuzuordnen. Abbildung 4.1 veranschaulicht das Vorgehen.

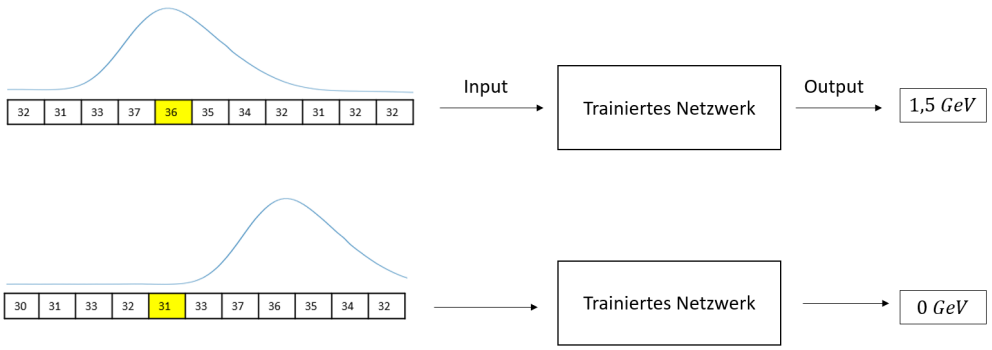


Abbildung 4.1: Es sind zwei Beispiele dargestellt, wie die grundlegende Struktur der Netzwerke aufgebaut ist. Die dargestellte Sequenz aus dem ADC-Output hat eine Länge von 11 BC und stellt die Feature dar. Der darüber aufgezeichnete Puls befindet sich an dieser Stelle im ADC-Output. Die sensitive Position in der Sequenz ist gelb markiert und entspricht dem 5. Eintrag. Befindet sich dort die maximale Amplitude eines Pulses, soll das Modell die Energie berechnen und ausgeben (oberes Beispiel). Ist dies nicht der Falle, gibt das Netzwerk idealerweise eine Energie von 0 aus (unteres Beispiel).

Die implementierten Hits können Energien zwischen 0 und 10 GeV haben. Das Training zielt darauf ab, alle Hits unabhängig ihrer Energie zu finden. Da dies bei niedrigen Energien, wegen des thermal Noise, aber unmöglich ist, wird auf die Ausgabe der Netze ein Noisecut angewendet. Die Bestimmung des Noisecuts wird in 5.2 thematisiert. Zu diesem Zeitpunkt wird vorerst kein Noisecut verwendet.

4.1 Verwendete Netzwerke

Für die spezielle Architektur der Netze wurden mehrere Ansätze realisiert und deren Performance später verglichen. Im folgenden Teil werden die unterschiedlichen Netze vorgestellt.

Trigger Netzwerk

Die Architektur des Trigger Netzwerks ist inspiriert durch [3]. Das Modell setzt sich aus zwei Teilen zusammen: dem Trigger und dem E_cal Netzwerk. Der Trigger ist dafür zuständig, in der Inputsequenz Hits zu erkennen. Seine Ausgabe liegt zwischen Null und Eins. Eine Eins bedeutet, dass die Wahrscheinlichkeit für einen Hit an der sensitiven Position hoch ist. Bei einer geringen Wahrscheinlichkeit liegt der Output nahe Null. Das E_cal Netzwerk verarbeitet die Ausgabe des Triggers zusammen mit der Sequenz, um daraus die Energie eines möglichen Pulses zu bestimmen. Die Verwendung der Informationen des Triggers unterliegen keinen Vorgaben und wird während des Trainings vom Netz selbstständig festgelegt. Der Trigger und auch das E_cal Netzwerk bestehen aus CNNs mit zwei Convolutional Layern. Das Training findet zuerst getrennt statt, indem der Trigger vortrainiert wird. Erst danach optimiert sich das gesamte Modell aus Trigger und E_cal Netzwerk. Durch diesen Ansatz erhofft man sich eine bessere Leistungsfähigkeit, da die Detektion und Energieberechnung von Hits durch zwei separate Netze durchgeführt wird. In Abbildung 4.2a ist die Struktur des Trigger Netzwerks dargestellt. Die Performance des vortrainierten Triggers ist im Anhang unter B.1 zu finden.

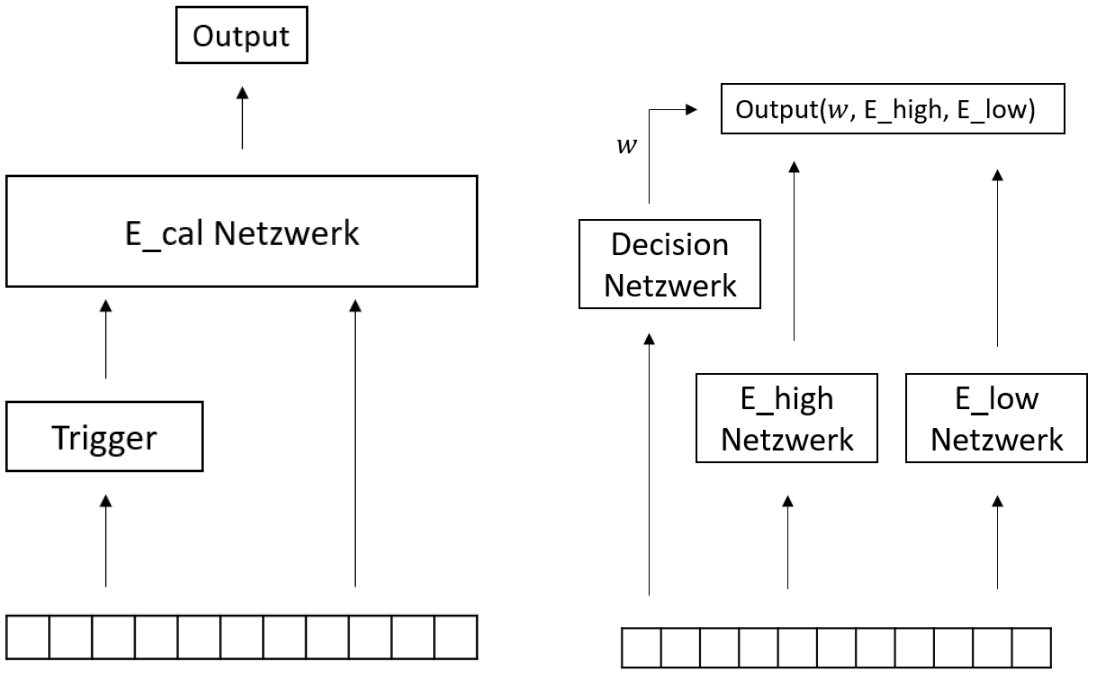
Combinational Netzwerk

Wie das Trigger Netzwerk, so besteht auch das Combinational Netzwerk aus vortrainierten Modellen. Aber anstatt eines Triggers verwendet das Combinational Netzwerk drei Subnetzwerke, welche verschiedene Aufgaben erfüllen. Das Erste heißt E_low und ist darauf spezialisiert, die Energie von niedrigenergetischen Hits ($<$ 1.5 GeV) zu berechnen und vom Hintergrund zu unterscheiden. Beim zweiten Subnetzwerk, dem E_high, findet die Energiebestimmung für höherenergetische Hits

($> 1.5\,\mathrm{GeV}$) statt. Die Entscheidung welches Subnetzwerk für die Ausgabe verantwortlich ist, trifft das Decision Netzwerk. Dessen Output w liegt zwischen 0 und 1, wobei 0 für $E_T < 1.5\,\mathrm{GeV}$ bzw. 1 für $E_T > 1.5\,\mathrm{GeV}$ steht. Der finale Output entsteht aus einer gewichteten Summe aus dem Output von E_low, E_high und den Gewichten w:

$$\mathrm{Output}(w, \mathrm{E\_high}, \mathrm{E\_low}) = \mathrm{E\_high} * w + \mathrm{E\_low} * (1 - w). \tag{4.1.1}$$

In Abbildung 4.2b ist diese Struktur graphisch dargestellt. E_high und E_low werden durch MLP-Netze realisiert, während das Decision Netz aus einem CNN mit zwei Convolutional Layern aufgebaut ist. Durch die Aufteilung des Energiebereichs während des Trainings auf zwei separate Netze, wird sich eine bessere Detektion der niedrigenergetischen Hits erhofft, ohne die hochenergetischen zu vernachlässigen. Die Performance der drei Subnetzwerke befindet sich im Anhang unter B.

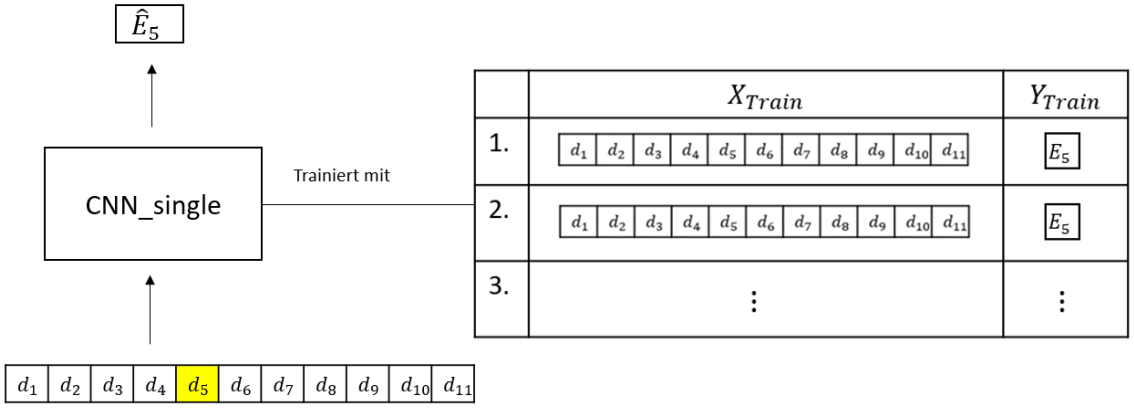


(a) Trigger Netzwerk (b) Combinational Netzwerk

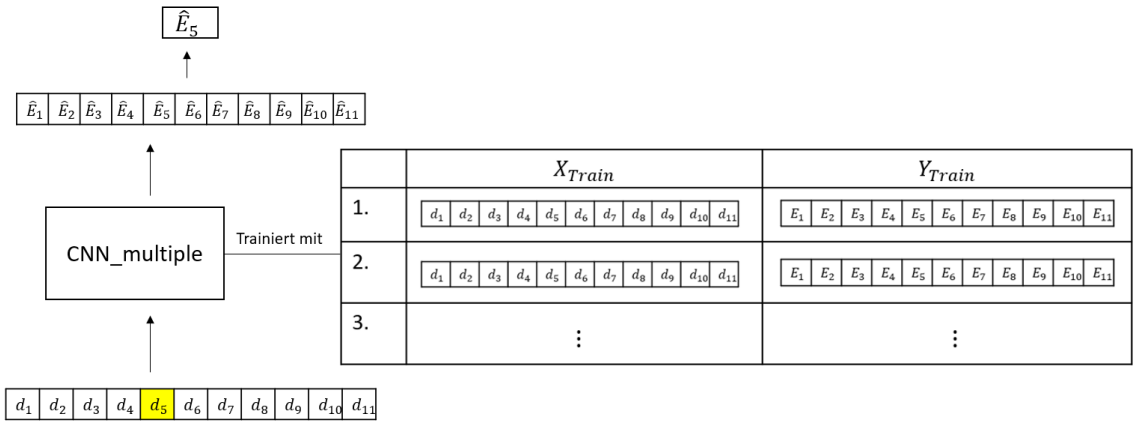
Abbildung 4.2: Datenfluss des verwendeten Trigger und Combinational Netzwerks. (a) : Der vortrainierte Trigger verarbeite zuerst die Sequenz und das E_cal Netzwerk berechnet die Energie mit der Sequenz und dem Output des Triggers. Der leer Array symbolisiert eine beliebige Sequenz des ADC-Outputs analog zu Abbildung 4.1
(b) : Das E_low und E_high Netzwerk berechnen ihren Output basierend auf der Input Sequenz. Das Decision Netzwerk bestimmt mit seinem Output w die Gewichtung der beiden Netzwerke. Der finale Output wird über Gleichung 4.1.1 berechnet.

CNN_single und CNN_multiple

CNN_single und CNN_multiple sind Netzwerke mit einer CNN-Architektur. Der einzige Unterschied zwischen beiden ist die Art und Weise, wie die Trainingsdaten verwendet werden. Die Abbildung 4.3 zeigt die Unterschiede zwischen CNN_single und CNN_multiple.



(a) CNN_single



(b) CNN_multiple

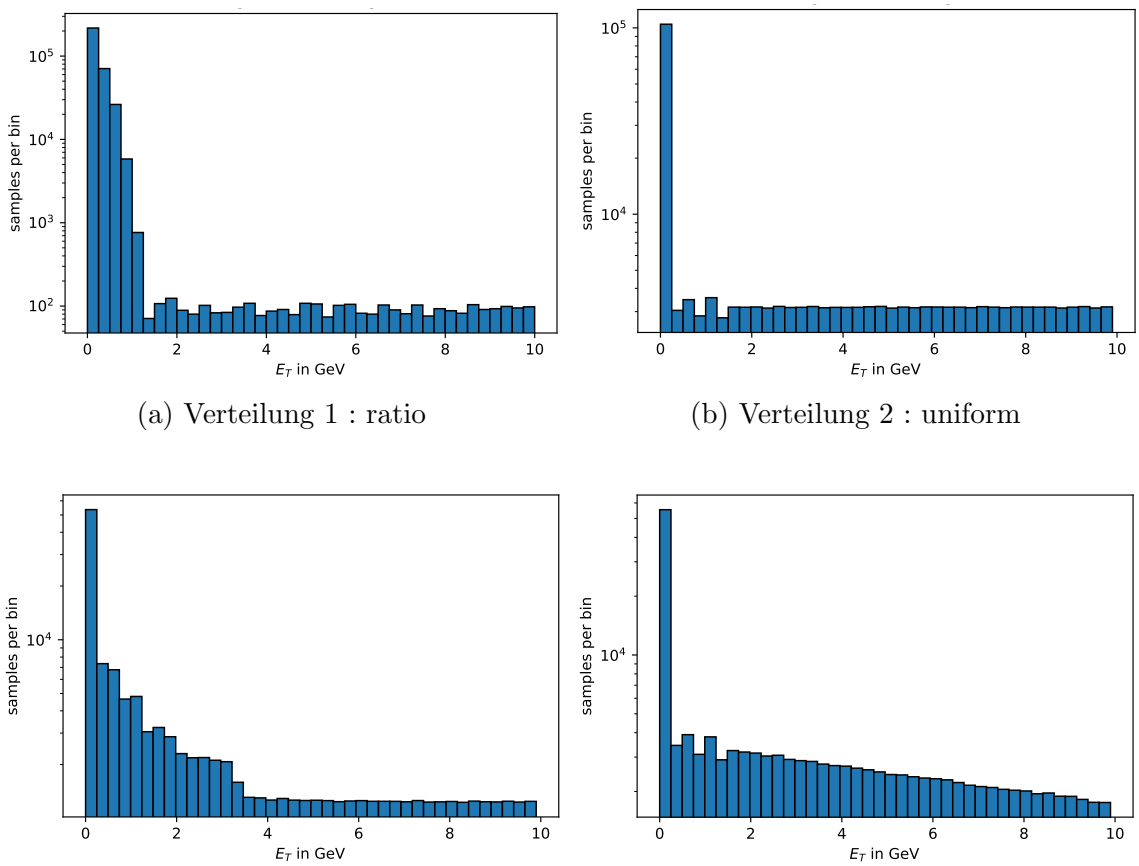
Abbildung 4.3: Unterschied zwischen den Netzwerkstrukturen CNN_single und CNN_multiple.
(a) : Das Netzwerk bestimmt die Energie $\hat{E}_5$ des 5. Eintrags in der Inputsequenz bestehend aus den ADC-Werten d_i. Trainiert wird es mit Sequenzen und der wahren Energie der 5. Stelle E_5.
(b) : Das CNN_multiple ist darauf designt, die Energie von jeder Stelle der Inputsequenz zu bestimmen. Die finale Ausgabe des Netzwerks bildet jedoch nur die Energie des 5. Eintrags analog wie beim CNN_single. Der entscheidende Unterschied ist jedoch, dass das Training nicht nur mit E_5 sondern, wie dargestellt, mit allen wahren Energien durchgeführt wird. Auf diese Weise erhält das Netzwerk während des Trainings deutlich mehr Informationen.

Wie schon vorher beschrieben und in Abbildung 4.1 dargestellt, wird für eine

Sequenz aus ADC-Werten nur die Energie der 5. Stelle vorhergesagt. Die Trainingsdaten enthalten dementsprechend Paare aus den Sequenzen und deren wahren Energien an der 5. Stelle. Dieses Vorgehen wird von allen zuvor beschriebenen erstellten Netzwerken sowie den CNN_single umgesetzt. Ein Aspekt, der dabei nicht beachtet wird ist, dass alle Informationen über Energiedepositionen an anderen Positionen in der Sequenz unberücksichtigt bleiben. Die Idee des CNN_multiple ist es, diese zusätzlichen Informationen über die Sequenz mit zu benutzen. Die Eigenschaft, der Equivarianz gegenüber Translation [21], von CNNs wird dabei ausgenutzt. Sie besagt, dass ein in der Input Sequenz verschobener Puls ein um den selben Betrag verschobenen Output erzeugt. Das CNN_multiple hat somit die Möglichkeit, auch von Pulsen zu lernen, deren Amplitude nicht an der 5. Stelle der Input Sequenz liegt.

4.2 Trainingsdaten

Die Trainingsdaten sind neben der Architekur der Netze ein zentraler Bestandteil für die Arbeit mit ANNs. Für die Simulation der Umläufe des LHC wurde das ToyMC verwendet. Aufgrund des seltenen Auftretens von hochenergetischen Hits in der Zeitserie ist ein Training mit den kompletten Daten aus mehreren Umläufen nicht möglich. Andererseits entsteht ein Modell, indem nur niedrigenergetische Hits aufgrund ihres häufigen Auftretens (mittlere Energie der implementierten Hits liegt bei 0.29 GeV siehe Abbildung A.1a) erkannt werden und die Anpassung an hochenergetische Hits nicht gelingt. Um dem entgegenzuwirken, wurde die Energieverteilung der Hits verändert und die Sequenzen mit Hits aus den Umläufen aussortiert. Zusätzlich fand noch eine Auswahl von Sequenzen ohne Hit ($\frac{1}{15}$ des Trainingsets) und mit verschobenen Hit($\frac{1}{3}$ des Trainingsets) statt. Um die Auswirkungen der Energieverteilung der Trainingsdaten zu studieren, wurden 4 verschiedene Trainingssets erstellt. Abbildung 4.4 zeigt die Energieverteilung der verwendeten Trainingsdaten. Die Anzahl der verwendeten Sequenzen pro Trainingssets ist im Anhang B.1 zu finden.



(a) Verteilung 1 : ratio

(b) Verteilung 2 : uniform

(c) Verteilung 3 : exp 1

(d) Verteilung 4 : exp 2

Abbildung 4.4: Die Abbildung zeigt die verschiedenen benutzten Energieverteilungen der Trainingsets aus ausgewählten Sequenzen.
(a) : Bei der ratio Verteilung sind die Hits aus vielen Umläufe zufällig ausgewählt. Dadurch bildet sich die große Anzahl niedrigenergetischer Sequenzen ($E_T < 1.5\,\text{GeV}$). Die unifrom verteilten hochenergetischen Hits entstehen aus der Modifizierung der zugrundeliegenden Energievertielung in der Simulation. Die Trainingsdaten bilden dadurch die tatsächliche Verteilung der Daten am genauesten ab, werden aber dominiert von Energien kleiner 1.5 GeV.
(b) : Die zweite Verteilung verwendet uniform verteilte Hitenergien mit zusätzlichen Daten ohne Hit, wodurch alle Energien größer Null eine gleiche Gewichtung besitzen.
(c) & (d) : Verteilung 3 und 4 stellen den Mittelweg zu den Extremen in Verteilung 1 und 2 dar. Die Energien der Trainingsdaten wurden dafür möglichst genau an die Funktion $f(x) = e^{-a \cdot x} + b$ mit $a = 1$ und $a = 0.1$ für Verteilung 3 bzw. 4 angepasst. Die ausgewählte Form soll kleine Energien stärker berücksichtigen, da diese auch öfters auftreten, aber gleichzeitig hohe Energien nicht vernachlässigen.

4.3 Training und Hypertuning

Vor dem Training der Netzwerke mit einem Trainingsset müssen die Hyperparameter der verschiedenen Modelle festgelegt werden. Die dafür verwendete Methode heißt Hypertuning. Der benutzte Algorithmus [28] initialisiert dabei das Netzwerk mit vielen unterschiedlichen Sets von Hyperparametern, trainiert und evaluiert es. Die Evaluation findet mit 30 % der Trainingsdaten statt, dem Validation Set. Es wird nicht für das Training verwendet, sondern nur zum Testen der unterschiedlich initialisierten Modelle. Das finale Netzwerk erhält schließlich das Set an Hyperparametern, welches den geringsten Validierungsfehler[1] hatte. Der Vorteil des Hypertunings ist, dass die Hyperparameter, wie z. B. die Anzahl an Neuronen, auf eine optimale Generalisierung optimiert werden. Das in Abschnitt 2.2 beschriebene Over-/Underfitting wird somit verhindert. Nach dem Hypertuning fand das finale Training mit einem der vier erzeugten Trainingsdatensätze statt. Um dabei einen Overfit durch zu langes Training zu verhindern, wurde ein Early Stopping verwendet. Der minimale Validierungsfehler pro Epoche bestimmt dabei die Anzahl der verwendeten Epochen. Folgende Kombinationen aus den erstellten Netzen (Abschnitt 4.1) und Trainingsdaten (Abschnitt 4.2) kamen zum Einsatz:

Netzwerk	Trainingsdaten	Abkürzung	Parameter
Trigger Netzwerk	ratio/uniform	Trigger_Netzwerk	13247
Combinational Netzwerk	ratio/uniform	Combinational_Netzwerk	5173
CNN_single	ratio	CNN_single_ratio	5496
CNN_single	uniform	CNN_single_uniform	3996
CNN_single	exp1	CNN_single_exp1	3851
CNN_single	exp2	CNN_single_exp2	3726
CNN_multiple	ratio	CNN_multiple_ratio	3418
CNN_multiple	uniform	CNN_multiple_uniform	4743
CNN_multiple	exp1	CNN_multiple_exp1	7618
CNN_multiple	exp2	CNN_multiple_exp2	7618

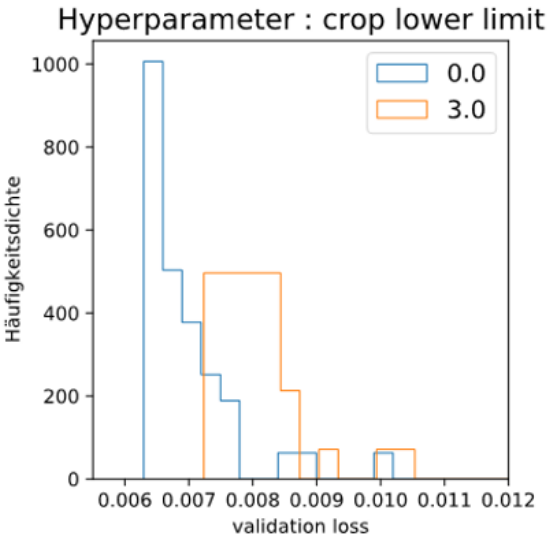
Tabelle 4.1: Benutzte Kombinationen aus den verschiedenen Trainingssets und Netzwerken und der finalen Anzahl an Parametern nach dem Hypertuning

Die insgesamt möglichen Kombinationen an Hyperparametern betrugen bis zu ca. 8200 für ein einzelnes Netz. Da es unmöglich war, alle möglichen Sets zu initialisieren, wählte der Algorithmus pro Netzwerk 200 zufällige aus. Die Validierungsfehler der 200 verschiedenen Sets geben einen Anhaltspunkt, welche Hyperparameter einen großen Einfluss auf die Performance haben und welche nicht. Diese Information ist besonders bei einer möglichen Implementierung auf Field Programmable

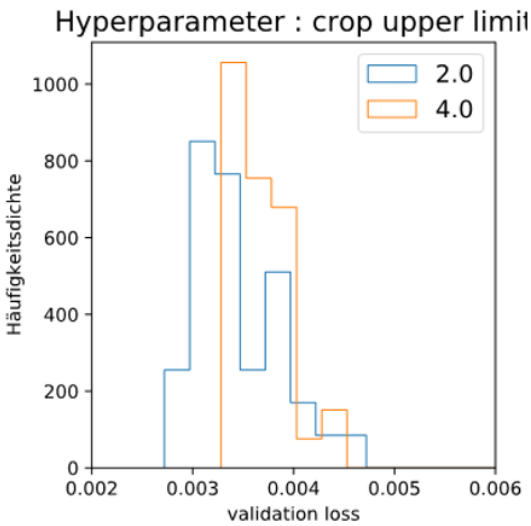
[1] Dieser wird analog zum Testfehler, aber mit dem Validationset, berechnet (siehe Gleichung 2.2.2)

Gate Arrays (FPGAs) interessant. Die Größe der Netzwerke oder z. B. die Anzahl an Inputfeatures können so bezüglich vorhandener Ressourcen optimiert werden.

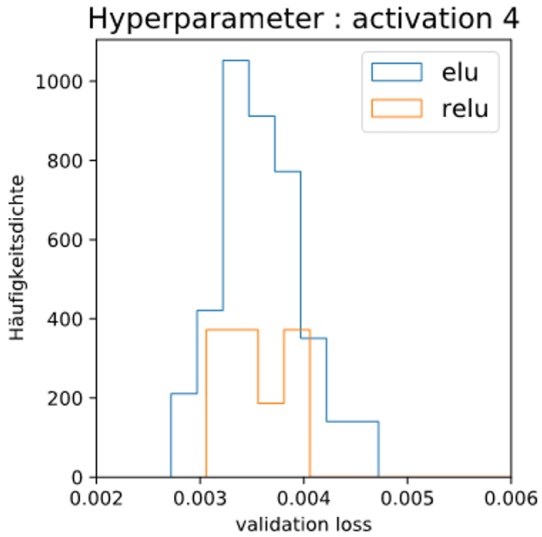
In Abbildung 4.5 sind die Ergebnisse des Hypertunings von 4 ausgewählten Hyperparametern dargestellt. Aufgetragen ist die Verteilung des Validierungsfehlers für die möglichen Werte des gezeigten Hyperparameters. Die in Abbildung 4.5d dargestellten Hyperparameter crop lower limit und crop upper limit beschreiben die Kürzung am Anfang beziehungsweise am Ende des benutzten Receptive Fields von 11 BCs. Bei einem crop lower limit von 3 wird von den selektierten Sequenzen die ersten 3 BCs abgeschnitten. Teile des Pulses gehen dabei möglicherweise verloren, aber eine Reduktion des Receptive Fields führt andererseits zu einer Reduktion der Parameter. Das Hypertuning hat gezeigt, dass für die Netzwerke ein Abschneiden am Anfang der Sequenz zu einer sichtbaren Erhöhung des Validierungsfehlers führt (siehe Abbildung 4.5a). Das Kürzen am Ende der Sequenz, beschrieben durch crop upper limit, hat nur negative Auswirkungen für CNN_multiple (siehe Abbildung 4.5b), nicht aber für CNN_single. Abbildung 4.5c zeigt das Ergebnis für die Aktivierungsfunktion in der Ausgabeschicht von CNN_multiple_exp3 mit einer starken Tendenz für die ELU. In vielen Schichten wurden die in Abschnitt 2.1 beschriebenen Aktivierungsfunktionen ELU und RELU getestet, aber ohne eine allgemeine Präferenz für eine Funktion zu erhalten. Das Histogramm 4.5d zeigt beispielhaft, dass die Erhöhung der Parameter wie hier durch eine größere Anzahl an Feature Maps nicht immer eindeutig zu einem geringeren Validierungsfehler führte und damit die Größe der erstellten Netze der Aufgabe angemessen war.



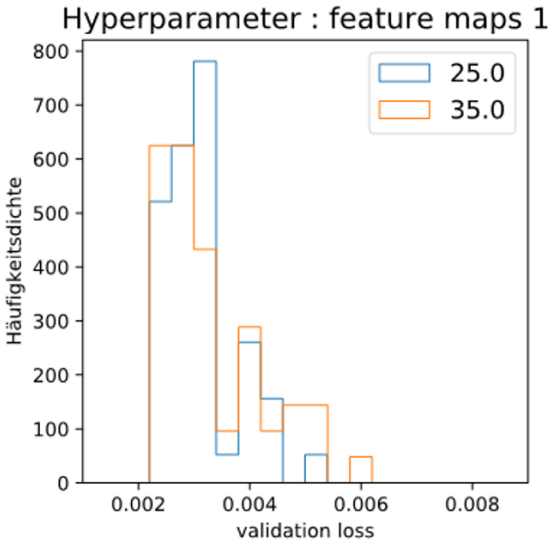
(a) Modell : CNN_single_uniform



(b) Modell : CNN_multiple_exp2



(c) Modell : CNN_multiple_exp3



(d) Modell : CNN_multiple_ratio

Abbildung 4.5: Die Abbildung zeigt 4 Ergebnisse des Hypertunings. Es sind einige Beispiele von Hyperparametern dargestellt, bei denen es zu großen Unterschieden des Validierungsfehlers zwischen den eingestellten Möglichkeiten kam. Die Diagramme zeigen jeweils für jeden möglichen Wert eines Hyperparameters das normierte Histogramm der erzielten Validierungsfehler. Das Histogramm hängt jedoch stark von den anderen Hyperparamtern ab, da diese nicht konstant sind, sondern stattdessen bei jeder Initialisierung zufällig ausgewählt wurden. Auch ist, aufgrund der Sichtbarkeit, nur der relevante Bereich des Histogramms bei minimalem Validierungsfehler dargestellt.

Kapitel 5: Ergebnisse

5.1 Energierekonstruktion

Für die Untersuchung der Leistungsfähigkeit der Netzwerke und dem Vergleich mit dem FIR-Filter kommt der Trainingsdatensatz zum Einsatz. Dieser enthält ausschließlich Daten, die in keiner Weise für das Training oder Hypertuning zum Einsatz kamen, um Aussagen über die Generalisierung treffen zu können. Im Gegensatz zu den Trainingsdaten enthält der Testdatensatz keine selektierten Sequenzen, sondern benutzt die vollständigen Daten aus 4378 oder 5837 Umläufen. Die der Simulation zugrundeliegende Energieverteilung der Hits wurde, wie zuvor beschrieben, modifiziert. Die Tests sind somit keine exakte Nachbildung der ATLAS-Daten während mehrerer Umläufe. Für die Leistung der in dieser Arbeit betrachteten Modelle ist das jedoch ausreichend. Die verwendeten Daten bilden durch die Anzahl der niedrigenergetischen Hits sowie durch die Verschiebung des Pedestals den ADC-Output realistisch nach, während zusätzlich die Performance für hochenergetische Hits verglichen werden kann.
Abbildung 5.1 zeigt die Genauigkeit der Energierekonstruktion in Form der relativen Abweichung zur wahren Energie E_T^{truth}. Die Ergebnisse von Hits in einem Energieabschnitt, mit der Breite von 1 GeV, wurden gemittelt. Punkte mit der gleichen Farbe deuten auf die gleiche Netzwerkstruktur (CNN_single/CNN_multiple) hin und gleiche Marker auf die gleichen Trainingsdaten. Bei niedrigen Energien ($E_T^{\text{truth}} < 4\,\text{GeV}$) besitzen die Modelle den größten relativen Fehler von 0.1 beim CNN_multiple_exp2 bis hin zu 0.6 beim FIR-Filter. Die wahre Energie E_T^{truth} wird im Mittel modellunabhängig unterschätzt. Die größten Unterschiede zwischen den Modellen befinden sich ebenfalls in diesem Energiebereich. Es lassen sich dabei zwei Gruppen erkennen: Netzwerke mit den Trainingsdaten ratio, wie CNN_single_ratio, CNN_multiple_ratio und Trigger Netzwerk, erzielen einen höheren relativen Fehler als die Netzwerke mit den anderen Trainingsdatensätzen. Das Combinational Netzwerk wechselt zwischen beiden Gruppen aufgrund seines Aufbaus. Für kleine Energien befindet es sich in der Gruppe der ratio trainierten Daten. Ab ca. 1.5 GeV, der Grenze zwischen den Subnetzwerken verbessert sich die relative Abweichung und das Combinational

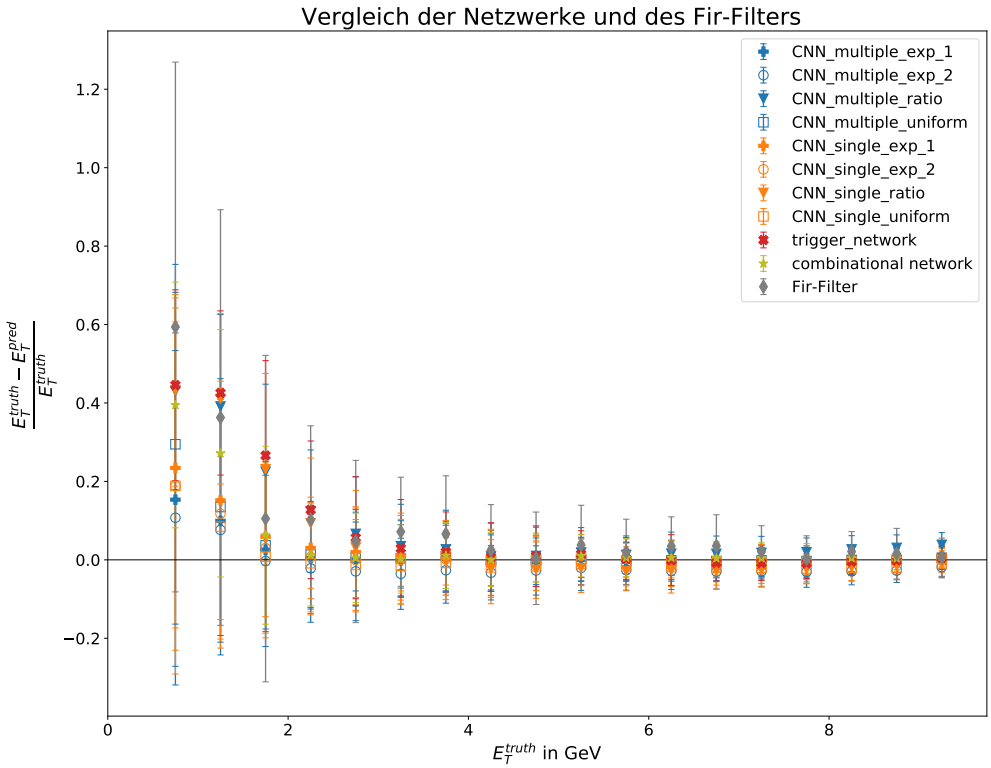
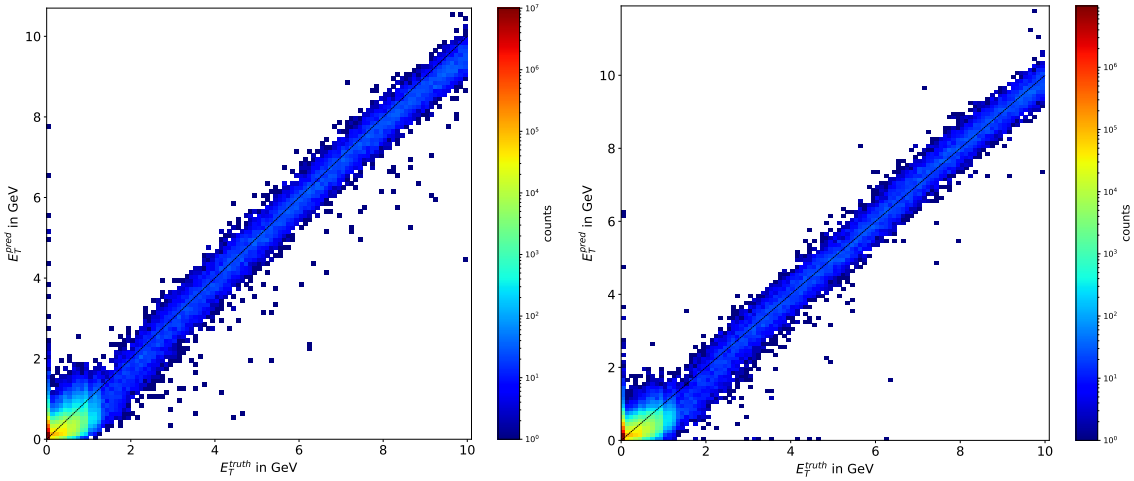


Abbildung 5.1: Der mittlere relative Fehler in Abhängigkeit der wahren Energie für alle trainierten Modelle und den FIR-Filter. Die Testdaten stellen die vollständigen Daten mehrere LHC-Umläufe dar.

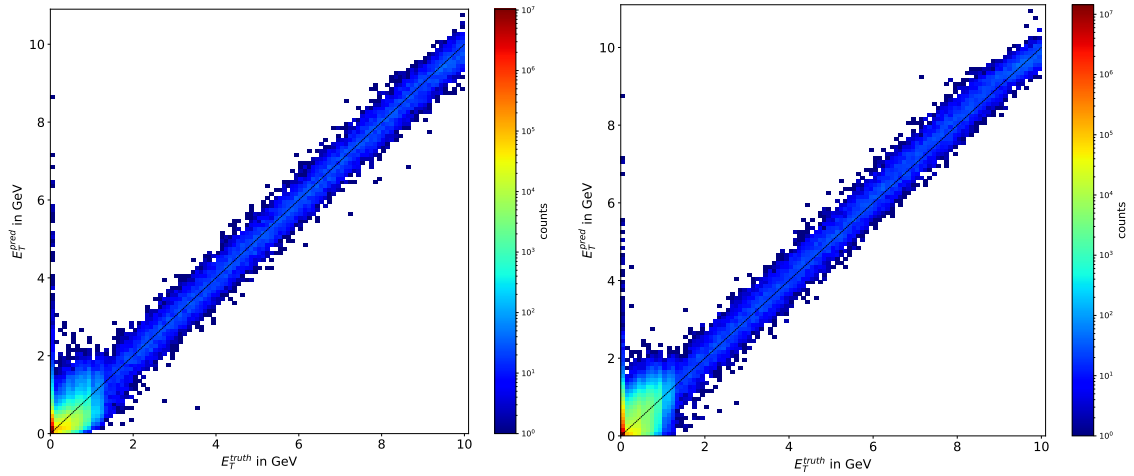
Netzwerk gelangt in die Gruppe der Netzwerke, die nicht mit ratio trainiert wurden. Die Genauigkeit der Energiebestimmung dieser Gruppe ist insbesondere bei kleinen Energien deutlich besser als beim eingesetzten FIR-Filter. Dieser hat aufgrund der verfügbaren 8 Bits für die Energiebestimmung einen deutlichen Nachteil gegenüber den Netzwerken. Bei Hits mit Energien über 4 GeV liegen die Mittelwerte aller getesteten Modelle größtenteils bei 0, wobei manche einen leichten aber konstanten Offset beibehalten. Das CNN_multiple_exp2 hat, z. B. für Energien über 4 GeV, eine mittlere relative Abweichung von -0.3. Abhängigkeiten des Offsets von der Netzwerkstruktur oder den Trainingsdaten sind dabei nicht erkennbar. Auch ist zwischen den Netzwerkstrukturen CNN_single und CNN_multiple kein großer Unterschied feststellbar.

Die Abhängigkeit von dem Trainingsdatensatz ratio wird auch in der Abbildung 5.2 deutlich. Für jede Sequenz in den Testdaten ist die vorhergesagte Energie E_T^{pred} gegen die wahre Energie E_T^{truth} aufgetragen. Die Winkelhalbierende zeigt die Position einer perfekten Vorhersage an. Die Abbildung 5.2 bestätigt die Erkenntnisse aus der Abbildung 5.1. Die größten Unterschiede der Netzwerke liegen bei Energien



(a) Modell : CNN_multiple_ratio

(b) Modell : Trigger Netzwerk



(c) Modell : Combinational Netzwerk

(d) Modell : CNN_single_uniform

Abbildung 5.2: Die $E_T^{\text{truth}}(E_T^{\text{pred}})$ Diagramme zeigen die allgemeine Struktur der Vorhersage für den trainierten Energiebereich. Die Ergebnisse für die anderen Netzwerke sind im Anhang unter C.1 zu finden

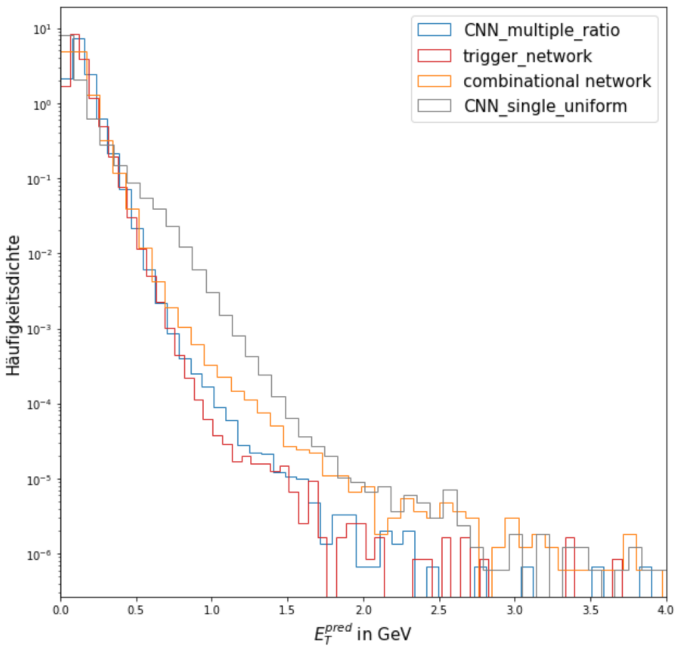


Abbildung 5.3: Die Abbildung zeigt die Verteilung der Energie der fake Hits (Sequenzen, die keinen Hit beinhalten oder bei denen die Pulsamplitude nicht an der 5. Stelle in der Sequenz liegt, sie aber trotzdem vom Netzwerk als Hit detektiert werden). Die dargestellten Modelle entsprechen den aus Abbildung 5.2.

unterhalb von 4 GeV. Bei höheren Energien ist die Vorhersage größtenteils um die wahre Energie zentriert mit eine mittleren Standardabweichung von 0.35 GeV für das gezeigte CNN_multiple_ratio. Die Auflösung in diesem Energiebereich ist durch das thermische Rauschen limitiert. Die Abbildung 5.2a zeigt, dass die in Abbildung 5.1 beobachteten größeren relativen Fehler der Netzwerke mit dem ratio Trainingsdatensatz aus einer Anpassung an eine charakteristische Form resultieren. Für Energien kleiner 2 GeV sind Vorhersagen weniger zentriert, wie es bei CNN_single_uniform (5.2d) der Fall ist, aber auch deutlich weniger gestreut. Dies zeigt auch die Betrachtung der Sequenzen ohne implementierte Pulse (Abbildung 5.3), welche die Testdaten dominieren. Diese Anpassung führt allerdings zu schlechteren Vorhersagen bei Sequenzen mit der Enegie von ca. 2 GeV, sodass Vorhersagen erst wieder ab 4 GeV um den wahren Wert zentriert sind. Diese Form entsteht aufgrund der Zusammensetzung der Trainingsdaten ratio, welche hauptsächlich aus Sequenzen mit Energien kleiner als 1 GeV bestehen. Das Combinational Netzwerk (5.2c) erreicht für kleine Energien eine geringe Streuung, ohne die charakteristische Form von CNN_single_ratio anzunehmen.

Das Trigger Netzwerk (Abbildung 5.2b) erzielt die beste Performance für die Sequenzen ohne Hit. Da es aber auch mit dem ratio Trainingsdatensatz trainiert wurde, ist die gleiche charakteristische Form wie bei CNN_m_ratio erkennbar. In Abbildung 5.2 sind bei allen Netzwerken auffälligen Sequenzen mit $E_T^{\text{truth}} = 0$, welche

aber mit Energien von bis zu 9 GeV vorhergesagt wurden, sichtbar. Diese Fehler entstehen durch Situationen, bei denen hochenergetische Hits einen Abstand von nur wenigen BCs besitzen. Aufgrund der Überlagerung der Pulse erkennt das Netzwerk in dem BC zwischen den Pulsen einen hochenergetischen Hit. Diese Ergebnisse sind selten und Resultat der veränderten Energieverteilung für die Hits.

5.2 Noisecut und Triggereigenschaften

Neben der Energieberechung der Hits ist die zweite wichtige Aufgabe die Bestimmung des richtigen BCs, in dem ein Hit auftrat. Abbildung 5.3 zeigt, dass die Netzwerke bei Sequenzen ohne einen Hit an der sensitiven Stelle häufig triggern und einen sogenannten fake Hit detektieren. Das Netzwerk CNN_multiple_ratio hat z. B. eine mittlere Ausgabe von 0.13 GeV bei Sequenzen ohne Hit. Die Ursache für die hohe fake Hit-Rate liegt in den Trainingsdaten. Diese beinhalten selbst Pulse mit einer Energie kleiner als $\sigma_{thermal}$. Für das Netzwerk war es nicht möglich, Hits dieser geringen Energie von Sequenzen mit reinem Noise zu unterscheiden. Als Resultat ist das Netzwerk sehr sensitiv auf leichte Erhöhungen des ADC-Outputs und generiert viele fake Hits. Vermieden werden die fake Hits durch die Einführung eines Noisecuts. Dieser setzt alle Ausgaben des Netzwerks auf Null, welche unterhalb einer Schwelle liegen. Die Höhe der Schwelle muss dabei festgelegt werden.

Berechnung des Noisecuts

Die Energie von fake Hits definiert die Höhe des benötigten Noisecuts. Die Schwelle sollte so klein wie möglich sein, damit weniger Hits verloren gehen. Für die Netzwerke zeigt sich, dass die Energien der fake Hits 1-2 BCs neben einem hochenergetischen Hit maximal sind. Ein Maß für die Anzahl der fake Hits um einen echten Hit stellt der Timing Score dar. Er ist definiert als die Anzahl von fake Hits, die in den 2 BCs vor und nach einem Hit entstehen, gemittelt über den gesamten Energiebereich der getesteten Hits. Der FIR-Filter erreicht mit seinem Noisecut ($\approx$ 1.4 GeV) einen Timing Score von 0.005. Dies bedeutet, dass es im Mittel bei 0.5 % der getesteten Hits zu einem fake Hit kam. Abbildung 5.4 zeigt den gemessenen Timing Score als Funktion des Noisecuts für das Netzwerk CNN_multiple_exp2. Der finale Noisecut des Netzwerks wird durch den Schnittpunkt mit dem Timing Score des FIR-Filters festgelegt. Liegt der Schnittpunkt zwischen zwei Datenpunkten, wie es für das gezeigte Netzwerk der Fall ist, wurde exponentiell interpoliert. Die erreichten Noisecuts der Netzwerke sind in Tabelle 5.1 zu sehen. Der in Abbildung 5.3 angedeutete Vorteil der Modelle Trigger_Netzwerk und CNN_multiple_ratio macht sich durch die geringen Noisecuts von 0.93 GeV und 1.05 GeV bemerkbar. Die restlichen Netzwerke erzie-

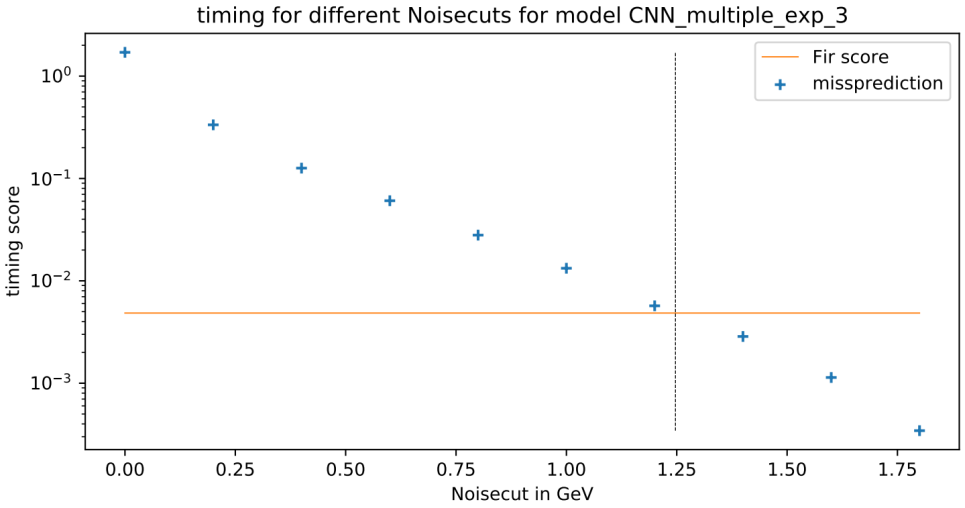


Abbildung 5.4: Der Timing Score in Abhängigkeit des angewendeten Noisecuts. Der Timing Score des FIR-Filters ist als orange Linie dargestellt. Der interpolierte Schnittpunkt ergibt den Noisecut des Netzwerks.

len untereinander einen ähnlichen Noisecut im Bereich von $1.2 - 1.27\,\text{GeV}$. Lediglich CNN_single_exp2 hat mit $1.36\,\text{GeV}$ einen auffallend großen Noisecut. Alle Netzwerke erzielen einen niedrigeren Noisecut als der FIR-Filter.

Netzwerk	Noisecut in GeV
Trigger_Netzwerk	0.93
Combinational_Netzwerk	1.27
CNN_single_ratio	1.04
CNN_single_uniform	1.2
CNN_single_exp1	1.24
CNN_single_exp2	1.36
CNN_multiple_ratio	1.05
CNN_multiple_uniform	1.2
CNN_multiple_exp1	1.23
CNN_multiple_exp2	1.25

Tabelle 5.1: Erhaltene Noisecuts mit der Methode des Timing Scores für die Netzwerke.

Leistungsfähigkeit der Trigger

Für die im vorherigen Abschnitt berechneten Noisecuts kamen lediglich Sequenzen mit isolierten Hits zum Einsatz. Um die Leistungsfähigkeit der Netzwerke und des FIR-Filters zu untersuchen, werden diese, wie zuvor bei der Untersuchung der Energierekonstruktion, mit vollständigen LHC-Umläufen getestet. Zur Beurteilung der Leisungsfähigkeit wird die Präzision und Relevanz betrachtet. Die Präzision, auch Effizient genannt, beschreibt den Anteil der Ereignisse mit einer Energie größer als

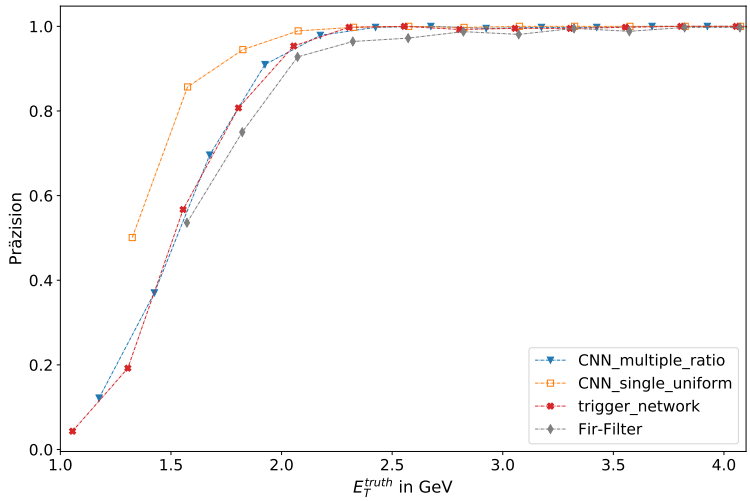
dem Noisecut, die im richtigen BC detektiert wurden. Werden alle dementsprechenden Ereignisse gefunden, liegt die Präzision bei 1. Dieses Ergebnis erreicht allerdings auch ein Trigger, der auf jedes beliebige Ereignis reagiert. Deshalb wird zusätzlich die Relevanz, auch Reinheit genannt, berechnet. Sie beschreibt Anteil aus allen Ereignissen mit einer vorhergesagten Energie über dem Noisecut, deren wahre Energie auch über dem Noisecut liegt. Präzision (Effizienz) und Relevanz (Reinheit) sind definiert durch

$$\text{Präzision} = \frac{\text{RP}}{\text{RP} + \text{FN}} \tag{5.2.1}$$

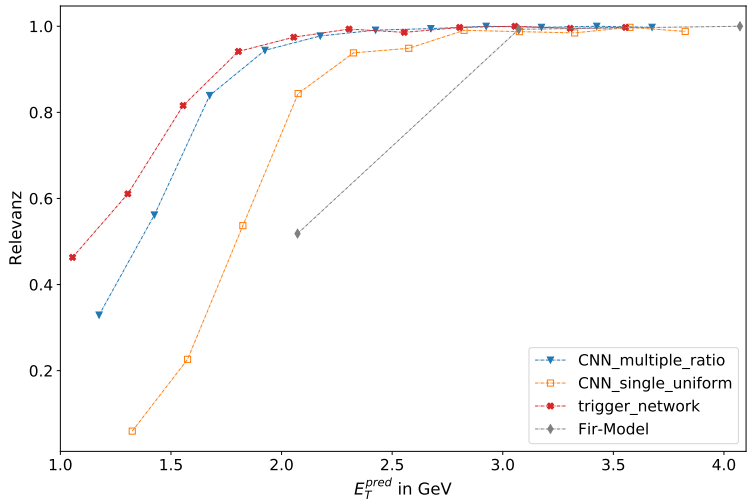
und

$$\text{Relevanz} = \frac{\text{RP}}{\text{RP} + \text{FP}}. \tag{5.2.2}$$

Die richtig positiven(RP) Klassifikationen sind Sequenzen mit einer Energie über dem Noisecut, welche auch dementsprechend detektiert werden. Bei den falsch positiven (FP) liegt die wahre Energie unterhalb des Noisecuts, die Vorhersage allerdings darüber. Ist die Vorhersage kleiner als der Noisecut und die wahre Energie größer, handelt es sich um eine falsch negative (FN) Klassifikation.
Abbildung 5.5 zeigt das Ergebnis für 3 Netzwerke und den FIR-Filter. Unter den gezeigten Netzwerken hat das Netzwerk CNN_single_uniform die beste Effizienz bei der Detektion von niedrigenergetischen Hits. Die Netzwerke CNN_multiple_ratio und Trigger_Netzwerk zeigen keine großen Unterschiede und erreichen erst ab ca. 2.2 GeV eine Präzision von 1. Alle Netzwerke erzielen eine bessere Effizienz bei der Detektion von Hits, als der eingesetzte FIR-Filter. Das Modell CNN_single_unifrom erzielt auf der einen Seite die höchste Effizienz, hat aber auf der anderen Seite eine schlechtere Reinheit der Vorhersagen. Erst ab einer vorhergesagten Energie von ca. 2.75 GeV erreicht es eine Relevanz von 1, wohingegen dem Trigger_Netzwerk dies schon bei ca. 2 GeV gelingt. Die Verwendung eines vortrainierten Triggers bei dem Trigger_Netzwerke resultiert in eine bessere Relevanz, wie der Vergleich mit dem CNN_multiple_ratio zeigt. Die Auflösung der Relevanz für den FIR-Filters ist aufgrund des für E_T^{pred} verfügbaren Bereichs von 8 Bit eingeschränkt. Eine höhere Reinheit und damit weniger fake Hits bei der Detektion von Hits ist bei den Netzwerken trotzdem erkennbar.



(a) Diagramm der Präzision



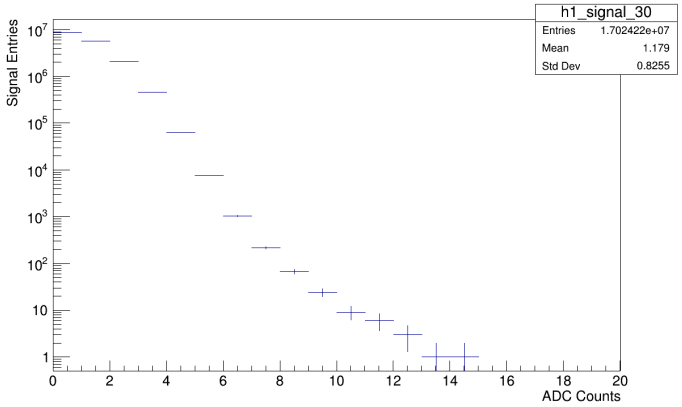
(b) Diagramm der Relevanz

Abbildung 5.5: In (a) ist die Präzision als Funktion von E_T^{truth} der getesteten Sequenzen für ausgewählte Netzwerke und dem FIR-Filter dargestellt. Die Relevanz (b) ist in Abhängigkeit von E_T^{pred} dargestellt und beschreibt die Wahrscheinlichkeit, dass die wahre Energie eines detektierten Ereignisses über dem Noisecut liegt. Die Ergebnisse für die anderen trainierten Netzwerke befinden sich im Anhang unter C.2.

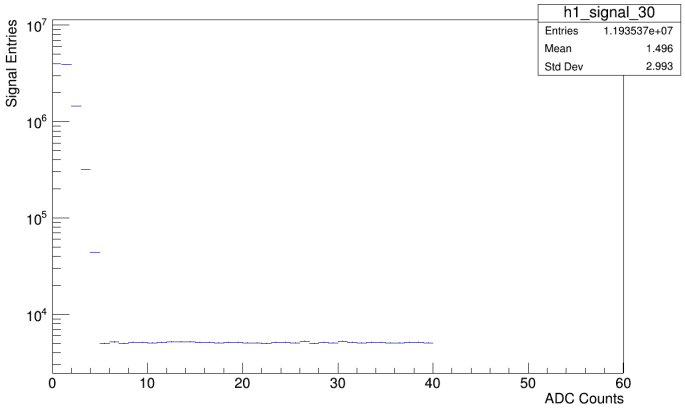
Kapitel 6: Zusammenfassung

Zur Reduktion der Datenrate benötigt der ATLAS-Detektor ein leistungsfähiges Triggersystem. Dieses besteht unter anderem in der ersten Stufe aus dem L1Calo-System. Zu dessen Aufgaben zählt es, die im Kalorimeter gemessenen Teilchen zu detektieren und deren Energie zu berechnen. Im Hinblick auf die höheren Luminositäten nach dem Phase-II Upgrade wurde untersucht, ob diese Aufgaben ein neuronales Netz genauer und zuverlässiger bewerkstelligen kann, als der bisher eingesetzte FIR-Filter.
Die für die Arbeit mit neuronalen Netzen benötigten Daten wurden durch die Simulation ToyMC generiert. Es wurde das Signal eines TTs im TileLB für ein $\langle\mu\rangle = 40$ aus vielen simulierten LHC-Umläufen verwendet. Die Untersuchung der Daten hat gezeigt, dass für diesen Detektorbereich das thermische Rauschen die dominierende Rauschquelle darstellt und der Pile-Up vernachlässigbar ist. Für die Untersuchung des Einflusses der Trainingsdaten wurden aus diesen Daten insgesamt 4 Sets mit unterschiedlicher Energieverteilung erstellt.
Das Training mit diesen Sets fand mit mehreren verschiedenen Netzwerkarchitekturen statt. Die grundlegende Struktur bildete dabei ein Convolutional Neural Network. Mehrere Ansätze zur Netzwerkstruktur wurden miteinander verglichen. Durch ein Hypertuning und Early Stopping konnten die Hyperparameter optimal festgelegt werden.
Der Test mit Daten aus LHC-Umläufen hat gezeigt, dass die Netzwerke bei der Auflösung der Energierekonstruktion und Effizienz der Detektion eine bessere Leistung erzielen als der bisher eingesetzte FIR-Filter. Die Netzwerke erreichen bei niedrigeren Noisecuts eine bessere Effizienz der Detektion bei zugleich höherer Reinheit. Die Leistungsfähigkeit des Triggers kann somit durch die Netzwerke erhöht werden. Aus dem Vergleich der Netzwerke untereinander wird sichtbar, dass die Zusammensetzung der Trainingsdaten den größten Einfluss auf die Leistung der neuronalen Netze hat. Veränderungen in der Architektur haben im Vergleich dazu geringe Auswirkungen auf die Leistungsfähigkeit.
In den weiteren Schritten muss die Leistungsfähigkeit bei einem erwarteten $\langle\mu\rangle$ von 140-200, für das geplante Phase-II Upgrade, getestet werden. Außerdem ist eine Erweiterung des Energiebereichs notwendig, sowie die Implementierung auf FPGAs.

Anhang A: Amplitudenverteilungen der implementierten Ereignisse



(a) unveränderte Verteilung

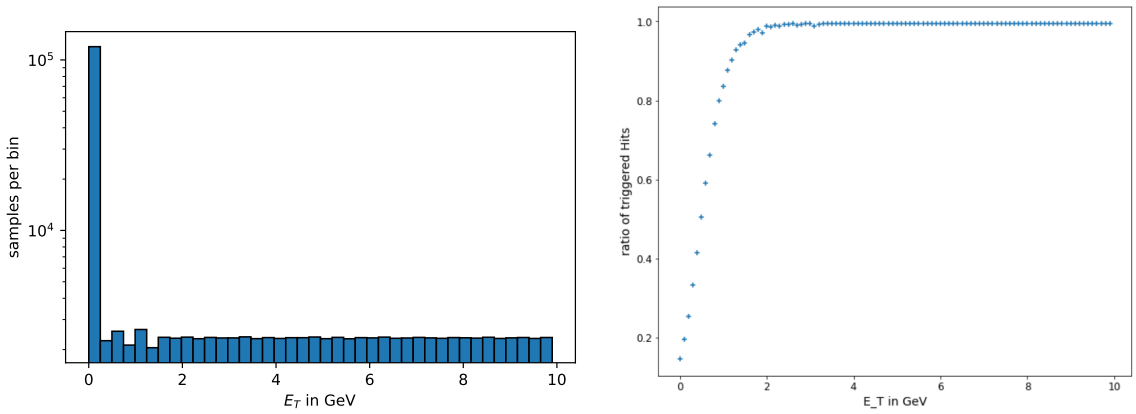


(b) veränderte Verteilung

Abbildung A.1: Verwendete Verteilung der Amplituden für die eingefügten Pulse. Die Veränderung der Verteilung in A.1a erhöht das Auftreten von höherenergetischen Ereignissen.

Anhang B: Leistungsfähigkeit von Subnetzwerken

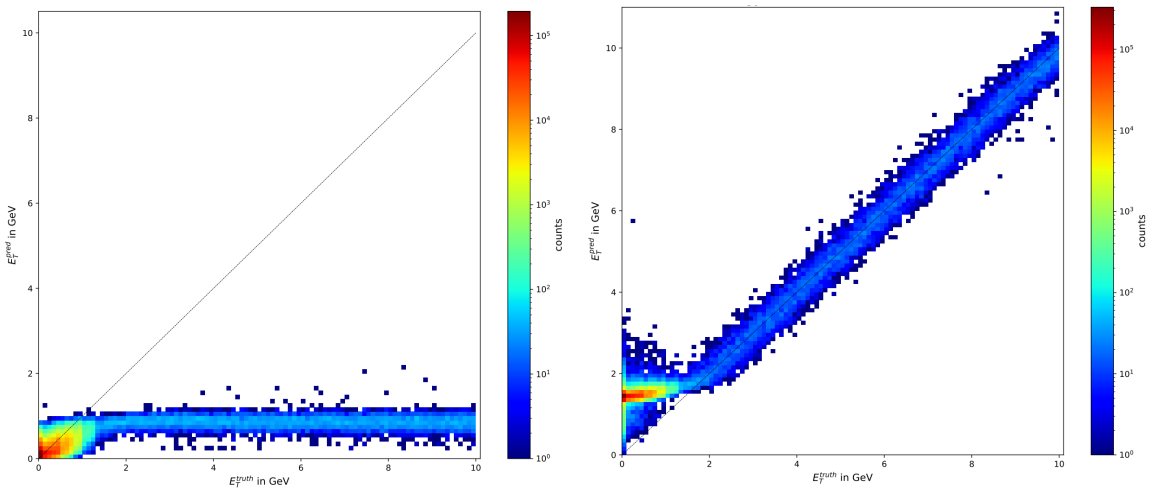
Trigger aus Trigger_Netzwerk



(a) Trainingsdaten des Trigger Subnetzwerks (b) Leistungsfähigkeit des Triggers

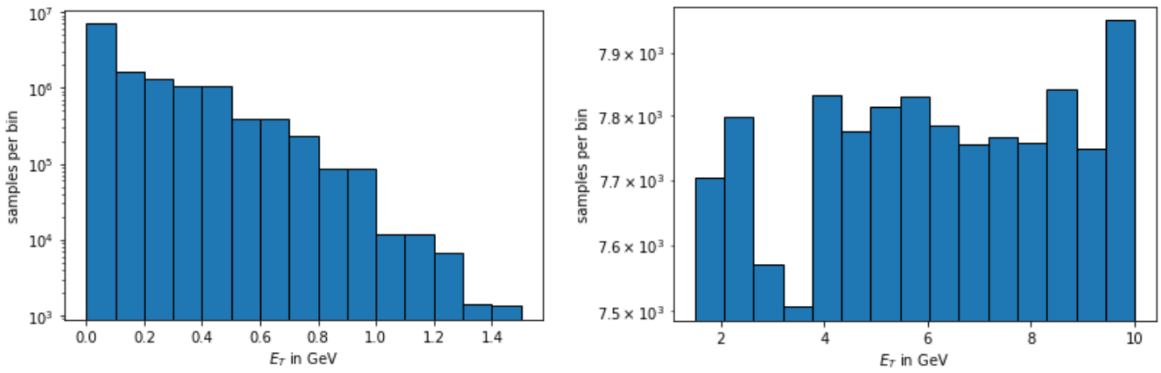
Abbildung B.1: Für die Trainingsdaten des Subnetzwerks Trigger wurde eine uniforme Energieverteilung ausgewählt (a). Anstatt einer wahren Energie enthalten die Trainingsdaten eine 1 für einen Hit und ansonsten eine 0. Die Ausgabe des Triggers liegt zwischen 0 und 1, wodurch sie mit der idealen Ausgabe verglichen werden kann. In (b) ist die Leistungsfähigkeit des Triggers dargestellt. Die Ausgabe des Netzwerks wurde für jeden Energiebereich gemittelt. Ab zwei 2 GeV erreicht der mittlere Output 1, sodass alle Hits detektiert werden. Es ist aber auffällig, dass der mittlere Output, insbesondere für den Energiebereich um 0 GeV, niemals Null ist. Dies lässt darauf schließen, dass viele Sequenzen ohne Hit falsch detektiert werden. Auf die Verwendung der Ausgabe des Triggers durch das E_cal Netzwerk hat man keinen Einfluss.

Combinational Netzwerk



(a) E_low Netzwerk (b) E_high Netzwerk

Abbildung B.2: Die Abbildung zeigt die Performance der Subnetzwerke in Form von $E_T^{\text{truth}}(E_T^{\text{pred}})$-Diagrammen. Die Testdaten stellen keine vollständigen Umläufe des LHC, wie in Abbildung 5.2, dar. Es wurden selektierte Sequenzen mit einer Verteilung ähnlich zu 4.4a verwendet.
(a) : Das Netzwerk hat sich bestmöglich an die Sequenzen mit $E_{\text{T}} < 1.5\,\text{GeV}$ angepasst. Da die Trainingsdaten keine Sequenzen mit $E_{\text{T}} > 1.5\,\text{GeV}$ beinhalteten, ist die Vorhersage für diesen Energiebereich schlecht.
(b) : Die Leistungsfähigkeit ist umgekehrt zu E_low in (a): Gute Anpassung für $E_{\text{T}} > 1.5\,\text{GeV}$ und schlechte für $E_{\text{T}} < 1.5\,\text{GeV}$



(a) Trainingsdaten von E_low (b) Trainingsdaten von E_high

Abbildung B.3: Die Trainingsdaten von E_low sind in (a) und von E_high in (b) dargestellt. Um die Spezialisierung auf die Energiebereiche zu erreichen enthalten die Trainingsdaten lediglich Sequenzen aus dem entsprechenden Energiebereich(für E_low von $0 - 1.5\,\text{GeV}$ und für E_high von $1.5 - 10\,\text{GeV}$).

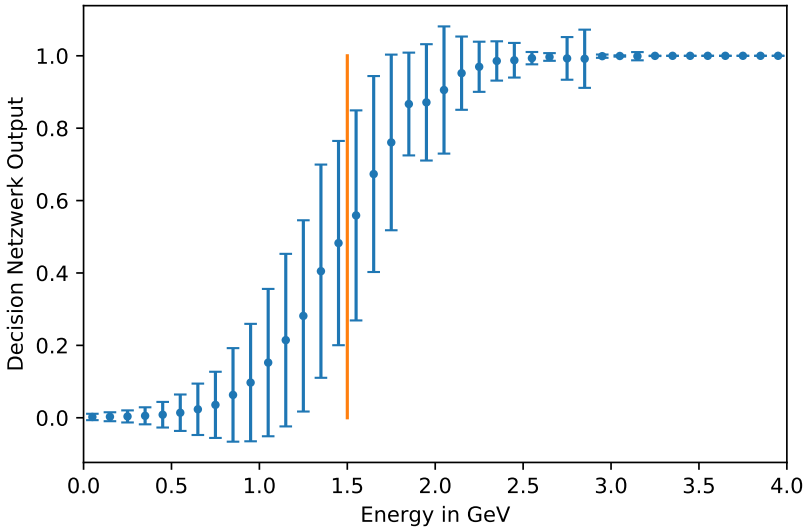


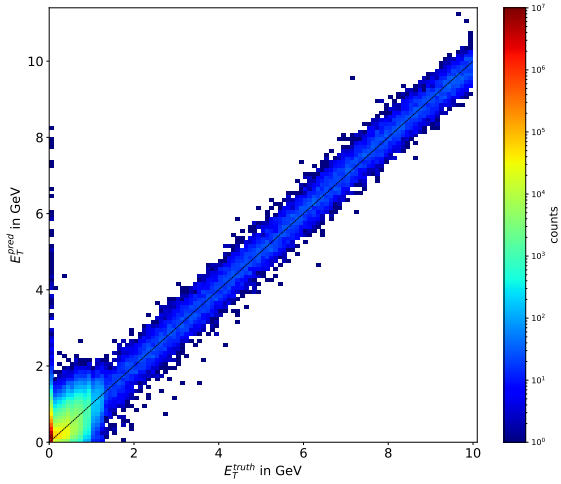
Abbildung B.4: Die Abbildung zeigt den mittleren Output des Decision Netzwerk für verschiedene Energien der Input Sequenzen. Die Ausgabe liegt zwischen 0 und 1. Die Fehlerbalken stammen aus einem Gauss'schen Fit. Die vertikale Linie zeigt die Grenze des Detektionsbereichs von E_small und E_high.

Trainingsdaten

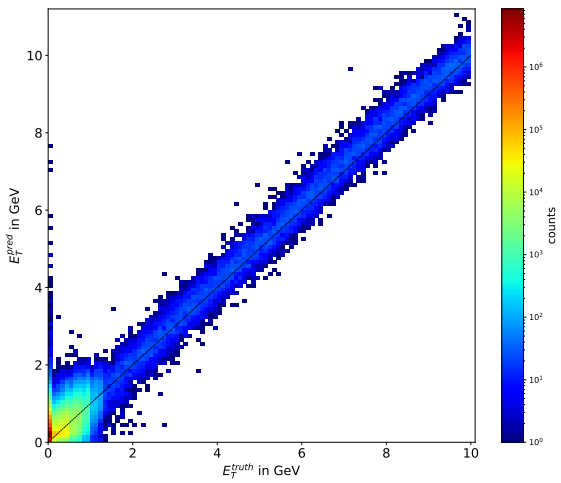
Trainingsdatenverteilung	Anzahl an verwendeter Sequenzen
ratio	324000
uniform	228096
exp 1	131364
exp 2	154772

Tabelle B.1: Aufgrund der charakteristischen Form der Verteilungen kommt es zu unterschiedlich große Trainingsets.

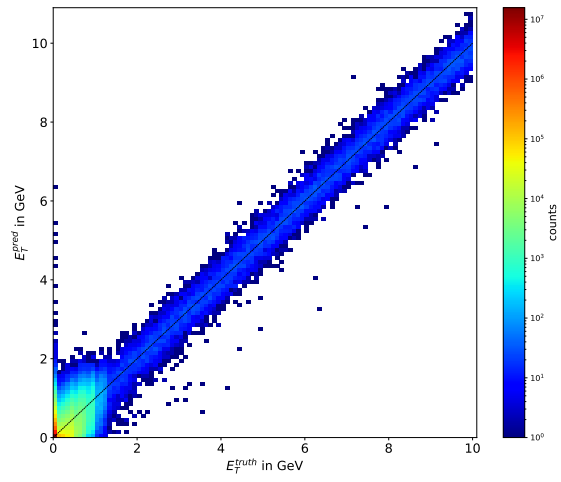
Anhang C: Zusätzliche Ergebnisse



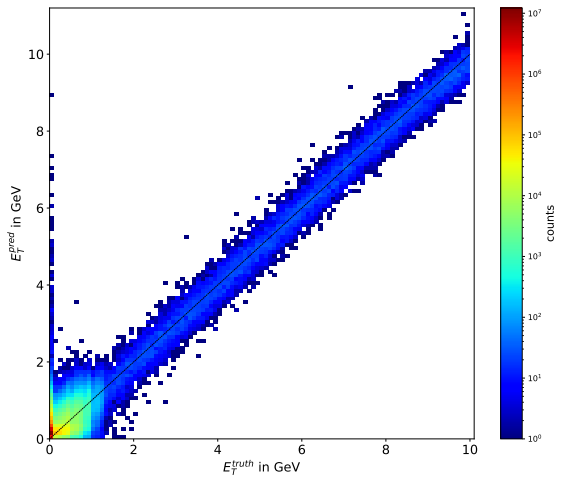
(a) Modell : CNN_multiple_exp1



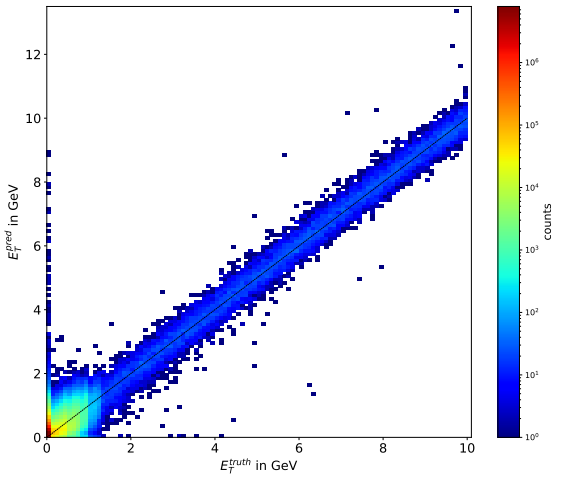
(b) Modell : CNN_multiple_exp2



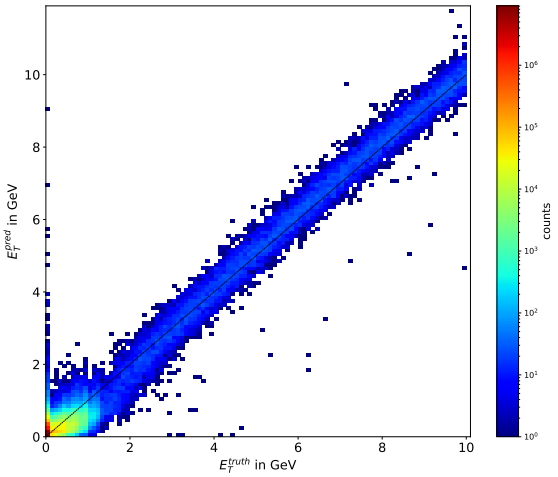
(c) Modell : CNN_multiple_uniform



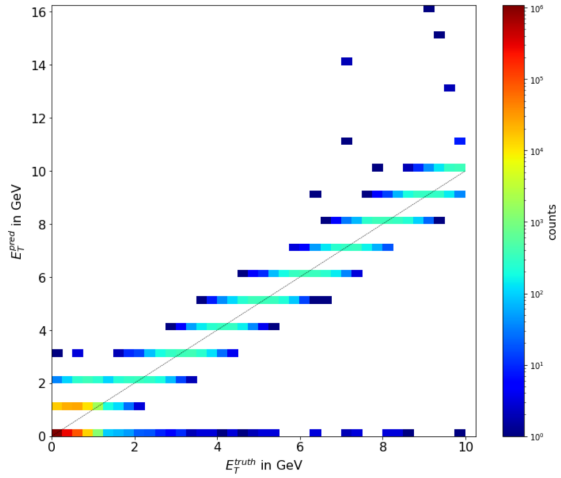
(d) Modell : CNN_single_exp1



(e) Modell : CNN_single_exp2

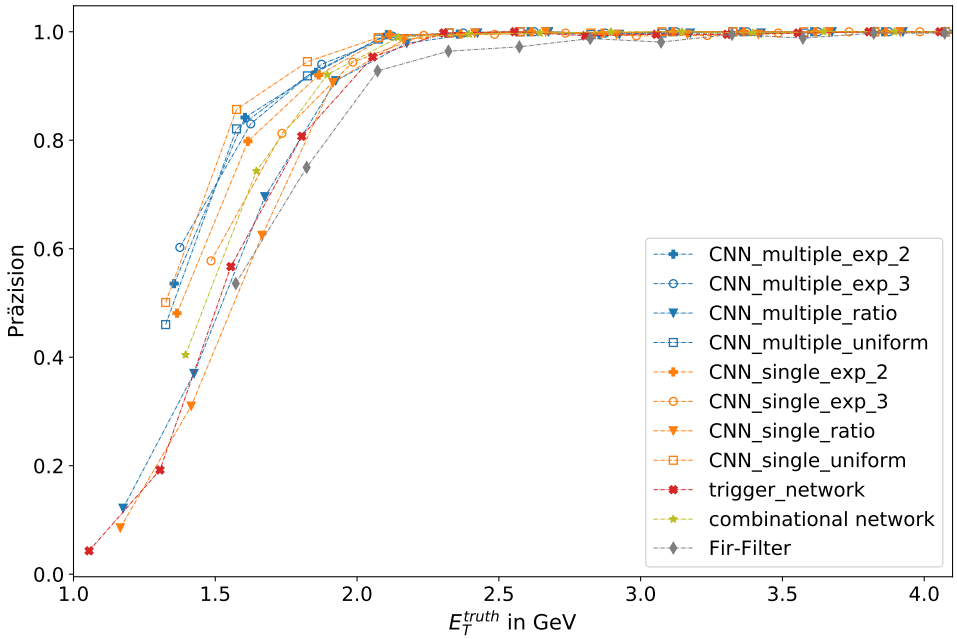


(f) Modell : CNN_single_ratio

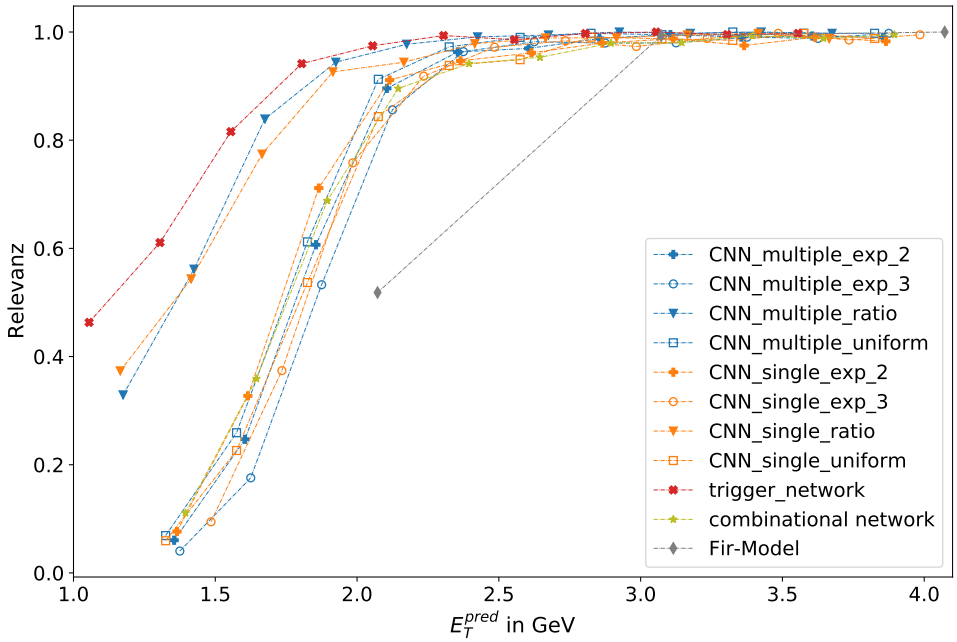


(g) Modell : Fir-Filter

Abbildung C.1: Die Abbildung zeigt die $E_T^{\text{truth}}(E_T^{\text{pred}})$ Diagramme der restlichen trainierten Netzwerke



(a) Diagramm der Präzision



(b) Diagramm der Relevanz

Abbildung C.2: In (a) ist die Präzision als Funktion von E_T^{truth} der getesteten Sequenzen für alle trainierten Netzwerke und den FIR-Filter dargestellt. In (b) ist die Relevanz in Abhängigkeit der vorhergesagten Energie E_T^{pred} für alle trainierten Modelle und den FIR-Filter dargestellt

Abbildungsverzeichnis

Literatur

[1] The ATLAS Collaboration u. a. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08003–S08003. DOI: 10.1088/1748-0221/3/08/s08003. URL: https://doi.org/10.1088/1748-0221/3/08/s08003.

[2] S. Chatrchyan u. a. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Physics Letters B* 716.1 (2012), S. 30–61. ISSN: 0370-2693. DOI: https://doi.org/10.1016/j.physletb.2012.08.021. URL: https://www.sciencedirect.com/science/article/pii/S0370269312008581.

[3] Georges Aad u. a. "Artificial Neural Networks on FPGAs for Real-Time Energy Reconstruction of the ATLAS LAr Calorimeters". In: *Computing and Software for Big Science* 5.1 (Okt. 2021), S. 19. ISSN: 2510-2044. DOI: 10.1007/s41781-021-00066-y. URL: https://doi.org/10.1007/s41781-021-00066-y.

[4] Andrew Christopher Daniells. "Pile-Up Suppression in the ATLAS Level 1 Calorimeter Trigger and Searches for Higgs Boson Pair Production". Dissertation. University of Birmingham, 2016. URL: https://etheses.bham.ac.uk/id/eprint/7025/.

[5] Philippe Mouche. *Overall view of the LHC.* [Online; besucht am 03.01.2022]. 2014. URL: https://cds.cern.ch/record/1708847.

[6] Tha ATLAS Collaboration. *Luminosity determination in pp collisions at* $\sqrt{s} = 13$ *TeV using the ATLAS detector at the LHC.* ATLAS CONF NOTE. [Online; besucht am 02.01.2022]. URL: http://cds.cern.ch/record/2677054/files/?ln=de.

[7] I. Béjar Alonso et al. (Eds.) *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report.* CERN-2020-010. Geneva: CERN Yellow Reports: Monographs, 2020. DOI: 10.23731/CYRM-2020-0010.

[8] Joao Pequenao. *Computer generated image of the whole ATLAS detector*. [Online; besucht am 03.01.2022]. 2008. URL: https://cds.cern.ch/images/CERN-GE-0803012-01.

[9] M. Aaboud u. a. "Operation and performance of the ATLAS Tile Calorimeter in Run 1". In: *The European Physical Journal C* 78.12 (Nov. 2018). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-018-6374-z. URL: http://dx.doi.org/10.1140/epjc/s10052-018-6374-z.

[10] G. Aad u. a. "Performance of the upgraded PreProcessor of the ATLAS Level-1 Calorimeter Trigger". In: *Journal of Instrumentation* 15.11 (Nov. 2020), P11016–P11016. DOI: 10.1088/1748-0221/15/11/p11016. URL: https://doi.org/10.1088/1748-0221/15/11/p11016.

[11] The ATLAS Collaboration, G Aad und E Abat. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08003–S08003. DOI: 10.1088/1748-0221/3/08/s08003. URL: https://doi.org/10.1088/1748-0221/3/08/s08003.

[12] W.E. Cleland und E.G. Stern. "Signal processing considerations for liquid ionization calorimeters in a high rate environment". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 338.2 (1994), S. 467–497. ISSN: 0168-9002. DOI: https://doi.org/10.1016/0168-9002(94)91332-3. URL: https://www.sciencedirect.com/science/article/pii/0168900294913323.

[13] J.D. Morris. *Analogue Input Calibration of the ATLAS Level-1 Calorimeter Trigger – TWEPP-09*. 2009. URL: http://cds.cern.ch/record/1212908/?ln=de.

[14] Falk Bartels. "Kalibration des Finite-Impulse-Response-Filters im PreProcessor des ATLAS Level-1 Calorimeter Triggers für den LHC Run-2". Bachelorarb. Heidelberg, Germany: Universität Heidelberg, 2015.

[15] Tom M. Mitchell. *Machine learning*. Boston: McGraw-Hill WCB, 1997.

[16] Ewen Callaway. "'IT WILL CHANGE EVERYTHING': AI MAKES GIGANTIC LEAP IN SOLVING PROTEIN STRUCTURES". In: *Nature* 588 (Dez. 2020), S. 203–204.

[17] WARREN S. MCCULLOCH und WALTER PITTS. "A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY". In: *BULLETIN OF MATHEMATICAL BIOPHYSICS* 5 (1943), S. 115–133.

[18] Aurélien Géron. *Praxiseinstieg Machine Learning mit Scikt-Learn, Keras und TensorFlow*. O'Reilly Verlag, 2020. ISBN: 978-3-96009-124-0.

[19] The ATLAS Collaboration. *Convolutional Neural Networks with Event Images for Pileup Mitigation in E_T^{miss} reconstruction with the ATLAS Detector*. ATLAS PUB Note. [besucht am 13.01.22]. Juli 2019. URL: https://cds.cern.ch/record/2684070?ln=de.

[20] The ATLAS Collaboration. *Identification of Jets Containing b-Hadrons with Recurrent Neural Networks at the ATLAS Experiment*. ATLAS PUB Note. [besucht am 13.01.22]. März 2017. URL: http://cds.cern.ch/record/2255226/files/?ln=de.

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

[22] Djork-Arné Clevert, Thomas Unterthiner und Sepp Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 2016. arXiv: 1511.07289 [cs.LG].

[23] Johann Christoph Voigt. "Optimisation of the FPGA Firmware Implementation of Convolutional Neural Networks for the ATLAS LAr Calorimeter Signal Processing". Magisterarb. Dresden, Tech. U., 2021.

[24] Diederik P. Kingma und Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[25] Martín Abadi u. a. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[26] François Chollet u. a. *Keras*. https://github.com/fchollet/keras. 2015.

[27] W. R. Bennett. "Spectra of quantized signals". In: *The Bell System Technical Journal* 27.3 (1948), S. 446–472. DOI: 10.1002/j.1538-7305.1948.tb01340.x.

[28] Tom O'Malley u. a. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

Tabellenverzeichnis

Abkürzungsverzeichnis

LHC Large Hadron Collider

ATLAS A Toroidal LHC ApparatuS

ALICE A Large Ion Collider Experiment

CMS Compact Muon Solenoid

LHCb Large Hadron Collider beauty

BC Bunch Crossing

FIR Finite-Impulse-Response

JEP JetEnergy Processor

ROIs Regions of Interest

CP Cluster Processor

L1Calo Level-1 Calorimeter Trigger

ADC Analogue-to-Digital Converter

L1Muon L1 Muon Trigger

CTP Central Trigger Processor

LUT Look-Up Table

PPr PreProcessor

TT Trigger Tower

LSB least significant Bit

ANN Artifical Neural Networks

ML Machine Learning

CNN Convolutional Neural Network

MLP Multilayer Perceptron

ReLU Rectified Linear Unit

ELU Exponential Linear Unit

FPGA Field Programmable Gate Array

TileLB Tile Long Barrel

HLT High-Level Trigger

L1 Level-1

L1A Level-1 Accept

Erklärung

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

Heidelberg, den 08.02.2022