TECHNISCHE
UNIVERSITÄT
DRESDEN

Anwendung von künstlichen neuronalen Netzen auf die Energierekonstruktion der ATLAS-Flüssigargon-Kalorimeter

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

vorgelegt von

PHILIPP WELLE
geboren am 27.05.2001 in BAD SALZUNGEN

Fakultät Physik
Bereich Mathematik und Naturwissenschaften
Technische Universität Dresden
2022

Eingereicht am 27. Juni 2022

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

Zusammenfassung

Für das Flüssigargonkalorimeter des ATLAS-Detektors ist im Zuge des LHC-High-Luminosity-Upgrade eine Verbesserung der Messelektronik geplant, die mit der erhöhten Zahl der Teilchenkollisionen besser umgehen kann. Zu diesem Zweck wird geplant neuronale Netzwerke auf FPGAs zu benutzen. In dieser Arbeit werden Möglichkeiten dargelegt, wie das Training von neuronalen Netzwerken mit Hilfe von Hyperparametersuchen optimiert werden kann. Des Weiteren werden die Effekte, die die sogenannte Bunch-Train-Struktur der Protonenpakete auslöst, im Bezug auf die Energierekonstruktion mit Hilfe von einem generierten Datensatz untersucht.
Die Bunch-Train-Struktur sorgt für ein zeitlich variables Pile-Up, das in der Energierekonstruktion klar erkennbar ist. Eine Baselinekorrektur kann die Effekte auf Kosten der Erkennungseffizienz in der Strahllücke korrigieren. Ein neu trainiertes Netzwerk mit optimierter Struktur mit binärer Bunch-Train-Struktur-Eingabe kann noch bessere Performance in Bezug auf die Bunch-Train-Struktur erreichen, aber verliert dabei Genauigkeit in der grundlegenden Energierekonstrukion. Die Hyperparameteroptimierung zeigt sich als flexible Lösung, um größere Anzahlen verschiedener neuronaler Netzwerke zu trainieren.

Abstract

For the liquid argon calorimeter of the ATLAS detector, an improvement of the measurement electronics is planned in the course of the LHC high luminosity upgrade, which can better handle the increased number of particle collisions. To this end, neural networks on FPGAs are planned to be used. In this thesis, ways to optimize the training of neural networks using hyperparameter searches are presented. Furthermore, the effects caused by the so-called bunch-train structure of the proton packets are investigated in terms of the energy reconstruction using a generated data set.
The bunch-train structure provides a time-varying pileup clearly visible in the energy reconstruction. Baseline correction can correct for the bunch-train effects at the expense of detection efficiency in the beam gap. A re-trained network with optimized structure for binary bunch-train structure input can achieve even better performance in terms of negating bunch-train effects, but loses accuracy in the basic energy reconstruction. Hyperparameter optimization is shown to be a flexible solution to train larger numbers of different neural networks.

Inhaltsverzeichnis

1 Einleitung

Die kleinsten Bausteine der Materie sind durch das Standardmodell der Teilchenphysik beschrieben. Mit dem Nachweis des Higgs-Bosons 2012 [1] am ATLAS sowie CMS-Detektor des Large Hadron Collider (LHC) wurden alle vorhergesagten Teilchen nachgewiesen. Seitdem wird der LHC für die Suche für Physik jenseits des Standardmodells sowie der weiteren Charakterisierung der gefundenen Teilchen benutzt. Um die Chance von Entdeckungen der sehr seltenen Ereignisse zu erhöhen, soll ab 2026 die Luminosität des LHCs mit dem High-Luminosity Upgrade [2] verzehnfacht werden.
Die Flüssigargonkalorimeter im ATLAS-Detektor messen bei Proton-Proton-Kollisionen die Energie von den Endprodukten, die in Form von Pulsen in der Ausleseelektronik vorliegen. Die Erhöhung der Luminosität führt dort zu einer verstärkten Signalüberlagerung, wodurch ein Upgrade der Elektronik notwendig wird. Die momentan benutzten Linearfilter sollen dann durch künstliche neuronale Netzwerke auf FPGAs, einer Art integrierten Schaltkreis, ersetzt werden, die die Energierekonstruktion übernehmen. Künstliche neuronale Netzwerke haben die Eigenschaft, dass sie trainiert werden und damit die Problemlösung lernen. Bevor ein solches Training jedoch stattfinden kann, müssen bestimmte Parameter definiert werden, welche Struktur des Netzwerks oder Art des Trainings festlegen.
Diese Hyperparameter bestimmen hauptsächlich die Fähigkeit der Netzwerke, die jeweilige Probleme zu lösen. In dieser Arbeit werden Algorithmen vorgestellt und implentiert, die bei der Wahl dieser Hyperparameter helfen sollen. Anstelle von normalen Trainings werden Hyperparameteroptimierungen durchgeführt.
In der bisherigen Forschung zu den neuronalen Netzwerken haben sich bestimmte Netzwerkstrukturen als optimal für das Problem durchgesetzt. Ein Teil der Arbeit ist es, mit Hilfe der Hyperparameteroptimierung nach Alternativen zu suchen, die möglicherweise bessere Performance aufzeigen.
Die Füllweise des LHC mit Protonen hat einen Einfluss auf die Messung der Flüssigargonkalorimeter. Eine weitere Anwendung der Hyperparameteroptimierung wird sein, nach Möglichkeiten zu suchen, diese Effekte zu begrenzen.

2 Theoretischer Hintergrund

2.1 Large Hadron Collider

2.1.1 Large Hadron Collider

Der *Large Hadron Collider* (LHC) ist ein Proton-Proton- sowie Ion-Ion-Teilchenbeschleuniger. Der LHC ist der größte Teilchenbeschleuniger der Welt mit seinem Hauptbeschleunigerring, der einen Umfang von 26.7 km hat. Im Hauptring werden momentan im Run3 Protonenstrahlen entgegengesetzt auf maximal 6.8 TeV beschleunigt, womit eine Schwerpunktsenergie bis zu 13.6 TeV erreicht werden kann. [3]

2.1.2 ATLAS-Detektor

Einer der Teilchendetektoren, der die Reaktionsprodukte der Protonkollisionen misst, ist der ATLAS-Detektor. Der zylindrische Detektor mit 46 m Länge und 25 m Durchmesser bei einer Masse von rund 7000 t ist der größte Teilchendetektor der Welt. Der Detektor folgt einem schalenförmigen Aufbau, zu sehen in Abb. 2.1. Nach einer Kollision erreichen Teilchen als erstes den inneren Teil des Detektors, der aus dem Pixeldetektor, Semiconductor Tracker (SCT) und dem Transition Radiation Tracker (TRT) besteht. Hier werden Richtung, Impuls und Ladung der in den Protonenkollision entstandenen Teilchen gemessen. [5]
Der Pixeldetektor besteht aus vier Schichten Siliziumpixel, die in hoher Auflösung von fast 10 µm den Ursprung und den Impuls des Teilchen aus der Bahnkrümmung in einem Magnetfeld bestimmen. Dieser ist umgeben vom Semiconductor Tracker, der aus über 4000 Siliziumsensormodulen besteht. Diese werden zur Flugbahnbestimmung benutzt.
Der Transition Radiation Tracker (Übergangsstrahlungsdetektor) nutzt die Abhängigkeit der Übergangsstrahlung vom Lorentzfaktor, die entsteht, wenn geladene Teilchen Grenzflächen zwischen zwei Materialien mit unterschiedlichen Dielektrizitätskonstan-

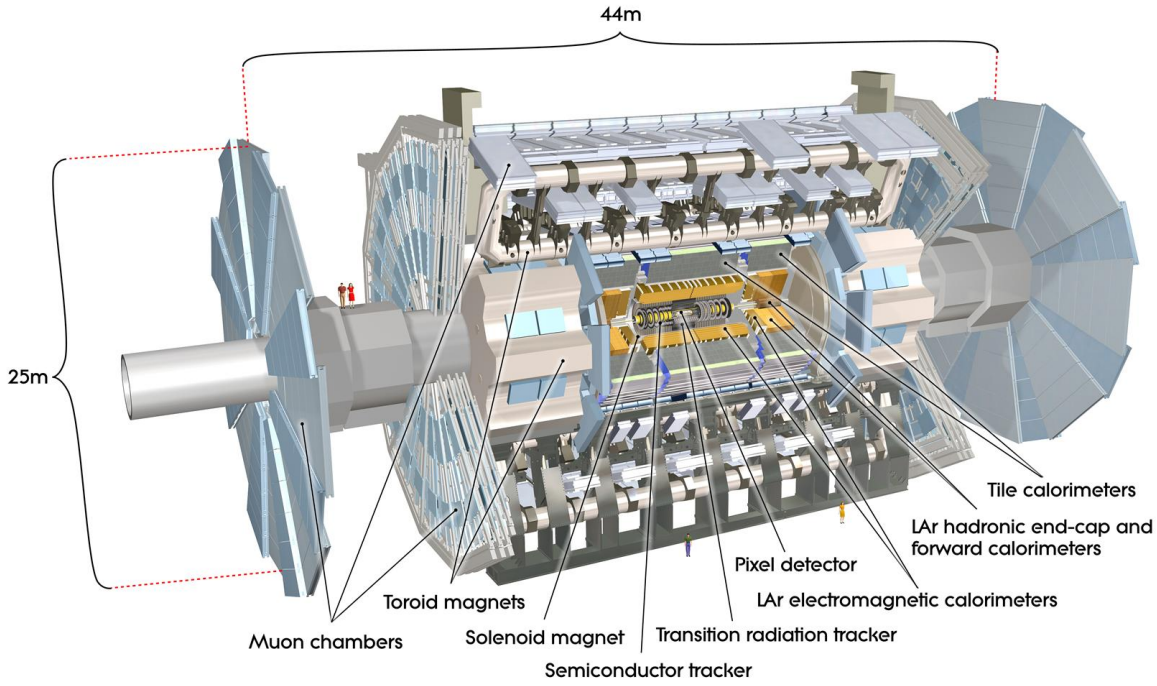


Abbildung 2.1: Computergenerierter Aufbau des ATLAS-Detektor [4]

ten durchqueren. Dadurch kann im TRT zwischen schweren Hadronen und leichten Leptonen unterschieden werden. [5]
Auf den inneren Detektor folgen die Kalorimeter, erst das elektromagnetische Flüssigargonkalorimeter, in dem Elektronen und Photonen gestoppt werden. Hadronische Teilchen geben dort nur einen Teil ihrer Energie ab, bevor sie im hadronischen Kalorimeter gestoppt werden. Das hadronische Kalorimeter ist unterteilt in den Endkappenkalorimeterteil (HEC) sowie Vorwärtskalorimeterteil (FCal) bestehend aus Flüssigargonkalorimetern und den Zylinderteil mit Siliziumdetektorkacheln. In den Kalorimetern wird die Energie der gestoppten Teilchen bestimmt. Myonen und Neutrinos durchqueren die bisherigen Detektorteile, ohne gestoppt zu werden. In den einrahmenden Myonenkammern werden Impuls und Spur der Myonen bestimmt. Neutrinos können nicht gemessen werden, aber ihre Energie kann durch die hermetisch abgeschlossene Messung bestimmt werden. [6]

2.1.3 Flüssigargonkalorimeter

Die Flüssigargonkalorimeter, zu sehen in Abb. 2.2, bestimmmen die Energie von elektromagnetischen Teilchen und im HEC-Teil von hadronischen Teilchen. Das Kalori-

meter ist symmetrisch in der Pseudorapidität η aufgebaut, die mit dem Polarwinkel θ definiert ist. (Gl. 2.1)

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] \tag{2.1}$$

Der mittlere Teil ist der elektromagnetische Zylinder (EMB) mit $|\eta| < 1.475$. Die elektromagnetischen und hadronischen Endkappen (EMEC und HEC) decken Bereiche von $1.375 < |\eta| < 3.2$ beziehungsweise $1.5 < |\eta| < 3.2$ ab. Das Vorwärtskalorimeter (FCal) deckt dann einen Bereich von $3.1 < |\eta| < 4.9$ hadronisch und elektromagnetisch ab.

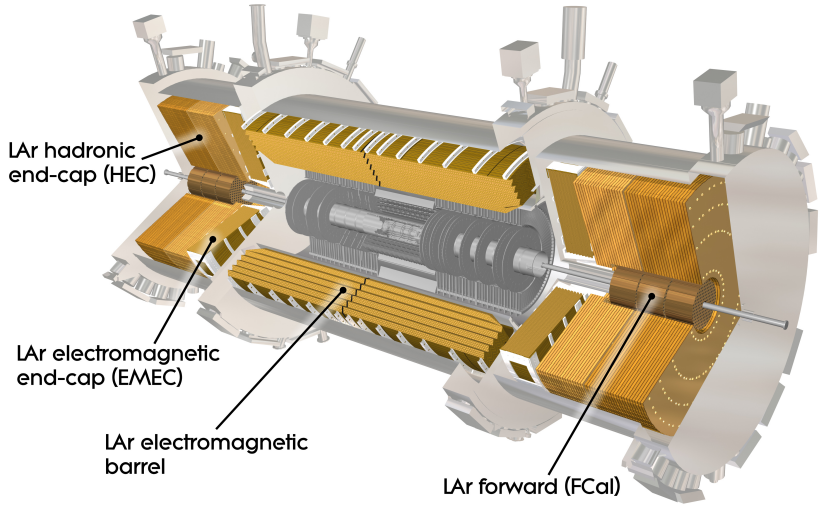


Abbildung 2.2: Computergeneriertes Bild des Flüssigargonkalorimetersystems [7]

Das Absorbermaterial im EMB und EMEC ist Blei, was in einer Akkordiongeometrie schichtenweise aufgebaut ist. Im HEC wird stattdessen Kupfer in parallelen Platten benutzt, damit hadronische Schauer entstehen. Das FCal funktioniert mit zylindrischen Elektroden, die aus konzentrisch angebrachten Stäben parallel zur Strahlachse bestehen. [8, S. 8]
Durch die Schichtstruktur kann eine Auflösung im Radius r hergestellt werden. Die

Auflösung in der Pseudorapidität η und des Azimutwinkels ϕ wird durch Aufteilung in Zellen erreicht.
Die LAr-Ausleseelektronik bestimmt die deponierte Energie von minimal 50 MeV bis maximal 3 TeV für insgesamt 182468 Kanäle. In der Front-End-Elektronik, die direkt in Detektornähe installiert ist, werden die Pulse mit einer Frequenz von 40 MHz abgetastet. Eine Frequenz von 40 MHz entspricht einer Abtastung alle 25 ns. Das ist auch die Frequenz in der die Protonenbündel, die sogenannten Bunches, getaktet sind. Somit passiert alle 25 ns eine Strahlkreuzung, welche als Bunch Crossing (BC) bezeichnet wird. [9]
Vor der Abstastung wird in der Front-End-Elektronik auch die Pulsform geändert. Die Umformung (Abb. 2.3) in bipolare Pulse sorgt dafür, dass das Integral über einen Puls 0 ist. In der Back-End-Elektronik werden aus diesen umgeformten, danach digitalisierten Pulsen eine Energie rekonstruiert. [8, S. 11]

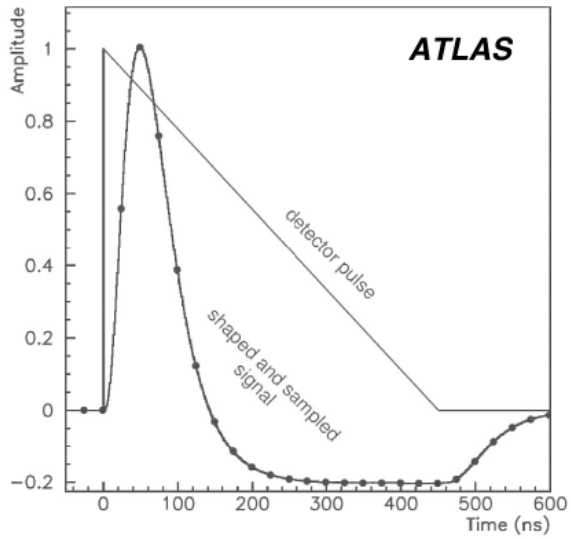


Abbildung 2.3: Umformung der Detektorpulse in der Front-End-Elektronik [8, S. 11]

Dazu wird bisher der *Optimalfilter* (OF) eingesetzt, der erst die bekannte Signalform benutzt, um Pulse zu finden und das Rauschen zu verringern. Das bekannte Signal wird aus der Optimierung der Signal-to-Noise-Ratio zu einem Linearfilter umstrukturiert. Dieser wird über das Signal gefaltet. Aus den geglätteten Pulsen wird dann die Peakhöhe als Energie abgelesen, mit dem sogenannten MaxFinder, der die Peaks als lokale Maxima identifiziert. Zusammen können der Optimalfilter und der MaxFinder

die Energie aus den Pulsen rekonstruieren. [10, S. 144ff]

2.1.4 LHC Phase-2-Upgrade

Der LHC ist mit einer Schwerpunktsenergie von 14 TeV konzipiert worden. Diese wurde nicht sofort erreicht. In Abb. 2.4 ist der Upgradeplan des LHC zu sehen. Seit 2011 wurde langsam die Schwerpunktsenergie bis zu dem angestrebten Maximum erhöht. Weiterhin wird auch die Luminosität erhöht, im jetzigen Run3, auf den doppelten nominellen Wert. [2]
Nach der Erhöhung der Schwerpunktsenergie steht ab 2026 die Erhöhung der Luminosität an. Das hat den Effekt, dass pro Bunch Crossing sich die Anzahl an Teilchenkollisionen erhöht. Das ist ein Problem für den Optimalfilter. Dies erhöht die Chance, pro BC eine bestimmte Zelle zu treffen, wodurch sich die Häufigkeit von erkannten Pulsen erhöht, und öfter zwei Pulse mit geringem zeitlichen Abstand zu beobachten sind. Dies sorgt für sogenanntes Pile-up, also die Überlagerung zweier Pulse. [9]

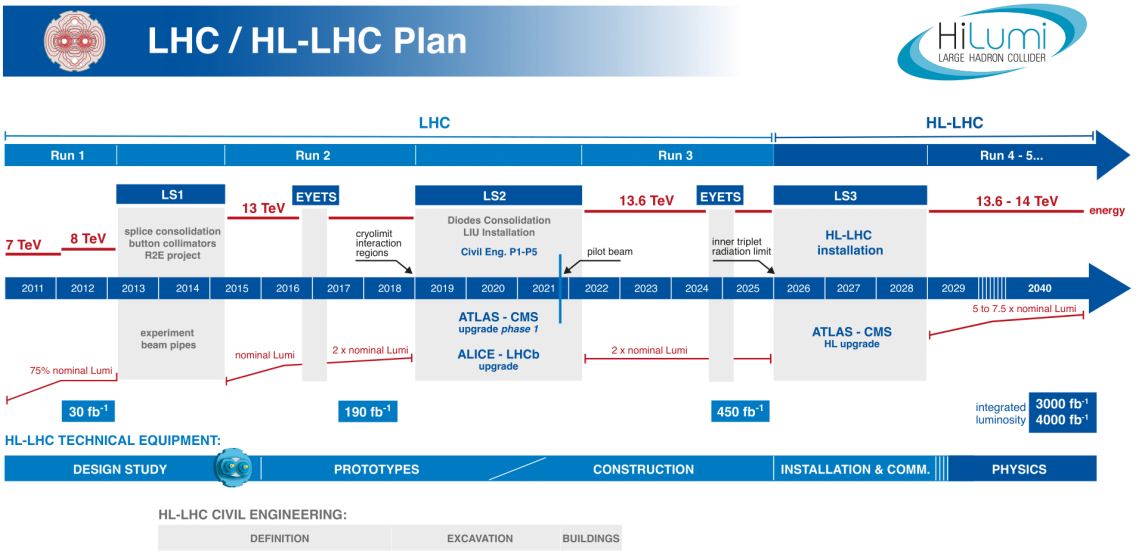


Abbildung 2.4: HL-LHC Upgrade Plan (Stand Februar 2022) [2]

2.2 Maschinelles Lernen

2.2.1 Grundlagen

Beim maschinellen Lernen werden Modelle aufgrund von Eingangsdaten, auch Trainingsdaten genannt, unter Nutzung von Algorithmen entwickelt. Diese Modelle können für verschiedenste Problemstellungen entwickelt werden, bei denen konventionelle algorithmische Lösungen zu komplex oder nicht zielführend sind. Mit Hilfe von mehr Trainingsdaten können bereits kreierte Modelle weiter verbessert werden. Maschinelles Lernen kann in drei Klassen eingeteilt werden:

- **Supervised Learning**: Hierbei sind in den Trainingsdaten die Eingabedaten sowie gewünschte Ausgabedaten. Ein mathematisches Modell wird in vielen Berechnungsdurchläufen die Fähigkeit antrainiert aus den Eingabedaten die Ausgabedaten zurückzugeben. In diese Kategorie fallen Regressionen oder neuronale Netzwerke.
- **Unsupervised Learning**: Für die gegebenen Trainingsdaten erzeugen Algorithmen mathematische Modelle, die Zusammenhänge oder erkannte Kategorien enthalten. Dadurch sind Vorhersagen möglich. Auch können so aus hochdimensionalen Daten mit Hilfe von "Dimensionalty Reduction"in niederdimensionale Daten umgewandelt werden. [11, S.146ff]
- **Reinforcement Learning**: Mit Hilfe von Feedbackloops beschäftigt sich dieser Bereich mit der Frage, wie ein Agent, ein Programm, welches zu autonomen und eigenständigen Verhalten fähig ist, sich in einer Umgebung zu optimal zu verhalten hat.

Die Form vom maschinellen Lernen, die in dieser Arbeit benutzt wird, ist supervised learning, genauer künstliche neuronale Netzwerke. Die Idee folgt biologischen neuronalen Netzwerken, in denen Neuronen chemisch mit weiteren verbunden sind. Diese Verbindungen werden Synapsen genannt. Die Signalübertragung funktioniert via elektrischer Potentiale. Wenn das Potential an einem Neuron die Aktionsschwelle überschreitet, wird ein Puls zu den nachfolgenden Neuronen gesendet. [12] [11, S.104ff] Die grundlegende Aufgabe neuronaler Netzwerke ist die Näherung einer Funktion, die die Abhängigkeit der Eingabe x von der Ausgabe y beschreibt.

$$y = f^*(x) \tag{2.2}$$

Diese Annäherung wird mit einer Abbildung in Abhängigkeit von den Parametern θ definiert, bei der das Netzwerk die beste Wahl von θ lernt. [11, S. 168 ff.]

$$y = f^*(x; \theta) \tag{2.3}$$

Die Bezeichnung Netzwerk folgt der Grundstruktur, Modelle sind Verschachtelung verschiedener Funktionen in der Form $f(x) = f^{(n)}(f^{(n-1)}(...f^1(x)))$. Die natürliche Zahl n gibt die Tiefe des Netzwerks an, jede $f^{(m)}$ wird als Schicht bezeichnet. Die Schichten bestehen aus Knoten, entsprechend der Dimension der vorherige Schicht. Das Modell wird meist mit azyklischen Graphen dargestellt, wie in Abb. 2.5.

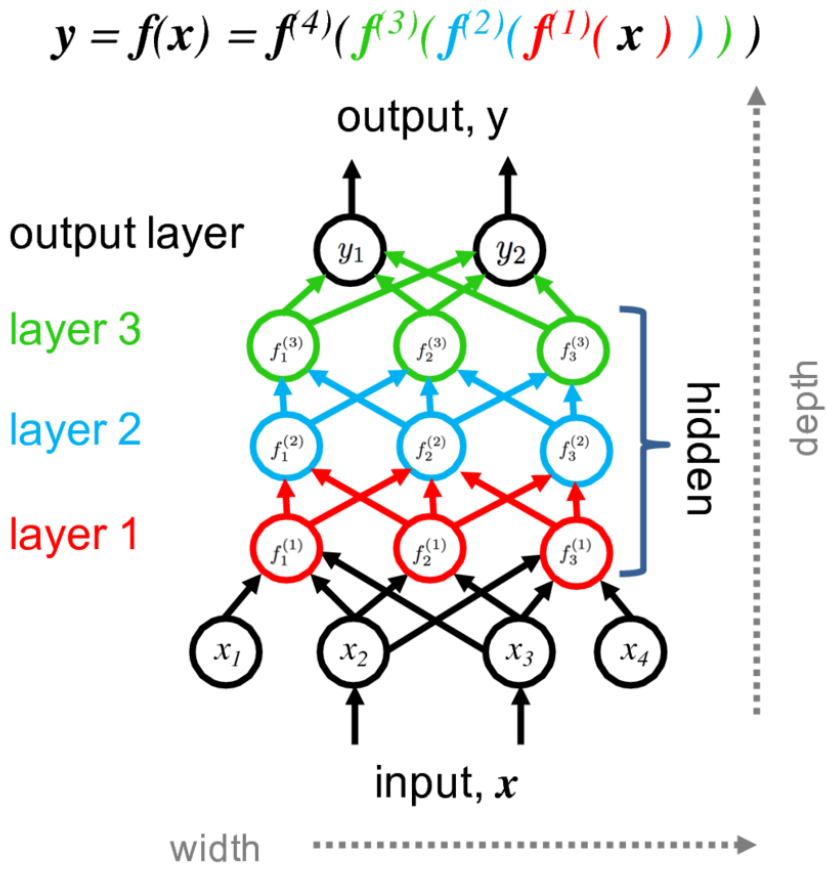


Abbildung 2.5: Graph eines neuronales Netzwerk [13]

Jede Knoteneingabe z in einer Schicht wird als Linearkombination mit den Werten der vorherigen Knoten $\vec{x}$ mit den Gewichten $\vec{W}$ und der Verschiebung (Bias) b berechnet.

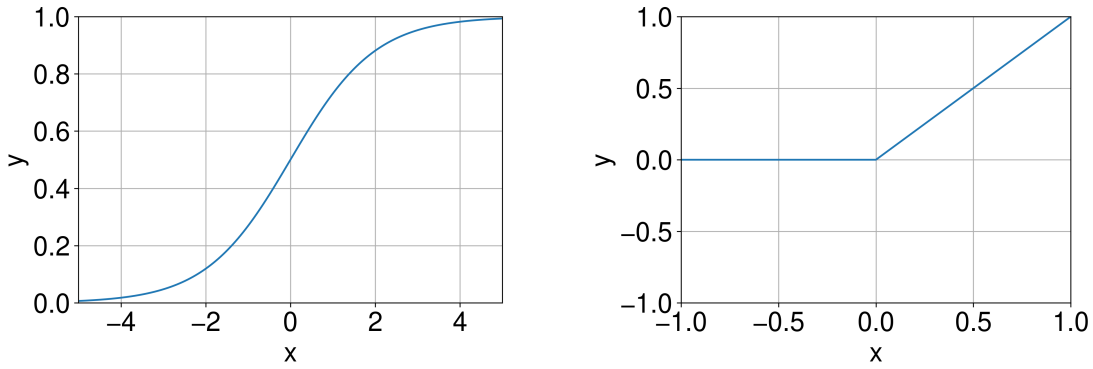
Im Knoten wird eine Funktion auf dieses Wert angewendet, die Aktivierungsfunktion Φ. Dadurch bildet sich die Ausgabe des Knoten x' in Gl. 2.4. [13]

$$x' = \Phi(\vec{W} \cdot \vec{x} + b) \tag{2.4}$$

Die Aktivierungsfunktion hat die Aufgabe für Nichtlinearitäten zu sorgen. Damit sollen auch komplexere Problemstellungen besser lösbar werden. Beispiele sind die Sigmoidfunktion (Gl. 2.5, Abb. 2.6a) oder die ReLU-Funktion. (Gl. 2.6, Abb. 2.6b). Beim Training werden sogenannte Verlustfunktionen benutzt, die die Abweichung zwischen Modellausgabe und Zielwert auf eine reelle Zahl abbilden. Während des Trainings wird dann versucht, diese Verlustfunktion zu minimieren. [11, S. 82]

$$\text{sig}(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

$$\text{ReLU}(x) = \max(0, x) \tag{2.6}$$



(a) Sigmoidfunktion **(b)** ReLU-Funktion

Abbildung 2.6: Beispiele für Aktivierungsfunktionen

2.2.2 Vom Linearfilter zu Faltungsnetzwerken

Das zu analysierende Signal (siehe Abb. 2.3) ist ein zeitdiskretes, wertkontinuierliches Signal. Für einen Linearfilter ist eine Folge von Koeffizienten gegeben. Sei das Signal x und $[y_0, y_m]$ die Linearfilterkoeffizienten, dann kann in Gl. 2.7 die diskrete Faltung

beschrieben werden.

$$(x * y)[n] = x'[n] = \sum_{i=0}^{m} y_i \cdot x[n-i] \tag{2.7}$$

Das neue Signal x' hat damit die gleiche Dimension wie das Eingangssignal x. Der einzelne Linearfilter kann auch als *Kernel* bezeichnet werden. [11, S. 331 - 333]
Diese Idee der Faltung kann in neuronalen Netzwerken benutzt werden. Grundlegende neuronale Netze beruhen auf der Verbindung jeder Eingabe mit jeder Ausgabe. Durch Benutzung solcher Faltungen wird es möglich, die Größe von neuronalen Netzwerken signifikant zu verringern sowie mit Eingaben unterschiedlicher Größe zu arbeiten. Eine Möglichkeit, diese Vorteile zu benutzen, sind **Convolutional Neural Networks** (CNN, Faltungsnetzwerke). Anstatt von einfachen Multiplikationen werden an den Knoten Faltungen durchgeführt. Das ist gerade sinnvoll bei der Suche nach lokalen Phänomenen (z.B. ein Puls). Die Eingabe liegt dabei als Matrix vor, wobei schrittweise die Filtermatrix darüber bewegt wird. Im eindimensionalen Fall entspricht die Filtermatrix einem Vektor, wie bei einem Linearfilter. [11, S.330ff]
Bereiche können parallel von verschiedenen Filtermatrizen ausgelesen werden, wodurch komplexe Analysen möglich werden. Die Anzahl von Kernel pro Schicht geben auch die Dimension des herausfließenden Datenstroms an. Um die Dimension von Eingabe und Ausgabe gleich zu halten, ist es wichtig, dass die letzte Schicht nur aus einem Kernel besteht. Die Ausgabe solcher Kernel wird auch als "Feature Map"bezeichnet.

Die CNNs werden mit Hilfe von Keras [14] definiert, trainiert und ausgewertet. Jedoch ist für den Einsatz am ATLAS-Detektor eine technische Lösung notwendig, die in der Back-End-Elektronik benutzt werden kann. Diese Lösung ist mittels logischen Schaltungen, den FPGAs (Field Programmable Gate Array) geplant. FPGAs besitzen die Kapazitäten mit der hohen Menge an Eingabedaten umzugehen. Jedoch haben sie begrenzte Ressourcen im Bezug auf Logik und Speicher. Dadurch wird die Anzahl und Art der mathematischen Operationen, die durchgeführt werden können, eingeschränkt. Für die softwareseitige Entwicklung der CNNs bedeutet das die Einschränkung der Gesamtparameteranzahl, damit diese konform mit den FPGAs ist. [9]
Ein beispielhafter Aufbau eines solchen CNNs zur Energierekonstruktion ist in Abb. 2.7 dargestellt. Die Eingabe ist ein zeitdiskretes Signal. Kernel bezeichnet nicht die Anzahl derer, sondern die Größe der Filtermatrix. In der ersten Schicht verarbeiten 5 Filter mit Filtergröße 3 parallel die Eingabe. Die zweite Schicht besteht aus einem Fil-

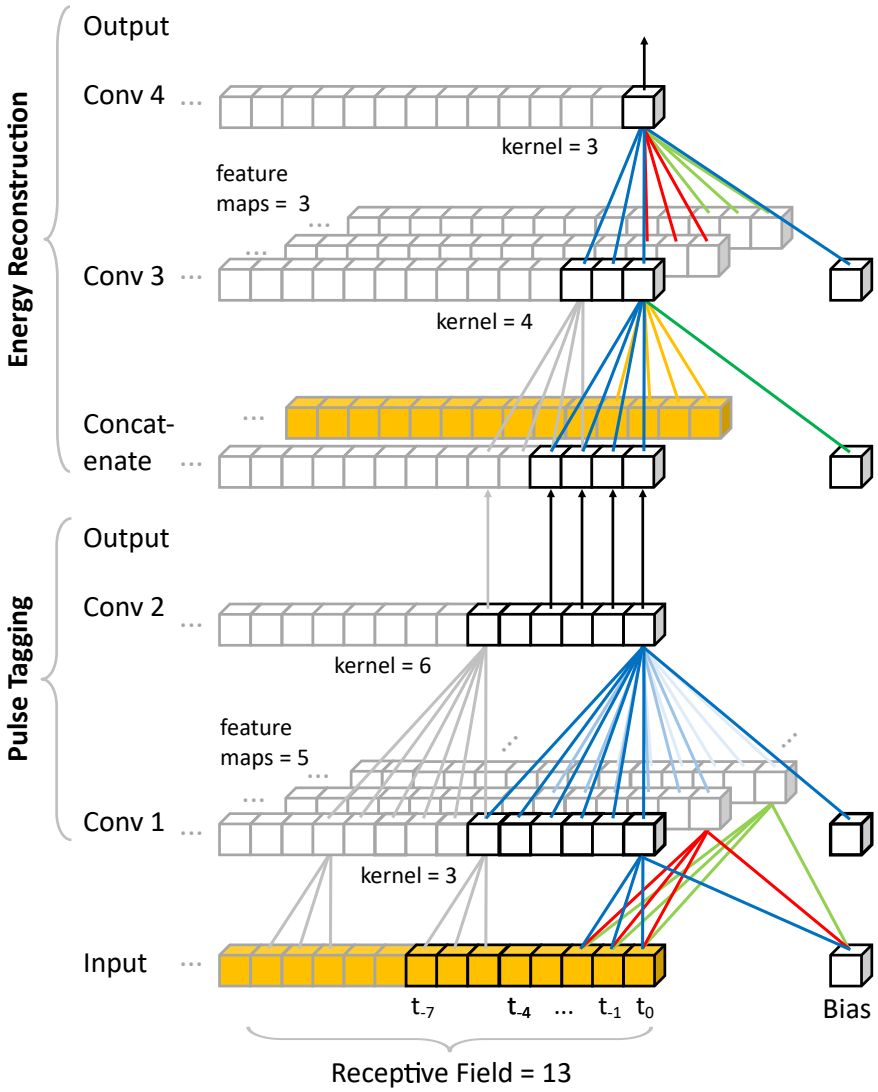


Abbildung 2.7: Aufbau eines Convolutional Networks [9]

ter mit der Filtergröße 6. So wird bis zur zweiten Schicht eine Überdeckungsreichweite (Receptive Field) von 8 Werten erreicht. Im Training werden diese beiden Schichten als erstes an eine sogenannte Taggingausgabe trainiert. Diese Taggingausgabe ist eine binäre Datensequenz bei der Signalen über 3σ-Abweichung über dem elektronischen Rauschen, was 240 MeV entspricht, eine 1 zugeordnet wird. [9]
In einer zweiten Trainingsstufe wird dieses Netzwerk an die Energieausgabe trainiert, wobei die Taggingausgabe mit der originalen Eingabe verkettet wurde und als Eingabedaten für die 3. Schicht verwendet wird. Hier wird unterschieden zwischen **Conv3**- und **Conv4**-Netzwerken. Im Conv3-Netzwerk folgt nach dem Tagging eine Schicht, im Conv4 zwei Schichten.[9]

Das Training fand an Datensequenzen für die mittlere Schicht des EMB statt. Eine Beispielsequenz mit Rekonstruktion ist in Abb. 2.8 zu erkennen.

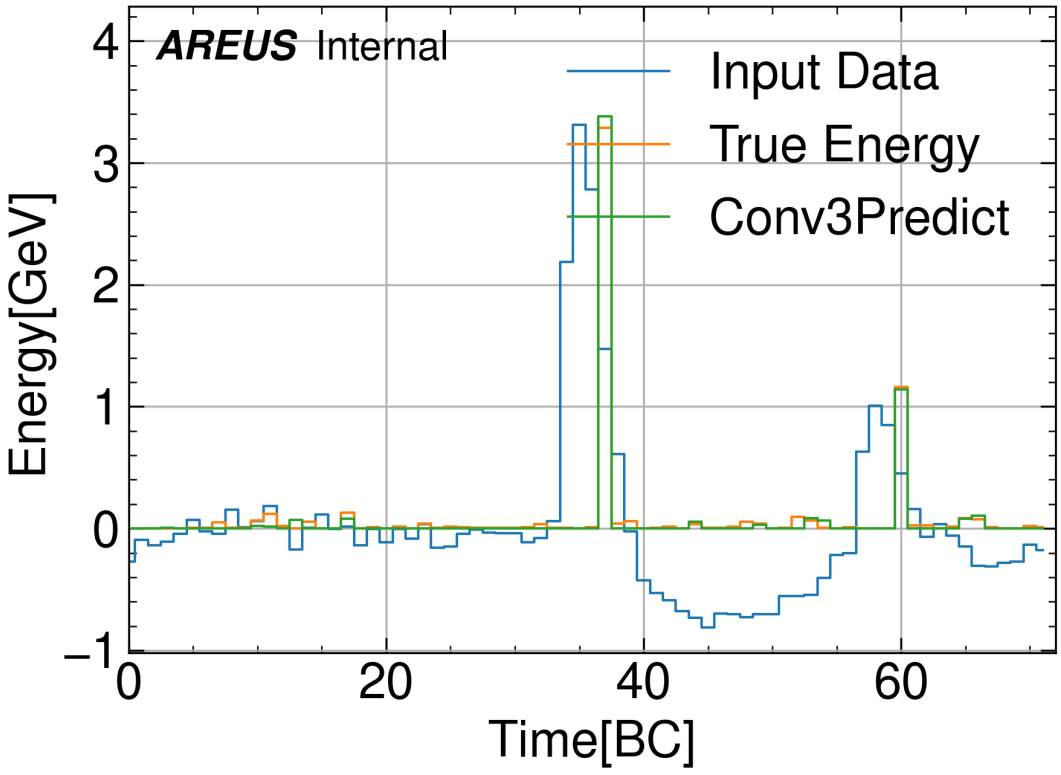


Abbildung 2.8: Eingabedatensequenz mit umgeformten, digitalisierten Pulsen mit den entsprechenden richtigen Energiewerten (TrueEnergy). Conv3Predict ist die Vorhersage des **Conv3**-Netzwerks ausgehend von der Eingabe.

3 Hyperparameteranalyse der CNNs zur Energierekonstruktion

3.1 Hyperparameter

Hyperparameter bezeichnen strukturelle Parameter der Modelle sowie Trainingsparameter. Hyperparameter werden nicht trainiert, sondern müssen vor dem Training festgelegt werden. Im Gegensatz dazu werden die anderen Parameter des Netzes, die Gewichte, während des Trainings verändert. Für CNNs kann zwischen Modellstrukturparametern und Trainingsparametern unterschieden werden. [15]
Zu den Strukturparametern gehören alle Parameter, die den Aufbau des Modells beschreiben. Dies umfasst z.B. Anzahl von Schichten, Anzahl von Filtern pro Schicht und Größe der Filter in der Schicht. Trainingsparameter hingegen sind z.B. Wahl der Verlustfunktion oder Wahl der Gewichtsinitialisierung. Beide Arten können mit Hilfe von sogenannten Hyperparameteroptimierungen gewählt werden. Die Wahl solcher Hyperparameter hat einen großen Einfluss auf die Leistung eines Netzwerks. [15]

3.2 Hyperparameteroptimierung

Ähnlich zu dem grundlegenden Ziel des Maschinellen Lernens; ein Modell zu trainieren, damit eine Verlustfunktion minimiert wird, verhält sich die Zielsetzung der Hyperparameteroptimierung. Es wird ein Hypermodell gebaut, das in Abhängigkeit bestimmter Hyperparameter neuronale Netzwerke bildet. Unter Wahl der richtigen Hyperparameter soll eine Funktion, "objective"genannt, je nach Problemstellung maximiert oder minimiert werden. Eine typische Wahl ein solchen Ziels ist die Verlustfunktion der einzelnen Modelle. Danach wird das beste Modell so ausgewählt, dass dieses den geringsten Wert der Verlustfunktion auf den Testdaten hat. Darüber hinaus können auch andere Ziele, unabhängig von der Verlustfunktion der einzelnen Modelle gewählt werden.

Zum Aufbau eines Hypermodells wird der Keras-Tuner benutzt. [16] Dieser benötigt eine Bauvorschrift von neuronalen Netzen in Abhängigkeit von der ausgewählten Hyperparametern. Daraus wird ein sogenannter **Search Space**, der Suchraum, der alle möglichen Hyperparameterkombinationen enthält. Da jeder zusätzlicher Parameter schnell diesen Raum vergrößert, unterliegt die Hyperparametersuche dem sogenannten "curse of dimensionality"[17]. Während zwei Hyperparameter mit je 10 Möglichkeiten einen Suchraum von der Größe $10^2 = 100$ aufbauen, sorgen schon 3 weitere Hyperparameter für eine Größe von $10^5 = 10000$. Um trotzdem optimale Hyperparameterkombinationen zu wählen, gibt es verschiedene Optimierungsalgorithmen:

- **Gittersuche:** Das ist der ursprüngliche Weg der Hyperparameteroptimierung. Alle Kombinationen des Suchraums werden nacheinander ausprobiert und am Ende wird das Optimum dem Ziel folgend ausgewählt. Dieser Algorithmus unterliegt damit sehr dem Problem der Dimensionalität und bietet für die wenigsten Anwendungen die richtige Lösung. Bei großen Rechenkapazitäten ist allerdings der Algorithmus leicht zu parallelisieren und dadurch zu beschleunigen. [18]

- **Zufallssuche:** Dieser Algorithmus ähnelt der Gittersuche, aber anstatt nacheinander alle möglichen Kombinationen auszuprobieren, werde diese zufällig ausgewählt. In vielen Problemen schneidet die Zufallssuche besser als die Gittersuche ab [18], vor allem wenn viele Hyperparameter geringen Einfluss auf die Performance haben.

- **Bayes-Suche:** Hierbei wird ein probabilistisches Modell für das Minimum des Ziels $f(X)$ mit den Hyperparametern X entwickelt. Dieses Modell wird erst mit zufällig gewählten Stützpunkten aufgebaut, um dann damit die Entscheidung zu treffen, welches X als nächstes zu überprüfen sei. Dabei wird ein Gleichgewicht zwischen Exploration (Hyperparameter in unsicheren Gebieten) und Exploitation (Hyperparameter nahe des erwarteten Optimum) gewählt. [19]

- **Hyperband-Algorithmus:** Dieser Algorithmus nutzt eine verbesserte Variante der Zufallssuche. Das grundlegende Prinzip beinhaltet die fortlaufende Halbierung unter der Annahme einer festen Rechenkapazität B. Es werden eine feste Zahl n an Modellen mit zufällig verschiedenen Hyperparametern initialisiert und m Iterationen (m ist abhängig von B und n) trainiert. Danach werden die besten 50 % der Modelle behalten und weitere m Iterationen trainiert. Dieser Prozess wird wiederholt, bis nur noch ein Modell übrig ist. Hyperband nutzt dieses Prin-

zip durch Iteration über verschiedene n und wählt danach das beste Modell aus. [20]

3.3 Hyperparameter: Standardabweichung der Initialparameter

Das Optimum der Netzwerke zur Energierekonstruktion ist bisher das **Conv3**-Netzwerk [9]. Ein Problem ist jedoch die geringe Erfolgsquote der Trainings. Aus 100 initialisierten Netzwerken lernen nach 600 Epochen nur rund 10 Netze gute Energierekonstruktion, während nur eins sehr gute Ergebnisse zeigt. Das folgt aus der Abhängigkeit kleiner Netzwerke (Netzwerke, die wenig trainierbare Parameter besitzen) von den Initialparametern. Diese werden mit Hilfe des Kernelinitializer gewählt, eine Funktion, die die Startgewichte zufällig bestimmt. Meist wird eine Gaußverteilung um 0 mit der Standardabweichung σ gewählt. Dieses σ kann als Hyperparameter betrachtet werden, dessen Wahl einen Einfluss auf die Trainingserfolge hat.
Wenn man die Modellstruktur gleich lässt und nur die Standardabweichung als Hyperparameter wählt, ist der Suchraum lediglich eindimensional. Andererseits ist $\sigma \in \mathbb{R}^+$, also benötigt es eine Diskretisierung des Suchraums. Diese Diskretisierung wurde logarithmisch skaliert im Keras-Tuner. Um das Problem einfach zu halten, wurden nur Netzwerke zur Tagging-Ausgabe 200 Epochen trainiert. Für das Tagging-Problem wird die binäre Kreuzentropie (Gl. 3.1) als Verlustfunktion gewählt, die für jeden Modellausgabewert y_i und korrekten Wert p_i den Verlustwert berechnet. Relevant für das Training ist dann der Mittelwert über die gesamte Datenreihe.

$$\mathrm{BCr}_i(y_i, p_i) = -y_i \cdot \log p_i - (1 - y_i) \cdot \log (1 - p_i) \tag{3.1}$$

Am Ende des Trainings wird die Verlustfunktion für das jeweilige Modell berechnet. Der angewandte Suchalgorithmus ist die Zufallssuche.
In Abb. 3.1 sind die Werte der Verlustfunktion für verschiedene Standardabweichungen logarithmisch aufgetragen. Das beste Netzwerk lag bei einer Standardabweichung von 3.5 vor. Während bei Standardabweichungen kleiner 1 die Variation ähnlich groß ist, kann man danach eine größere Schwankung erkennen. Das beste Ergebnis taucht bei einer Standardabweichung von 4.6301 auf. Diese Extremalstellen sind jedoch nicht skaleninvariant, sondern spezifisch für das Problem. Der Größenbereich der Netzwerk-

parameter hängt vom Wertebereich der Eingabedaten sowie der Aktivierungsfunktion, die hier die Sigmoidfunktion ist, ab.

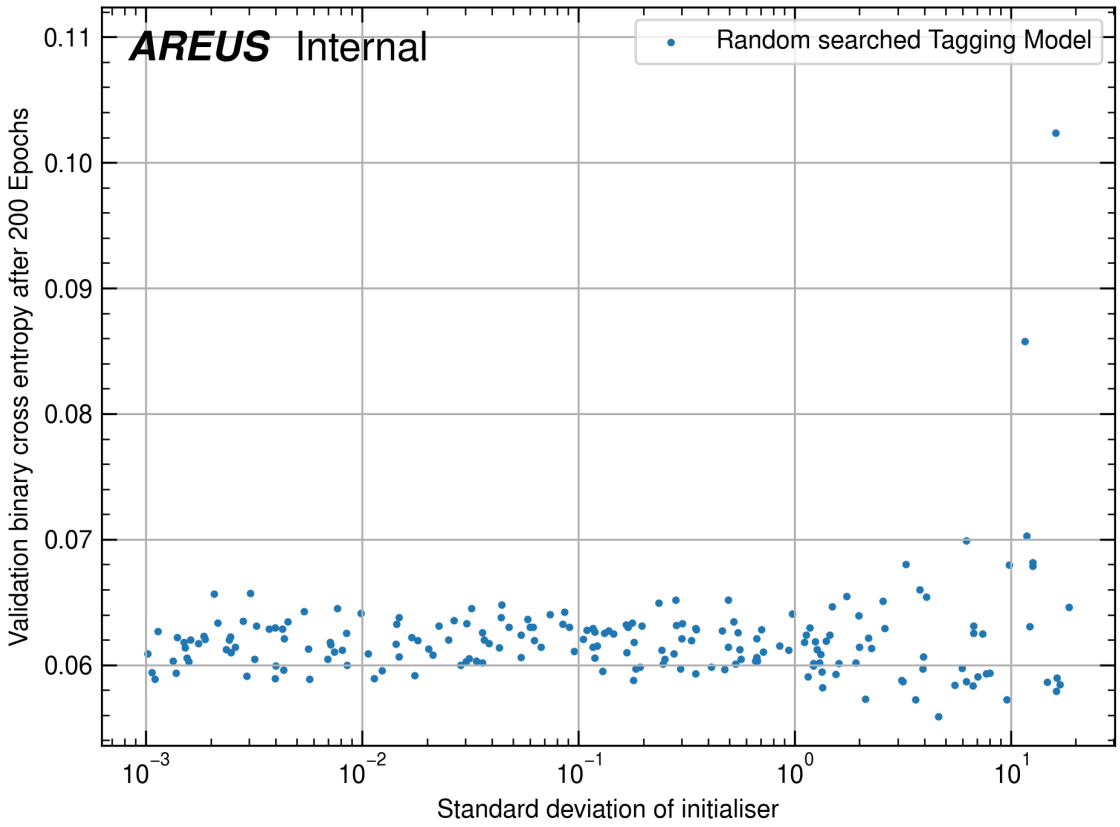


Abbildung 3.1: Streudiagramm der Hyperparametersuche für die Standardabweichung der Initialisers. Die Standardabweichung ist logarithmisch skaliert. Die Verlustfunktion ist die binäre Kreuzentropie.

Dieses Verhalten folgt der Natur der Standardabweichung. Nehmen wir an, dass es eine optimale Parameterwahl gibt und umso größer der Abstand der Initialparameter zu dieser ist, umso höher ist der Wert der Verlustfunktion nach 200 Epochen. Bei kleinen Standardabweichungen ist der Weg zur optimalen Parameterwahl immer ähnlich weit. Das erklärt die geringe Variation. Wenn die Standardabweichung erhöht wird, kann durch Zufall die Initialwahl sehr nahe an einem Optimum liegen, was den besten Wert bei hohen Standardabweichungen erklärt. Jedoch erhöht sich die Gefahr, dass die Wahl der Initialparameter so schlecht ist, dass das Training nicht gut konvergiert, zu sehen in den Ausreißern bei der Standardabweichung über 10.

3.4 Hyperparameter: Modellaufbau

3.4.1 Motivation

Die bisherige Forschung hat gezeigt, dass bestimmte Aufbauten der CNNs die beste Leistung aufzeigen, diese sind die zweistufigen **Conv3**-Modelle [9]. Die wichtigste Einschränkung des Modellaufbaus ist ihre Anzahl von trainierten Parametern, die ungefähr der Anzahl der benötigten Rechenschritte beim Modellausführen beschreibt. Durch den geplanten Einsatz auf FPGAs sowie der geplanten schnellen Berechnung wird hier ein Wert von ungefähr 100 angesetzt. Aber selbst mit nur 100 zur Verfügung stehenden Parametern gibt es trotzdem eine Vielzahl an möglichen Modellstrukturen, von vielen kleinen Schichten zu einzelnen, sehr großen Schichten, die nur mit sehr großer Rechenleistung alle ausprobiert werden können.
Der Modellaufbau kann als Hyperparameterproblem formuliert werden. Dadurch können Optimierungsalgorithmen angewandt werden, die das Problem lösbar machen.

3.4.2 Parameterberechnung

Die Parameter eines eindimensionalen CNNs sind durch die Inputdimensionen und Filteranzahl sowie Kernelgrößen der Layer fest definiert. Pro Layer gibt es die feste Parameterzahl N_i mit Layernummer i (Gl. 3.2), die sich aus der Filteranzahl des vorherigen Layers n_{i-1}, der Filteranzahl im Layer n_i und der Kernelgröße im Layer k_i zusammensetzt. [21]

$$N_i = n_i \cdot (1 + n_{i-1} k_i) \tag{3.2}$$

Die Gesamtanzahl an Parameter in einem Netzwerk folgt daraus als Summe über alle Layer mit der Zahl der Inputs n_0. Für ein Netzwerk mit m Layern folgt die Gesamtanzahl der Parameter in Gl. 3.3.

$$N_{\text{ges}} = \sum_{i=1}^{m} N_i = \sum_{i=1}^{m} n_i \cdot (1 + n_{i-1} k_i) \tag{3.3}$$

Für die Hyperparameteranalyse der Kernelgrößen und Filteranzahlen für die Netzwerke ist es nützlich eine Beschränkung der Gesamtanzahl der Parameter zu geben, während mögliche Kombinationen der beiden Hyperparameter ausprobiert werden. Mit einem festen N_{ges} gibt es für jede Kernelgröße und Filteranzahl eigene Maximal- und Minimalwerte, wobei ganze Zahlen zwischen beiden Werte ausgewählt werden

können. Für diese Berechnung braucht man die Kernelgröße n_j (Gl. 3.4) und Filteranzahl k_j (Gl. 3.5) in Abhängigkeit von den anderen Größen.

$$n_j = \frac{N_{\text{ges}} - \left(\sum_{i=1}^{j-1} n_i(1 + n_{i-1}k_i)\right) - \left(\sum_{i=j+2}^{m} n_i(1 + n_{i-1}k_i)\right) - n_{j+1}}{1 + n_{j-1}k_j + n_{j+1}k_{j+1}} \tag{3.4}$$

$$k_j = \frac{N_{\text{ges}} - \left(\sum_{i=1}^{j-1} n_i(1 + n_{i-1}k_i)\right) - \left(\sum_{i=j+1}^{m} n_i(1 + n_{i-1}k_i)\right) - n_j}{n_j n_{j-1}} \tag{3.5}$$

Da beide Werte natürliche Zahlen sein müssen, wird im Programm abgerundet. Der Minimalwert für jeden möglichen Parameter ist 1. In den Gln. 3.5, 3.4 sieht man für die Parameter zwei unterschiedliche Summen im Zähler, einmal die Summe der vorherigen Parameter, einmal die Summe der folgenden. Ordnet man die n_j, k_j nach ihren Indizes sowie n vor k, kann man den Maximalwert mit zwei Bedingungen definieren:

1. Alle vorherigen Werte sind schon festgelegt.
2. Alle nachfolgenden Werte sind auf 1 gesetzt.

So kann man iterativ bei n_0 starten, für jeden Parameter Minimal- und Maximalwert festlegen und einen Wert im möglichen Bereich auswählen und dann zum nächsten Parameter übergehen.
Folgt man diesen Schritten kann $n_{j,\max}$ für $j < m$ berechnet werden. Der obere Grenzfall wird extra betrachtet.

$$n_{j,\max} = \frac{N_{\text{ges}} - \left(\sum_{i=1}^{j-1} n_i(1 + n_{i-1}k_i)\right) - \left(\sum_{i=j+2}^{m} 2\right) - 1}{1 + n_{j-1} + 1} \tag{3.6}$$

$$n_{j,\max} = \frac{N_{\text{ges}} - \left(\sum_{i=1}^{j-1} n_i(1 + n_{i-1}k_i)\right) - 2(m - j - 1) - 1}{2 + n_{j-1}} \tag{3.7}$$

Für $j = m$ gibt es keine nachfolgenden n_{m+1}. Daraus folgt:

$$n_{m,\max} = \frac{N_{\text{ges}} - \left(\sum_{i=1}^{m-1} n_i(1 + n_{i-1}k_i)\right)}{1 + n_{m-1}} \tag{3.8}$$

Äquivalent dazu kann diesselbe Rechnung für $k_{j,\max}$ durchgeführt werden, n_j ist festgelegt. Es gilt wieder $j < m$.

$$k_{j,max} = \frac{N_{\text{ges}} - \left(\sum_{i=1}^{j-1} n_i(1 + n_{i-1}k_i)\right) - \left(\sum_{i=j+1}^{m} n_i(1 + n_{i-1}k_i)\right) - n_j}{n_j n_{j-1}} \tag{3.9}$$

$$k_{j,max} = \frac{N_{\text{ges}} - \left(\sum_{i=1}^{j-1} n_i(1 + n_{i-1}k_i)\right) - 1 - n_j - \left(\sum_{i=j+2}^{m} 2\right) - n_j}{n_j n_{j-1}} \tag{3.10}$$

$$k_{j,max} = \frac{N_{\text{ges}} - \left(\sum_{i=1}^{j-1} n_i(1 + n_{i-1}k_i)\right) - 2(m - j - 1) - 2n_j - 1}{n_j n_{j-1}} \tag{3.11}$$

Für $j = m$ entfällt der $2(m - j - 1)$ Term.

$$k_{m,max} = \frac{N_{\text{ges}} - \left(\sum_{i=1}^{j-1} n_i(1 + n_{i-1}k_i)\right) - n_j}{n_j n_{j-1}} \tag{3.12}$$

Da Hyperparameter mit festen Ober- und Untergrenzen definiert werden müssen, muss das Festlegen des Parameterwerts parametrisiert werden. Diese Parametrisierungen werden mit den Parametern $a_j, b_j \in [0, 1]$ ausgedrückt. So können im Hypermodel $2j$ Parametrisierungen festgelegt werden, aus denen im Nachhinein die genauen Filteranzahlen und Kernelgrößen berechnet werden.

$$n_j = (1 - a_j) + a_j n_{j,\max} \tag{3.13}$$

$$k_j = (1 - b_j) + b_j k_{j,\max} \tag{3.14}$$

Eine Implentierung dieser Berechnung ist in der Funktion parameter_calc zu finden. Die Filteranzahl im letzten Layer ist eigentlich immer vorgeschrieben und meistens 1. Für den letzten möglichen Parameter k_m kann auch gleich der Maximalwert genommen werden, sodass die Gesamtanzahl möglichst nah erreicht wird.
Dort übergibt man einer Funktion eine Liste von $m-1$ Tupeln (a_j, b_j), die Gesamtanzahl an Parametern $N_{\{ges}$ sowie die Inputzahl n_0 und die Filteranzahl im letzten Layer n_m und bekommt eine Liste mit m Filteranzahl und Kernelgrößen (n_j, k_j) zurück.

```
def parameter_calc(para_list, n_ges, n_0, n_m):
    m = len(para_list)
    res = []
    used_parameter = 0
    n_j_1 = n_0
    for j in range(m - 1):
        a_j = para_list[j][0]
        b_j = para_list[j][1]
        n_j = int(1 - a_j + a_j * ((n_ges - used_parameter - 2 * (m -
                                    j - 1) - 1)/(2 + n_j_1)))
        k_j = int(1 - b_j + b_j * ((n_ges - used_parameter - 2 * (m -
                                    j - 1) - 2 * n_j - 1)/(
                                   n_j * n_j_1)))
        res.append((n_j, k_j))
        used_parameter += n_j * (1 + n_j_1 * k_j)
        n_j_1 = n_j
    k_m = int((N - used_parameter - n_m)/(n_m * n_j_1))
    res.append((n_m, k_m))
    return res
```

4 Bunch-Train-Struktur

4.1 Bunch-Train-Performance von CNNs

4.1.1 Pileup-Problem

Für das EMB sind interessante Endprodukte bei Proton-Proton-Kollisionen Photonen und Elektronen. Jedoch sind Standardmodell-Prozesse mit Photonen und Elektronen seltener als inelastische *pp*-Streuung mit Pionenproduktion. Diese Pionen sind die Quelle eines niedrigenergetischen Hintergrunds. [22]
Würden durchgehend Proton-Proton-Kollisionen stattfinden, wäre durch die bipolare Pulsform ein grundlegender Mittelwert der Messdaten, die sogenannte Baseline (Grundlinie), bei 0 gegeben. Im Betrieb des LHC ist ein kontinuierlicher Betrieb mit Proton-Proton-Kollisionen jedoch nicht möglich. Stattdessen werden Protonen getaktet in Paketen eingefügt, auf die Lücken folgen. Diese Paketzugstruktur ist periodisch und wird Bunch-Train (BT) genannt. Das sorgt dafür, dass sich am Anfang eines Pakets viele positiven Spitzen der bipolaren Pulse des niederenergetischen Hintergrunds überlagern und durch das Fehlen der vorangegangen negativen Schweife eine Verschiebung der Baseline hervorgerufen wird. Nach einem Paket gibt es dann wiederum eine Überlagerung vieler negativer Schweife, die die Baseline ins Negative verschieben. (Abb. 4.1)

4.1.2 Analysedatensatz

Bisher wurden die Netze auf Datensätzen trainiert, die keine BT-Struktur aufzeigen. Um deren Performance bei fester BT-Struktur zu bewerten, benötigt es einen neuen Datensatz. Eine typische BT-Struktur ist das 25 ns Scheme (Abb. 4.2) [24]. Dort werden 3 oder 4 Pakete injiziert, die 72 BC breit sind und untereinander eine 8er Lücke haben. Nach den 3 oder 4 Paketen folgt eine 30 BC große Lücke.

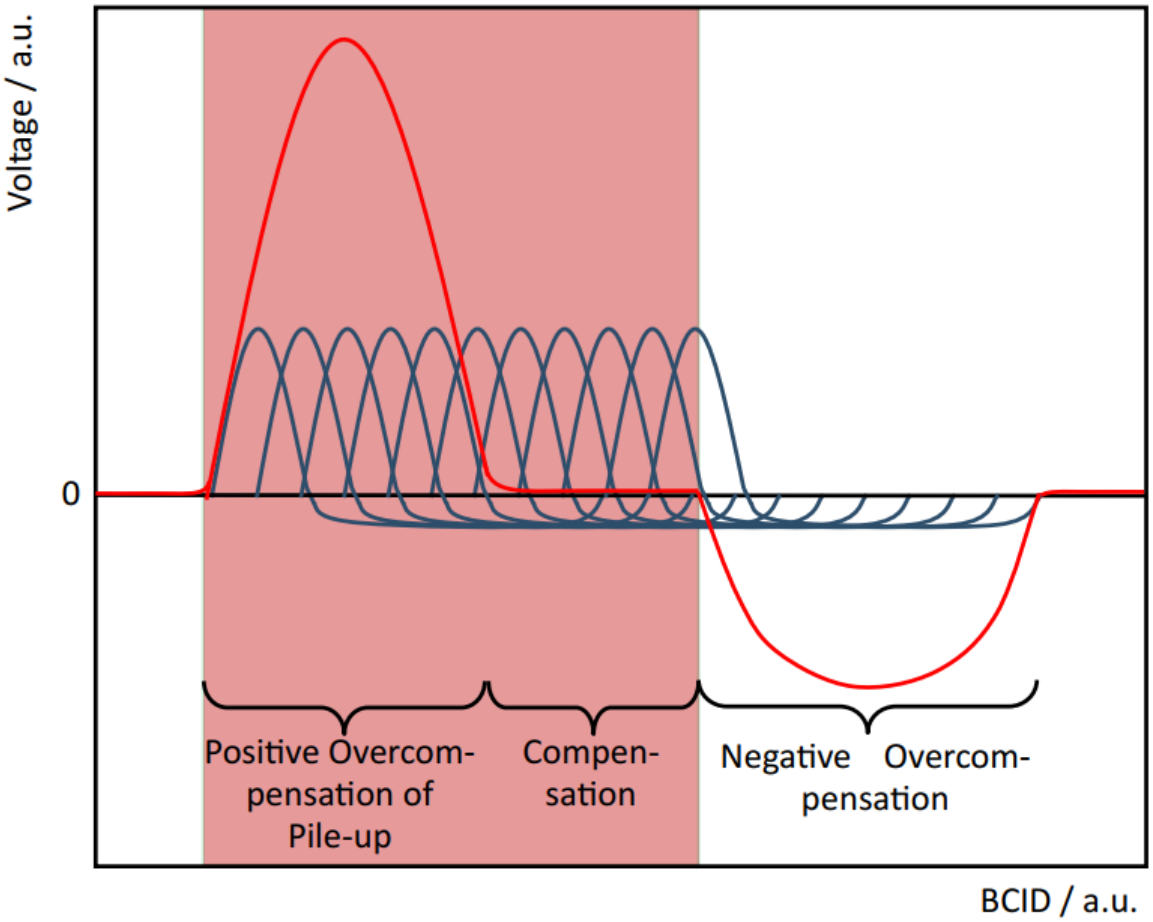


Abbildung 4.1: Darstellung der Baselineentstehung. Im roten Bereich tauchen Pulse auf. Die Summe der vielen einzelnen Signale in Blau sorgt dafür, dass das Gesamtsignal (orange) sich verschiebt. [23]

Damit solche BT-Strukturen leichter zu beschreiben sind, wird eine spezielle Notation benutzt, die auf der Anzahl von gefüllten oder leeren BCs beruht. Ein Paket der Länge n mit darauffolgender Lücke der Länge m wird als $nfme$ bezeichnet, während f für full und e für empty steht. In dieser Form wird ein ganzes Muster aufgeschlüsselt, während Wiederholungen zusammengefasst werden (8f8e8f8e=2(8f8e)).
Um das 25 ns Scheme zu reproduzieren, aber auch zu vereinfachen, wird eine 3(72f8e)72f30e-Struktur benutzt. Die 8er Lücken und die große Lücke bleiben erhalten, aber die abwechselnde Paketstruktur wird vernachlässigt. Dieser Datensatz wurde in der AREUS Simulation [25] erstellt, in der die Auslesekette von der analogen Elektronik des Kalorimeters bis hin zur Back-End-Elektronik simuliert wird. Für den Datensatz werden die mittleren Zellen der Flüssigargonkalorimeter benutzt.

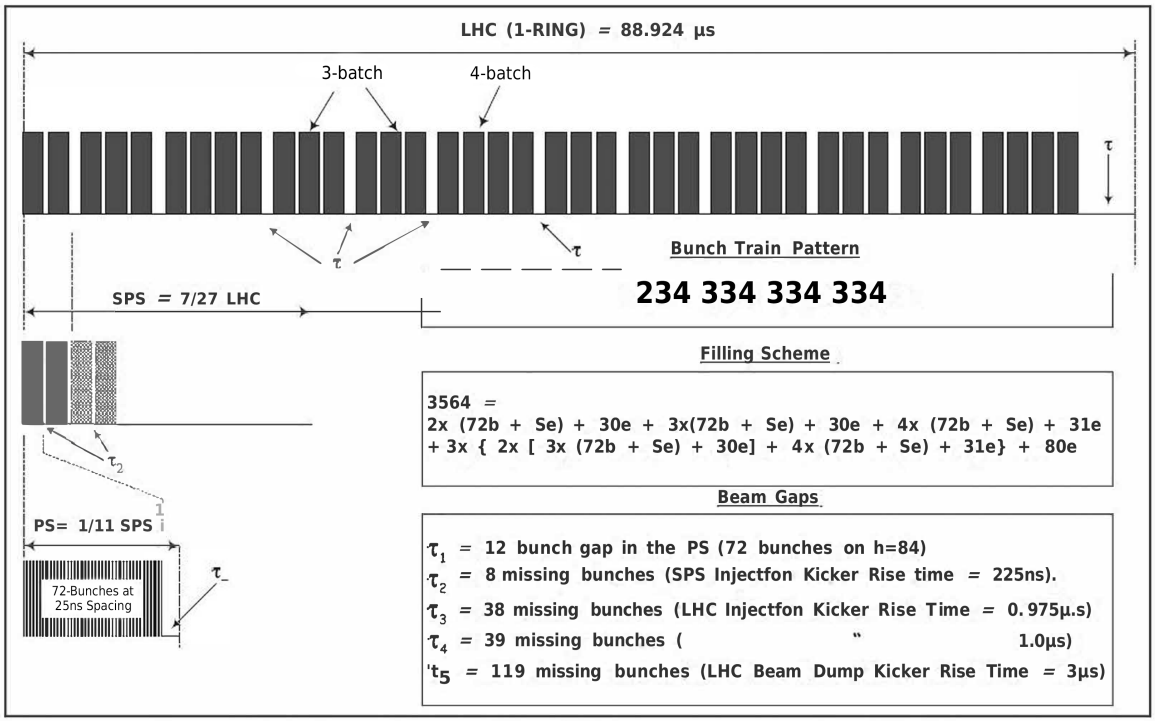


Abbildung 4.2: Standard-BT-Struktur am LHC im momentanen Betrieb [24]

Die Maximalhöhe der Energien sind 5 GeV in der Simulation. Auf einen konstant variierenden Pionenhintergrund werden mit einem zufälligen Abstand mit Mittelwert 30 BC und einer Standardabweichung von 10 BC höhere Peaks bis zu 5 GeV aufaddiert. Elektronisches Rauschen wird mitsimuliert.

4.1.3 Baseline-Berechnung

Um die Baseline des Hintergrunds zu berechnen, wird ein Datensatz ohne künstliche Treffer, sondern ausschließlich mit Pionenevents generiert. Dieser hat $2 \cdot 10^6$ Einträge. Der Datensatz wird anschließend entsprechend der Länge eines Bunch-Trains umgeformt und über alle Werte für jede periodische BT-Position der Mittelwert genommen. So erhält man einen neuen Datensatz in der Länge der BT-Struktur mit den mittleren Energie für jede Position. Die Baseline des Analysedatensatz ist in Abb. 4.3 dargestellt. Hier kann man den Einfluss der unterschiedlichen gut Lückengrößen erkennen. Für die Interpretation ist wichtig, dass die Baseline sich zyklisch verhält, also das Ende dem neuen Paket vorangeht. Während die Pakete immer gleich lang sind, ändern sich nur die Lückengrößen.

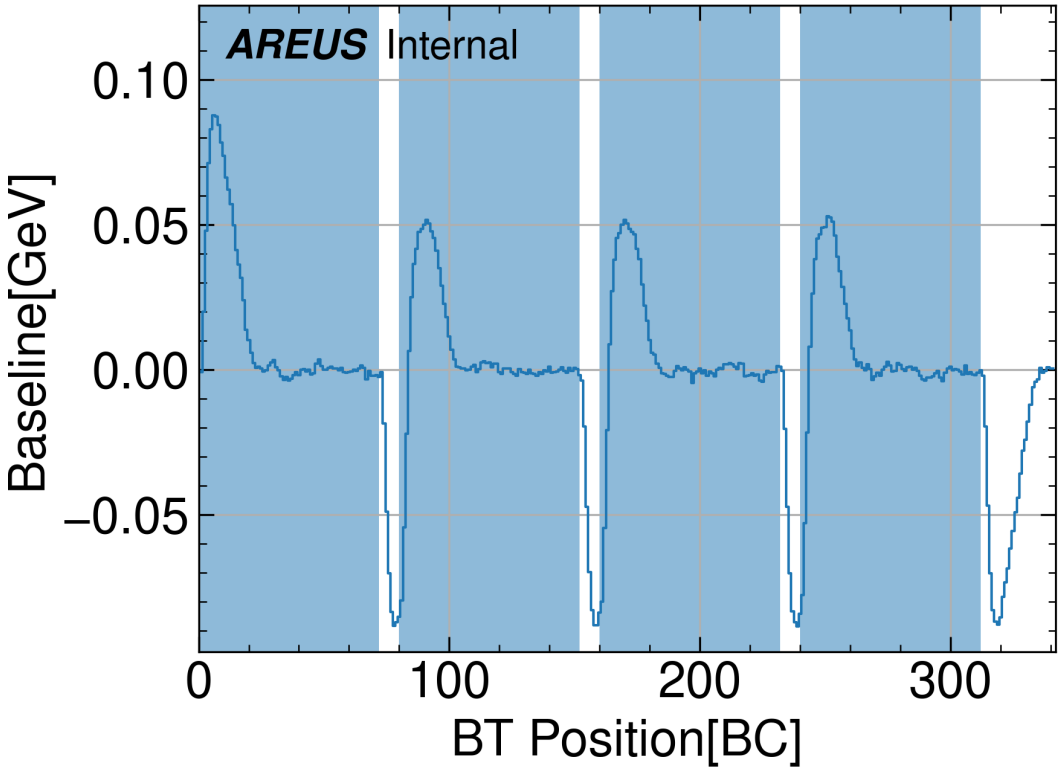


Abbildung 4.3: Baseline des Analysedatensatzes. Blaue Bereiche zeigen Pakete, weiße Lücken an.

Hier kann man den Einfluss der unterschiedlichen gut Lückengrößen erkennen. Für die Interpretation ist wichtig, dass die Baseline sich zyklisch verhält, also das Ende dem neuen Paket vorangeht. Während die Pakete immer gleich lang sind, ändern sich nur die Lückengrößen. Für das erste Paket ist durch die vorangegangene große Lücke die Baseline fast doppelt so hoch im Vergleich zu den anderen Paketen. Auch kann man ein längeres Unterschwingen in der letzten Lücke ab 312 BC erkennen. Die Dauer dieses Unterschwingens entspricht ca. 30 BC, ähnlich beim Überschwingen im ersten Paket. Bei den kurzen Lücken ist das Unterschwingen am Ende der Lücke noch nicht beendet, so dass die Baseline am Anfang der hinteren Pakete erst vom Negativen ins Positive wechselt. Diese 30 BC unterschwingen entsprechen ungefähr der Länge eines umgeformten Pulses, der eine Länge von 33 BC hat.

4.1.4 Einfluss auf die Energierekonstruktion

Die Netzwerke, die momentan zum Einsatz der Energierekonstruktion geplant sind, sind 3-schichtige CNNs, die ohne Beachtung der BT-Struktur trainiert wurden. Eine Möglichkeit, die Einflüsse der BT-Struktur zu verringern, ist die zuvor berechnete Baseline von den Eingabedaten zu subtrahieren, so dass etwaige Abweichung korrigiert werden.
Zwei Aspekte, die dabei betrachtet werden können sind zum Einen die Signalrekonstruktion in den Lücken und zum Anderen die Abweichung der Rekonstruktion in den Paketen. Zu diesem Zweck kann man zweidimensionale Histogramme, auch Heatmap genannt, einsetzen. Die durchschnittliche Performance gemittelt über die BT-Länge kann darin betrachtet werden. In Abb. 4.4 ist die Heatmap des Optimalfilters für Energierekonstruktion dargestellt, ohne dass eine Baselinekorrektur durchgeführt wurde.

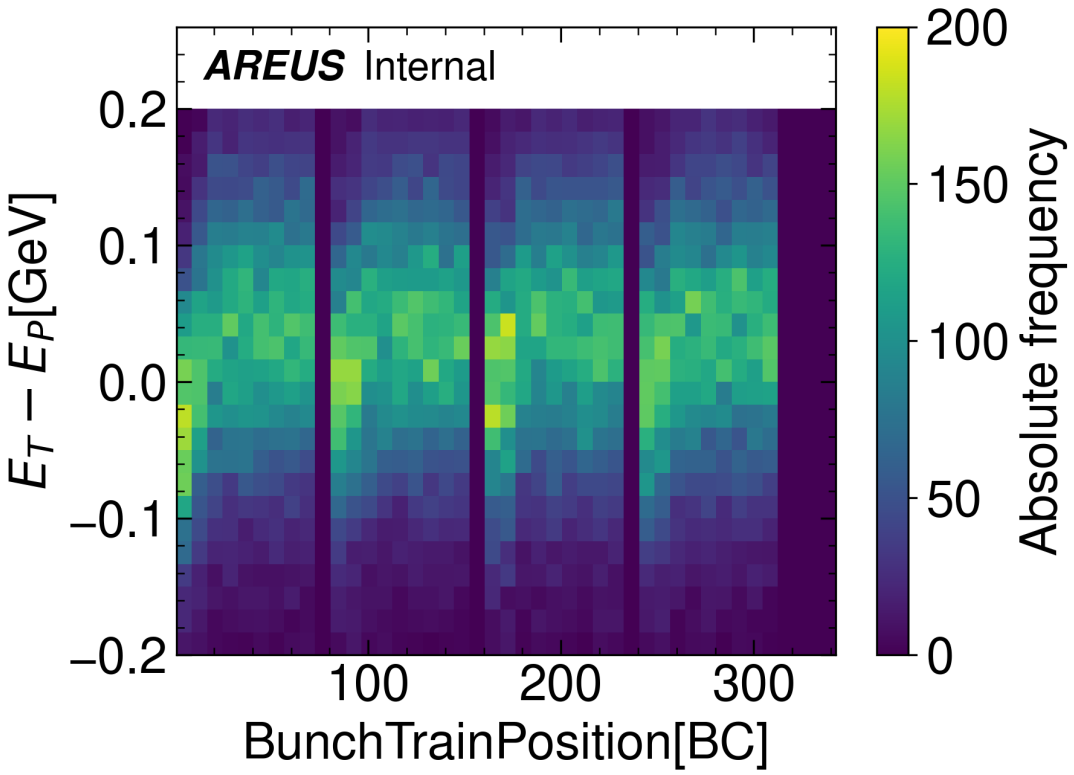
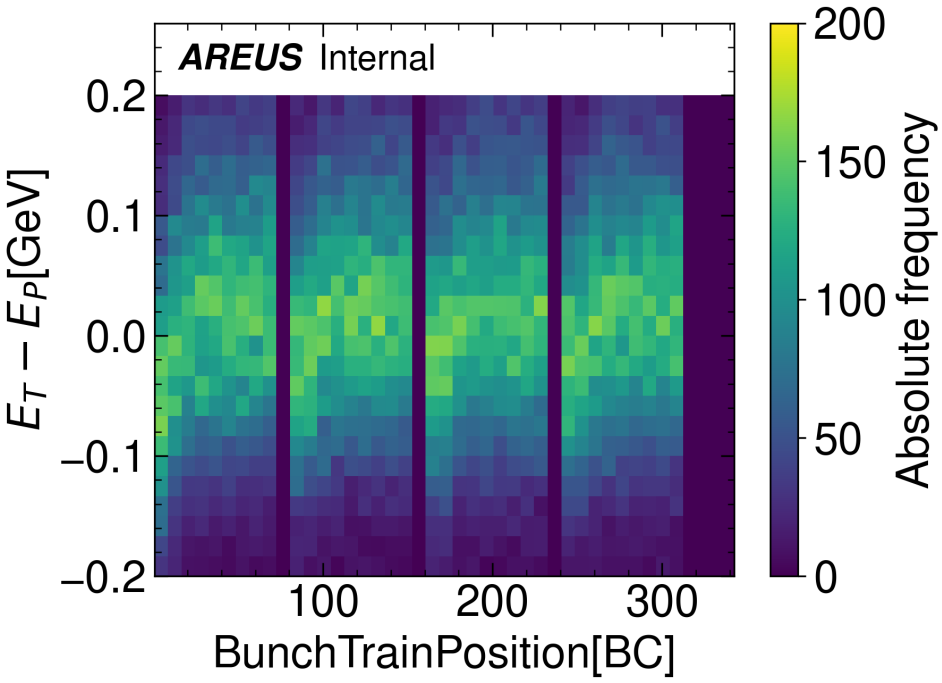
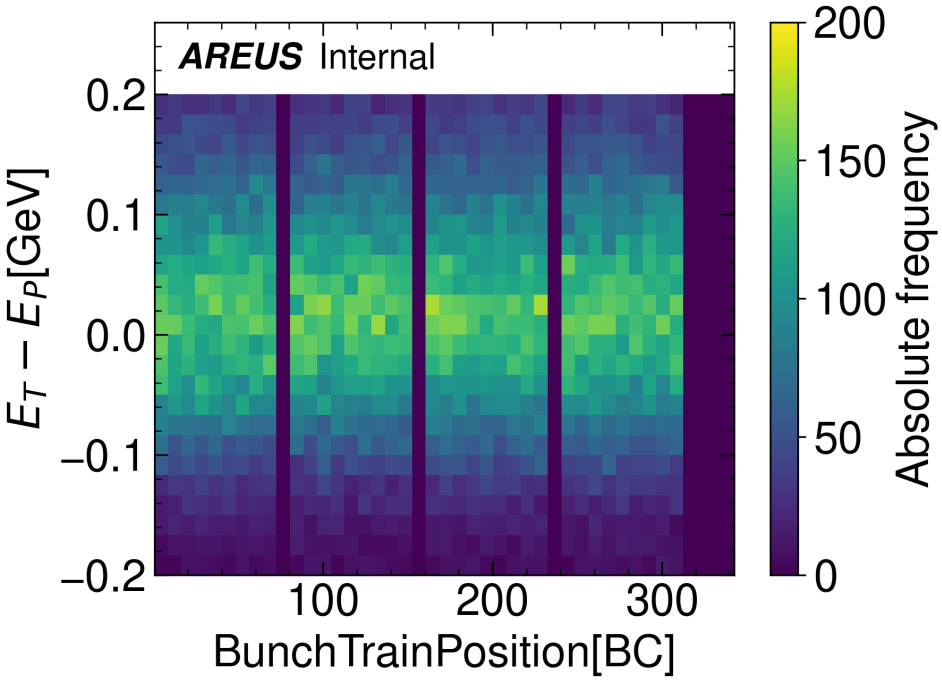


Abbildung 4.4: Heatmap des Optimalfilters. Die Farbe definiert die Anzahl der Einträge pro Bin. $E_T - E_P$ beschreibt die Differenz der richtigen Energie zur rekonstruierten.

Die Bins sind so gewählt, dass die Einteilung in der x-Achse genau den Lücken der BT-Struktur entspricht, dadurch kann im Diagramm die BT-Struktur gut erkannt werden, da dort komplett blaue (=0) Bins auftauchen. Die Daten, die verglichen werden, schließen alle Energieevents ein, die eine bestimmte Grenzenergie überschreiten, die 0.24 GeV ist. Das entspricht einem Signal größer als 3σ über dem Rauschniveau. Von diesen Signalen wird die Differenz der rekonstruierten zur eigentlichen Energie berechnet, eingetragen auf der y-Achse.
Eine optimale Rekonstruktion würde einer hellen Linie auf Höhe von 0 GeV (mit den Lücken) entsprechen. Realistischer für kontinuierliche Eingabedaten ist eine Hauptachse in der Mitte mit einem abnehmenden Farbgradienten für größere Abweichungen entsprechend einer Gaußverteilung.
Was man in Abb. 4.4 erkennen kann, ist eine Überschätzung der Energie zu Beginn jedes Pakets, weswegen die Differenz zum richtigen Wert negativ wird. Am prägnantesten taucht das im ersten Paket auf, wo auch die Baseline den höchsten Ausschlag hat. Somit kann der Einfluss der Baseline in der Energierekonstruktion gut erkannt werden. Interessant ist auch, dass der durchschnittliche Wert der Energierekonstruktion des Optimalfilters etwas zu niedrig, also die Differenz zu hoch ist. Das liegt wahrscheinlich daran, dass bei auf einanderfolgenden Pulsen, das Maximum durch den langen abnehmenden Schweif verringert wird und der Optimalfilter nur die Höhe der Maxima ungeachtet dessen bestimmt. Die Verteilung der Energierekonstruktion wird durch die BT-Struktur wenig beeinflusst und ist konsistent über die gesamte Länge.
In Abb. 4.5a ist die Heatmap des **Conv3**-Netzwerks zu sehen. Die Perfomance ist ähnlich der des Optimalfilters, nur ist die durchschnittliche Differenz am Ende von Paketen näher ist 0 GeV. Für Abb. 4.5b wurde auf den Analysesatz die Baselinekorrektur angewandt. Hierbei wird die zuvor berechnet Baseline periodisch zu einem gleichgroßen Datensatz wie der Analysedatensatz verlängert und dann von diesem abgezogen. So können die Effekte der Baseline reduziert werden.



(a) Conv3 Heatmap ohne Baselinekorrektur



(b) Conv3 Heatmap mit Baselinekorrektur

Abbildung 4.5: Heatmaps der Energierekonstruktion des Conv3-Netzwerks

Die Baselinekorrektur sorgt nun für eine mittlere Differenz von 0 GeV und es tauchen keine Schweife mehr auf. Für ein alternatives, gutfunktioniertes CNN-Modell, das **Conv3 CoRelu**-Netzwerk, taucht der Effekt in Abb. 6.1 ebenfalls auf.
Für die Lücken hat die Baseline des Datensatz negative Werte. Das heißt, wenn man die Baselinekorrektur durchführt, erhöhen sich die Energiewerte in der Lücke, obwohl eigentlich keine Pulse dort vorkommen sollten. Abb. 4.6 zeigt ein Histogramm der rekonstruierten Energien in der Lücke.

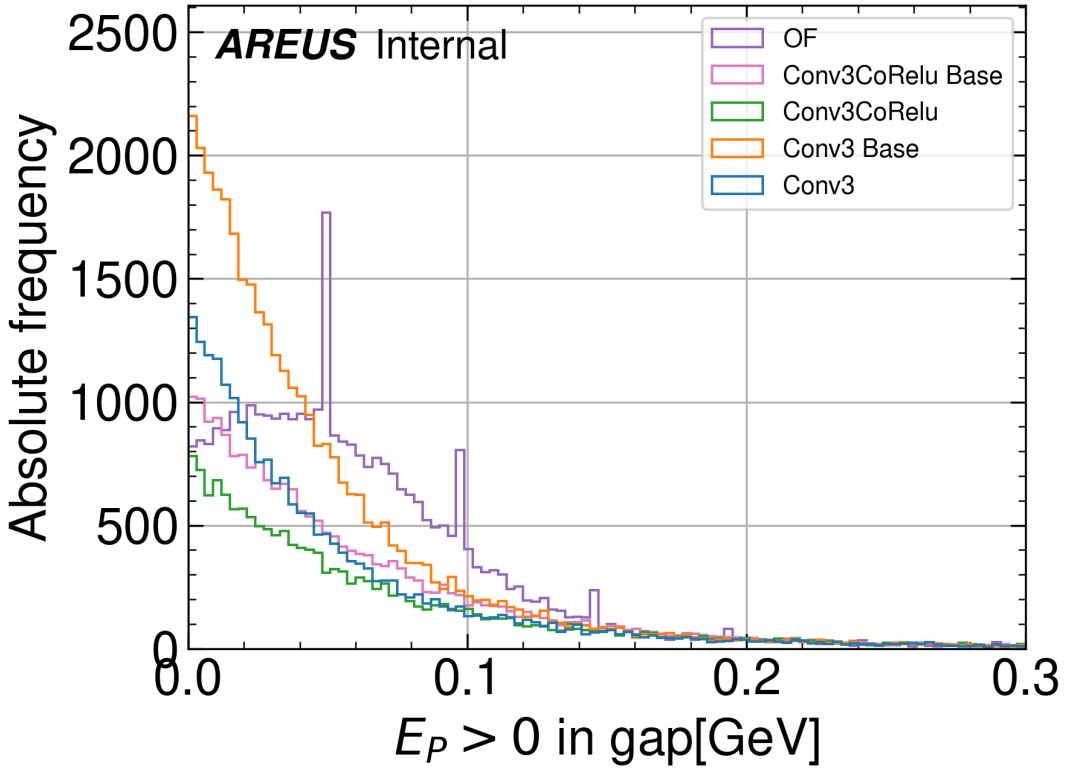


Abbildung 4.6: Histogramm der Energierekonstruktion in den BT-Lücken. Die eingegangenen Daten enthalten alle Rekonstruktion mit einer Energie größer 0 für verschiedene Rekonstruktionsmethoden. Die Bezeichnung Base hinter einem Netzwerk heißt, dass die Baselinekorrektur an der Eingabesequenz durchgeführt wurde.

Dort ist zu sehen, dass die Baselinekorrektur für die CNNs zu einer schlechteren Background Rejection führt. Durch die Erhöhung der Energie im Eingabesignal wird elektronisches Rauschen häufiger als richtiger Energiewert erkannt und somit rekonstruiert. Der Effekt ist vor allem für kleine Energien < 0.1 GeV erkennbar. Dieser negative Effekt wirft die Frage nach einem neuen Umgang mit der Baselinekorrektur

auf.
Das Histogramm des Optimalfilters hat alle 0.05 GeV Peaks in der absoluten Häufigkeit. Diese sind bis kurz vor 0.2 GeV zu erkennen. Der Ursprung ist hier nicht absolut klar. Eine mögliche Erklärung sind diskrete Rauschniveaus in der AREUS Simulation, die vom MaxFinder als Peak identifiziert werden und dann als Vielfache auftreten.

4.2 CNNs mit BT-Struktur als Eingabe

4.2.1 Realisierung

Um bessere Rekonstruktionen am Analysedatensatz möglich zu machen, gibt es die Möglichkeit spezialisierte Netzwerke zu trainieren. Als neue Eingabesequenz kann aus der bekannten BT-Struktur eine Binäre Sequenz generiert werden, die 1 für gefüllte BCs und 0 für leere BCs enthält. Diese Art zweite Eingabe wurde als nicht nützlich für die alte Netzstruktur festgestellt [26]. Stattdessen kann nach weiteren Netzstrukturen mithilfe der Hyperparameteranalyse für den Modellaufbau gesucht werden.
Die Modellstrukturen, die ausprobiert werden, unterscheiden sich grundlegend von den Vorherigen. Es wird auf eine Taggingschicht verzichtet, wodurch die Struktur einfacher bleibt. Zur Auswahl stehen 3-schichtige, 4-schichtige oder 5-schichtige Netze, bei denen mit Hilfe der Funktion parameter_calc die Parameteranzahl des Netzes konstant gehalten wird, aber unterschiedliche Strukturen ausprobiert werden, die den zweiten Eingabeparameter besser nutzen.
Für diese Einstellungen wird ein Hypermodell entwickelt, dessen Suchraumgröße abhängig von der Schichtzahl ist. Die Optimierungsparameter a und b werden in 10 Stufen diskretisiert und für ein n-schichtiges Netzwerk, müssen $2 \cdot (n-1)$ gewählt werden. Das ergibt $10^4 + 10^6 + 10^8 \approx 10^8$ Möglichkeiten, die ausprobiert werden könnten. Dieser große Suchraum wird mit Hilfe des Hyperband-Algorithmus durchleuchtet. Die Modelle können maximal 600 Epochen trainiert werden, was dem Standard der anderen Netzwerke entspricht. Durch den Hyperband-Algorithmus sollten schlecht funktionierende bzw. nicht konvergierende Modelle ausgeschlossen werden. Trotzdem bedeutet die Größe des Suchraums, dass, bei einer durchschnittlichen Epochentrainingszeit von 1 s, alle Möglichkeiten eine Epoche zu trainieren über 3 Jahre dauern würde.
Das Problem ist mit dem Keras-Tuner [16] implementiert. Die Modelle in den Hyper-

modellen werden an einen baselinekorrigierten Datensatz trainiert, Als Aktivierungsfunktion wird die ReLu-Funktion, als Verlustfunktion der mittlere absolute Fehler genutzt. Die Zielsetzung des Hypermodelltrainings wird gleich dem kumulativen Verlust gewählt.

4.2.2 Training

Das Training des Hypermodells ergab als bestes Modell für eine maximale Parameteranzahl von 100 ein 3-schichtiges Netzwerk (**BestHyper**). Die erste Schicht besteht aus 5 Filtern mit einer Kernelsize von 5, darauf folgen wieder 5 Filter mit einer Kernelsize von jeweils 1. Die letzte Schicht muss einen Filter haben, der eine Kernelsize von 2 hat. Somit ist die gesamte Überdeckungsreichweite des Netzwerks 7 BC. In Abb. 4.7 kann die Lückenperformance erkannt werden. Das neu trainierte Netzwerk hat eine

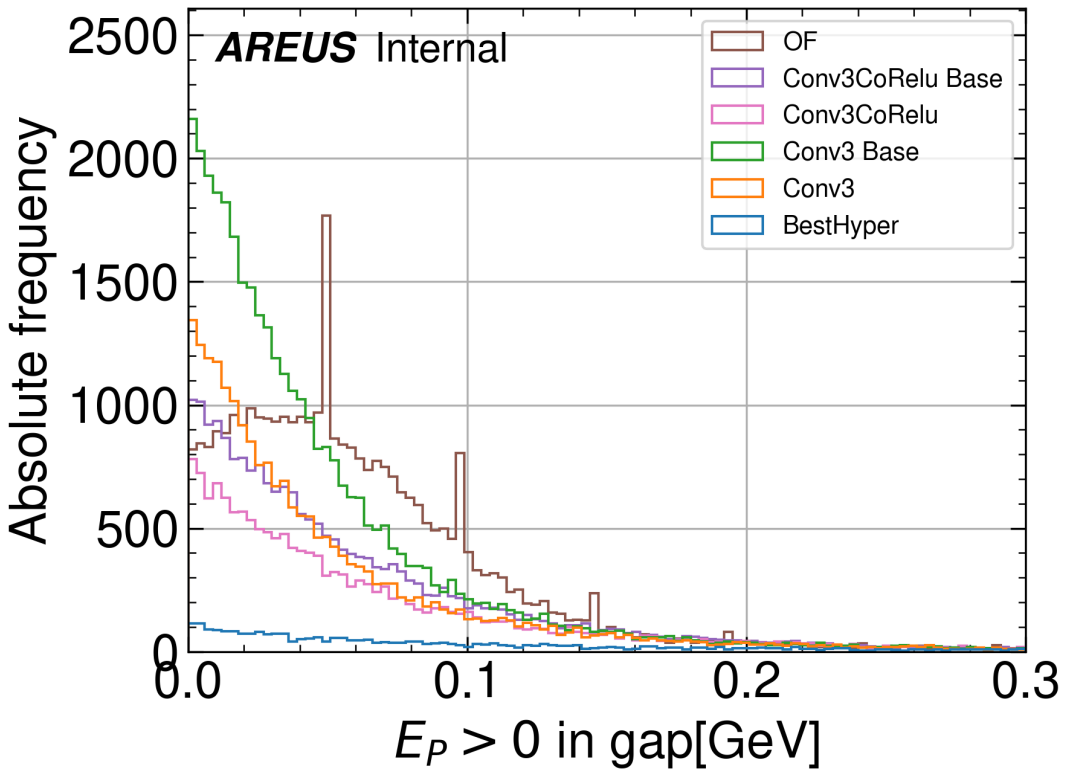


Abbildung 4.7: Histogramm der Energierekonstruktion in den BT-Lücken mit besten Modell des Hypertrainings (BestHyper).

bessere Background Rejection und rekonstruiert weniger Energien in den BT-Lücken.

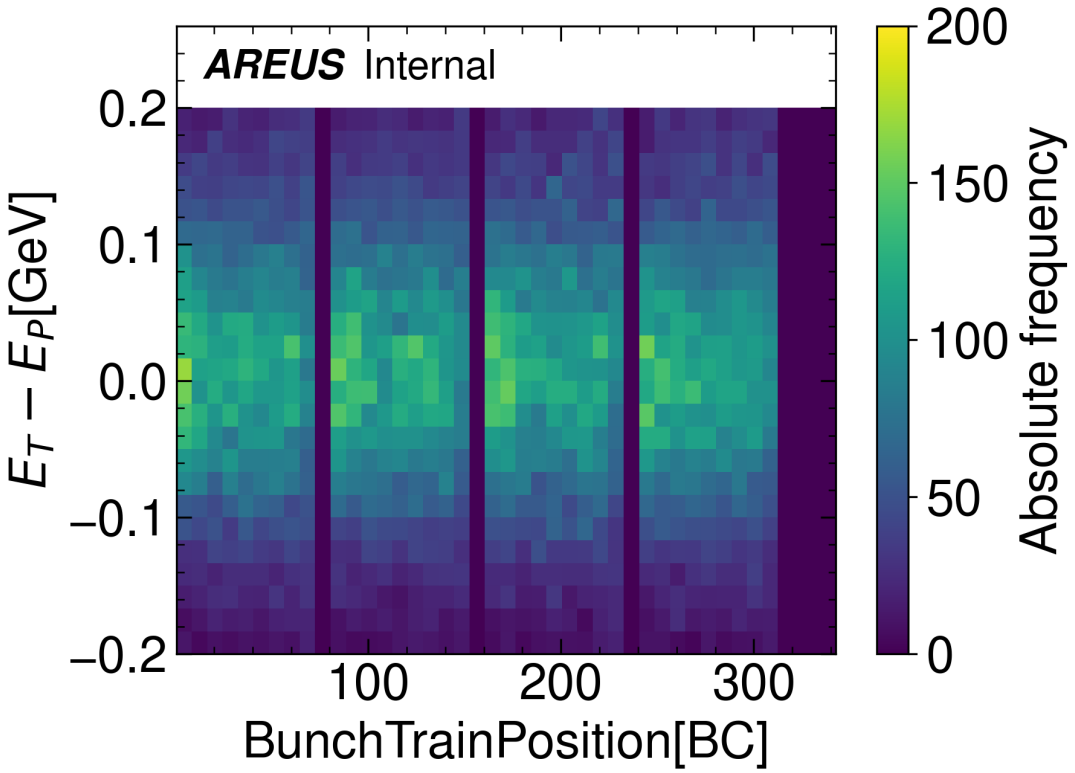


Abbildung 4.8: Heatmap des BestHyper-Netzwerks

In Abb. 4.8 ist die Heatmap der Energierekonstruktion dieses Netzwerks dargestellt. Die Performance zeigt keine Änderung in Abhängigkeit von der BT-Struktur. Die Energiedifferenz zeigt einen durchschnittlichen Wert von 0 GeV. Im Vergleich zu Abb. 4.5b sind die Bins weniger gefüllt, was entweder auf eine breitere Verteilung der Rekonstruktion pro BT-Position bzw. eine schlechtere Signal Rejection hinweist.
Um die Verteilungunterschiede besser zu betrachten, ist in Abb. 4.9 die Verteilung des relativen Fehler der Energierekonstruktion der ersten 30 BT-Positionen aufgetragen, weil dort die Effekte der BT-Struktur am ausgeprägtesten sind.

Der Peak bei 1.0 beschreibt die nicht gefundenen Pulse, da dort die Energiedifferenz gleich der Energiehöhe ist. Die unterschiedlichen Peakhöhen an der Stelle beschreiben somit die Signal Rejection, die Fehlquote der Energierekonstruktion. An dieser Stelle kann man sehen, dass **Conv3** die meisten Signale erkennt, gefolgt vom

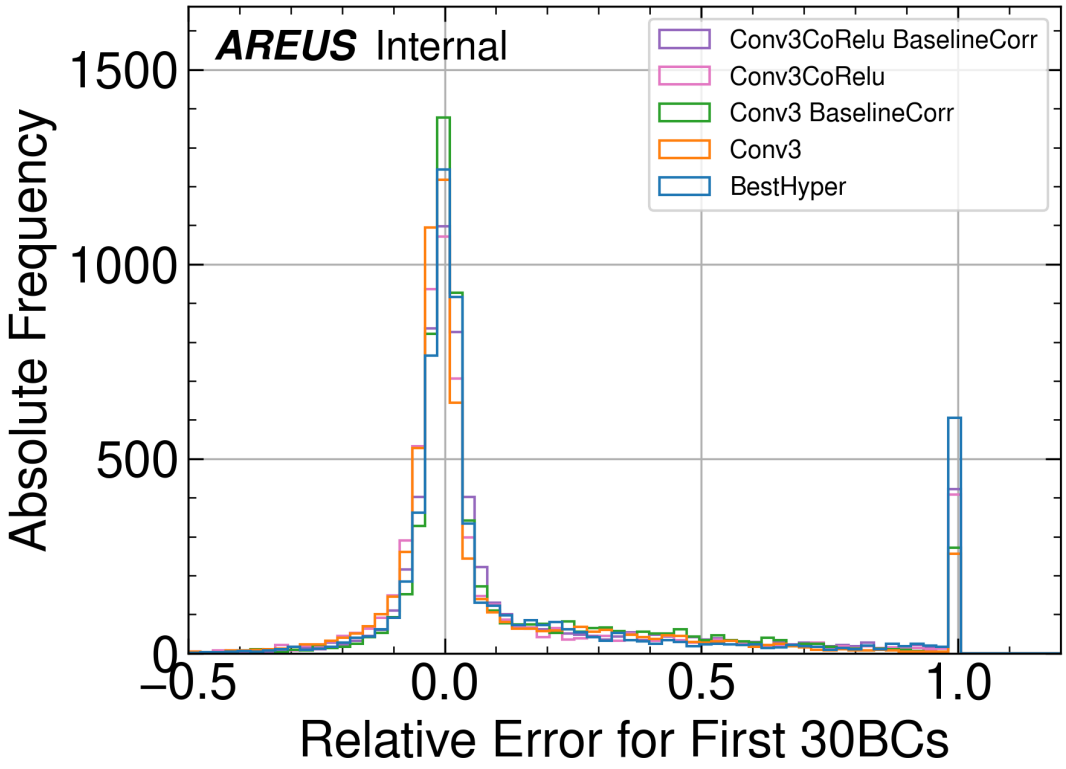


Abbildung 4.9: Verteilung des relativen Fehler der Energierekonstruktion für verschiedene Netzwerke. Es wird nur die Rekonstruktion von Energien über 0.24 GeV dargestellt.

Conv3CoRelu und dann dem neuen **BestHyper**. Das passt auch mit den Ergebnissen der Heatmap zusammen, die grundlegend mit höherer Streuung ausgefallen ist. Die Verteilung um 0 ähnelt sich hingegen bei allen Netzwerken. Sie ist gaußförmig und hat ähnlich große Standardabweichungen.
Das Training mit dem Hypermodell wurde mehrfach für lange Rechenzeiträume durchgeführt, die zu unterschiedlichen besten Modellen geführt haben. Jedoch hatten alle davon schlechtere Performance im Vergleich zum **BestHyper**-Netzwerk. Meistens können solche Training schnell ausgeschlossen werden mit dem Blick auf die in der Lücke rekonstruierten Energien. Schlechte Trainings tendieren nämlich dazu, eine mittlere Energie auf Höhe des Pionenuntergrunds auszugeben, wodurch die Verlustfunktion niedrig wird, ohne das eine gute Rekonstruktion gelernt wird. In Abb. 6.2 sind 3 unterschiedliche Ergebnisse dargestellt. Diese Netzwerke sind ungeeignet für die Energierekonstruktion.

4.2.3 Effekt der BT-Struktur als Eingabe

Das Training eines neuen Modells, das die BT-Struktur mit als Eingabe übernimmt, hat aufgezeigt, dass teilweise bessere Performance in Bezug auf die BT-Struktur erreicht werden konnte. Jedoch ist die Frage, inwiefern die Eingabe der BT-Struktur einen wirklichen Effekt auf Pulse in unterschiedlichen BT-Positionen hat. Zum Beispiel wäre es möglich, dass das Training mit der binären Eingabe zu einer Art logischen Konjuktion geführt hat, die Energierekonstruktion größer null nur zulässt, wenn die BT-Eingabe eine 1 ist.
Das hätte zwar den Effekt, dass keine Energie an Stellen, an denen keine Pulse auftauchen rekonstruiert wird, jedoch würde jegliche Messung in den Lücken dann sinnlos sein, weil das Netzwerk dort immer 0 rekonstruiert.
Dieser Effekt könnte Nachteile bei der Messung von Physik jenseits des Standardmodells bringen. Für die Messung von hypothetischen langlebigen Teilchen, die sich mit niedriger Geschwindigkeit bewegen und erst im Detektor zerfallen, könnten Signale auch in der BT-Lücke auftauchen. [27]
Zu diesem Zweck wird ein Puls in einer Lücke simuliert, um die Antwort des **BestHyper**-Netzwerkes zu untersuchen. In Abb. 4.10 wird ein solcher Extrapuls mit 4.5 GeV in der letzten Lücke dargestellt. Dazu wurde die ideale Pulsform im Nachhinein im Eingabedatensatz an der richtigen Stelle hinzugefügt, bevor die Modelle die Rekonstruktion berechnen.
Man kann erkennen, dass trotz einer Lücke (zweite Eingabe mit 0) eine Rekonstruktion des **BestHyper**-Netzwerks durchgeführt wurde. Stichprobenartig wurden weitere Stellen in der Datensequenz untersucht (Abb. 6.3) an denen dann ein Puls in die Lücke hinzugefügt wurde, die gezeigt haben, dass beide Netzwerke eine Rekonstruktion durchführen. Jedoch ist es sinnvoll weitere systematische Betrachtungen im Hinblick auf die Qualität und Konsistenz der Rekonstruktion in der Lücke durchzuführen.

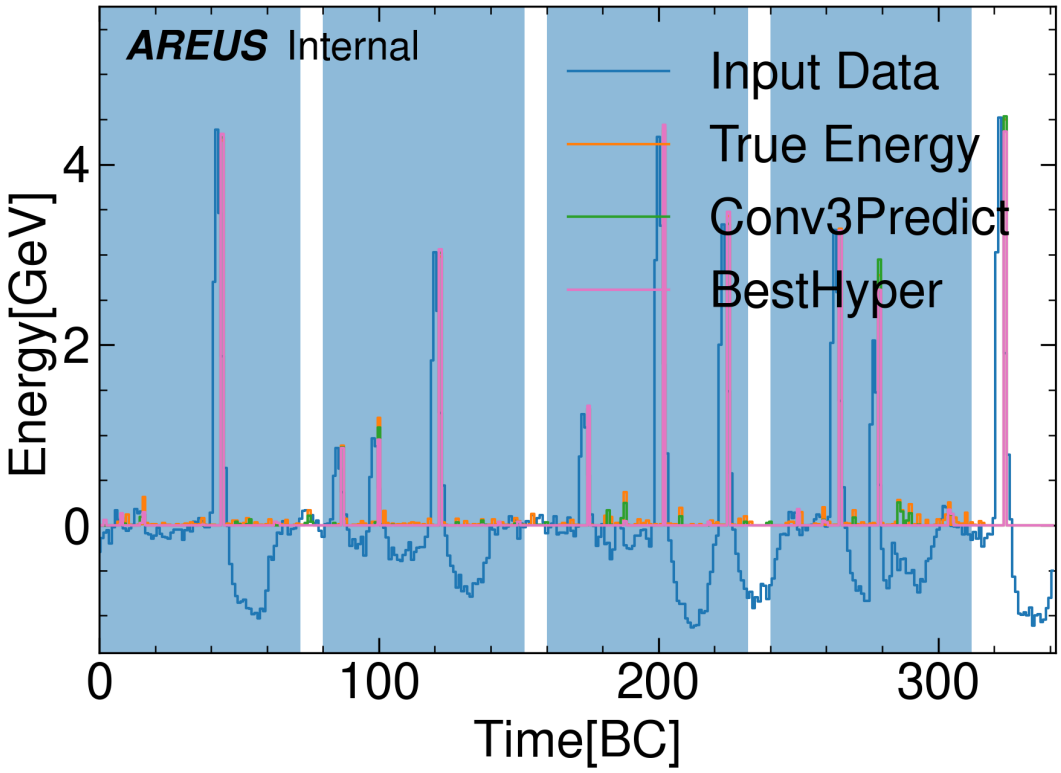


Abbildung 4.10: Ausschnitt des Analysedatensatz (3. BT-Durchgang) mit einem extra eingefügten Puls in der letzten Lücke. Input Data entspricht der digitalisierten Datensequenz und True Energy der korrekten Ausgabe. Zum Vergleich ist die Rekonstruktion **Conv3**-Netzwerks mitaufgetragen. Blaue Bereiche zeigen volle BCs, weiße leere.

4.3 Diskussion

In diesem Kapitel wurde der Einfluss der BT-Struktur auf die Energierekonstruktion sowie mögliche Lösungen betrachtet. Es hat sich gezeigt, dass dedizierte Netzwerke mindestens gleich gute, auch in den Lücken bessere, BT-Performance aufzeigen. Allerdings gibt es im Vergleich zwischen **BestHyper** und **Conv3** noch einen weiteren Leistungspunkt, die eigentliche Energierekonstruktion. Hier ist das **Conv3**-Netzwerk mit einer vorher angewandten Baselinekorrektur genauer und erkennt vor allem mehr Signale, was insgesamt die Genauigkeit hin zur besseren Energierekonstruktion beeinflusst.

Eine mögliche Lösung wäre die Baselinekorrektur nur in den gefüllten Paketen anzuwenden, damit die Blindsignalen in den Lücken auf das Niveau vor der Baseline-

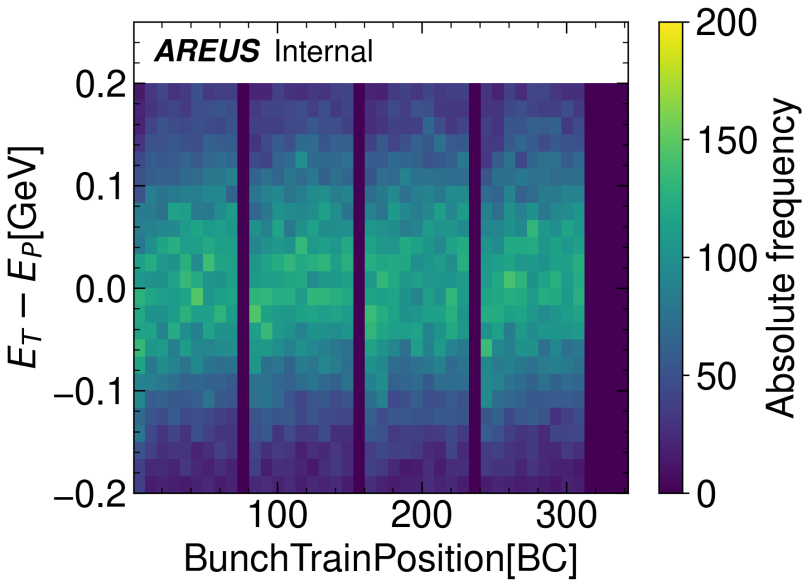
korrektur reduziert werden(siehe Abb. 4.7). Jedoch gibt es dann das Problem, falls Signale in der Lücke auftauchen, dass deren Energierekonstruktion nicht korrigiert ist.
Eine Alternative ist die Baselinekorrektur durchzuführen und die Datenaufzeichnung in der Lücke an ein Triggersystem zu binden, das misst, ob eine bestimmte Schwellenenergie überschritten ist. Diese Schwelle könnte knapp über dem absoluten Wert des Minimums der Baseline (Abb. 4.3) gewählt werden. So ist die Menge an Blindsignalen reduziert, aber die Messung von langlebigen Teilchen nicht ausgeschlossen.

5 Zusammenfassung und Ausblick

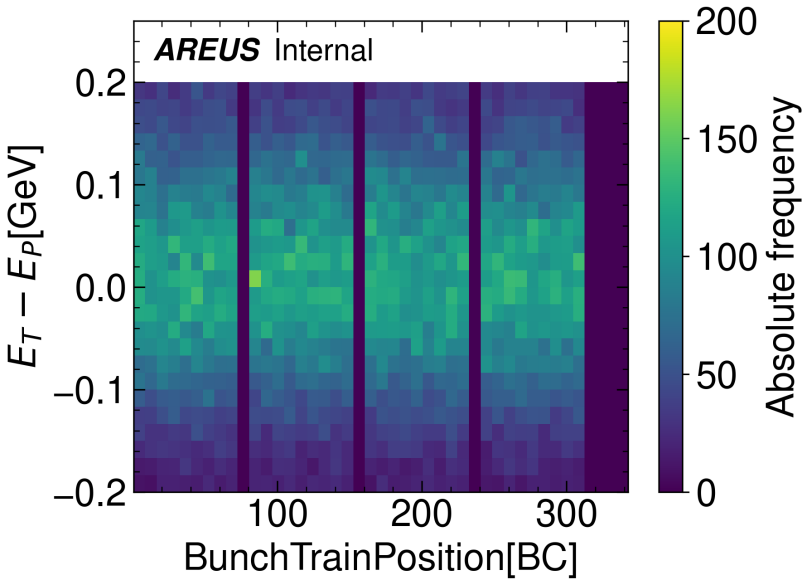
In dieser Arbeit wurden Möglichkeiten dargelegt, wie das Training von neuronalen Netzwerken mit Hilfe von Hyperparametersuchen optimiert werden kann. Für das Anwenden von neuronalen Netzwerken für Energierekonstruktion am Flüssigargonkalorimeter des ATLAS-Detektors wurde das Pileup-Problem der Bunch-Train-Struktur der Protonenpakete analysiert sowie durch Hyperparameteroptimierungen versucht zu lösen.
Es hat sich gezeigt, dass der Einsatz von Hyperparameteroptimierungen auf verschiedene Probleme möglich ist. Das Problem der Initialparameter eines Netzwerks kann durch eine Random Search modelliert werden. Dort haben kleinere (< 1) Standardabweichungen zu konsistenteren Ergebnissen geführt, jedoch haben höhere Werte zu besseren sowie schlechteren Werten geführt. Diese Möglichkeit der hohen Standardabweichungen kann in Verbindung mit anderen Optimierungsalgorithmen wie Hyperband im Training verwendet werden, so dass Rechenleistung gespart wird.
Die Modellstruktur als größte Variable im Training von neuronalen Netzwerken wurde auch als Hyperparameterproblem formuliert. Für die Beschränkung auf kleine Netzwerkgrößen ist dort ein Algorithmus notwendig, der entsprechende Parameterwahlen trifft, die im Einklang mit der Randbedingung der Netzwerkgröße ist. Dadurch kann eine große Anzahl von verschiedenen Modellstrukturen ohne größeren Aufwand trainiert werden.
Die BT-Struktur sorgt für eine Baseline-Verschiebung, die die Energierekonstruktion beeinflusst. Die bipolare Form der Pulse sorgt für Effekte an den Übergängen zwischen gefüllte Protonenpaketen und Lücken. Zur Analyse dieser Effekte wurde ein neuer Datensatz generiert, der eine der Realität nahen Struktur hat. Hier war das Pileup klar in der Energierekonstruktion zu finden. Eine Lösung dafür ist die Baselinekorrektur, also die Berechnung des Energiemittelwerts für die periodisches Position in der BT-Struktur. Wenn dieser Mittelwert abgezogen wird, ist die Performance der Netzwerke fast ohne Störeffekte der BT-Struktur, lediglich die Signale in den Lücken werden erhöht.

Weiterhin wurde eine Hyperparameteroptimierung für verschiedene Netzwerkstrukturen an einem Datensatz mit BT-Struktur durchgeführt, wobei die Netzwerke eine zweite Eingabe in Form von einer binären Sequenz hatten. Diese führte zu einem Netzwerk mit hoher Erkennungseffizienz, was noch besser auf die BT-Struktur reagiert. Jedoch ist die eigentliche Rekonstruktionsleistung schlechter.
Von diesen Ergebnissen ausgehend empfiehlt es sich, die individuell besten Netzwerke zur Energierekonstruktion mit einer Baselinekorrektur zu benutzen, die nur in den gefüllten Paketen angewandt wird. So muss der Blindstrom in den Lücken nicht erhöht werden, der die Messung beeinflusst. Andererseits muss die Suche nach Alternativmodellen nicht aufgegeben werden, der Suchraum ist nicht erschöpft. Andere Formen ohne vorgelagerte Taggingstruktur, aber auch ohne extra BT-Eingabe können weiterhin ausprobiert werden. Dazu sind Hyperparameteroptimierungen ein mächtiges Werkzeug das den Programmier- sowie Analyseaufwand stark verringert, sobald es eine Implentierung gibt.
Zum jetzigen Zeitpunkt sind die Netzwerke für den Einsatz in den mittleren Schichten des elektromagnetischen Barrels spezialisiert. Es muss untersucht werden, inwiefern diese Spezialisierung einen Einfluss auf die Messung hat und weitere Trainings für andere Detektorbereiche notwendig sind. Falls jede Zelle des Flüssigargonkalorimeter ein eigenes Training benötigt, wäre das mit Hilfe von Algorithmen wie Hyperband, die die Rechenzeiten verkürzen, leicht implementierbar.

6 Anhang

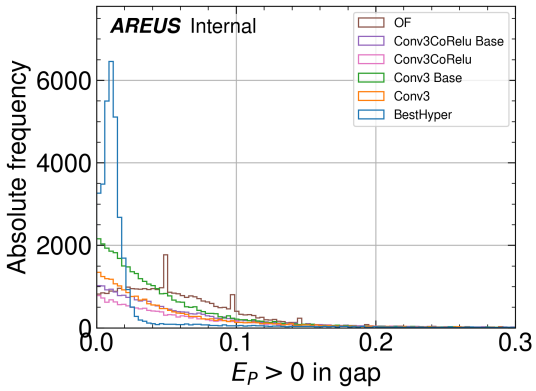


(a) Conv3CoRelu Heatmap ohne Baselinekorrektur

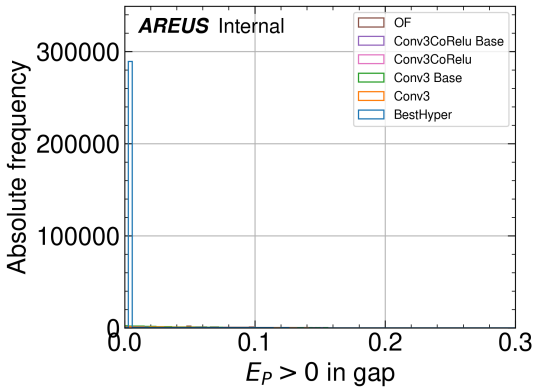


(b) Conv3 Heatmap mit Baselinekorrektur

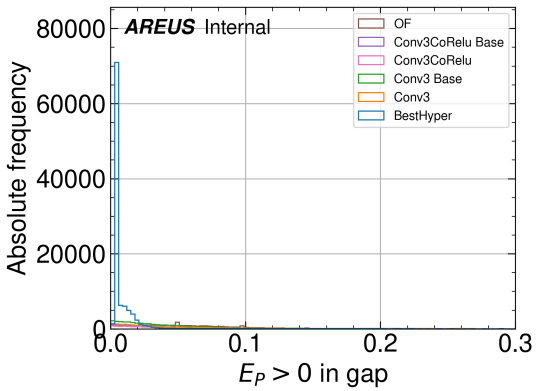
Abbildung 6.1: Heatmaps der Energierekonstruktion des Conv3CoRelu-Netzwerks



(a) 3-schichtiges Netzwerk als bestes Netzwerk dieses Trainings.



(b) 4-schichtiges Netzwerk als bestes Netzwerk dieses Trainings.



(c) Weiteres 4-schichtiges Netzwerk als bestes Netzwerk dieses Trainings.

Abbildung 6.2: Weitere Ergebnisse der Hyperparametersuche mit BT-Eingabe. Die Bezeichnung BestHyper steht in diesem Fall für das jeweilige Modell. Ein Training hat ungefähr kumulativ 80000 Epochen trainiert (für verschiedene Modellstrukturen).

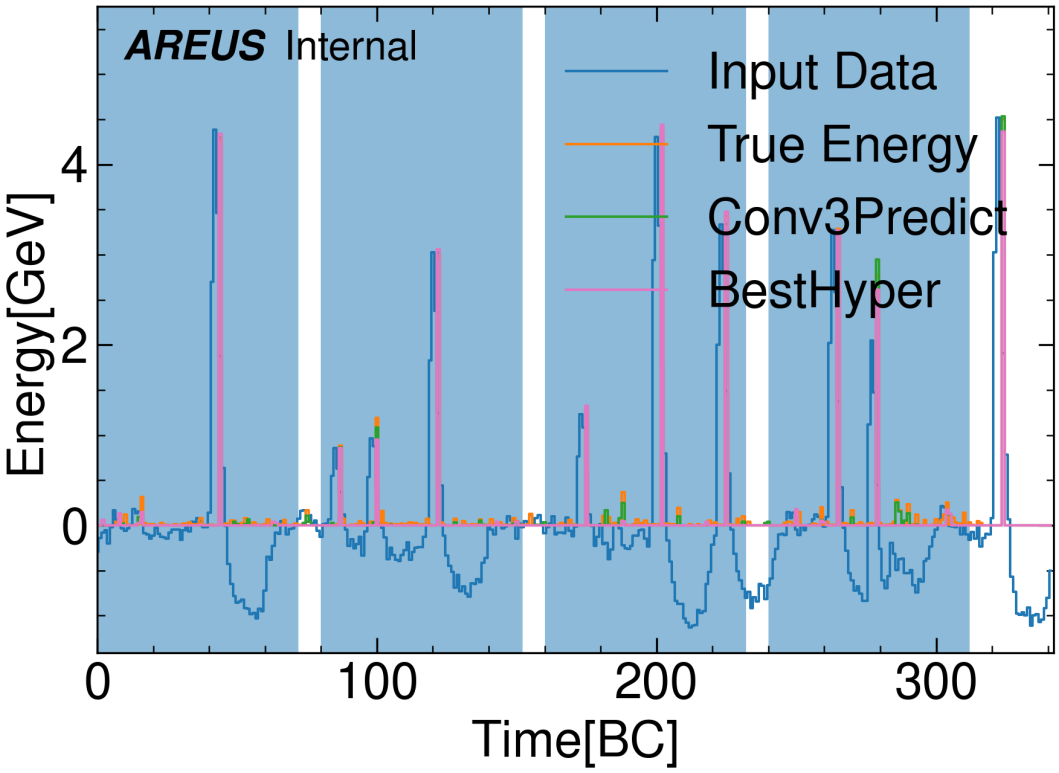


Abbildung 6.3: Ausschnitt des Analysedatensatz (2. BT-Durchgang) mit einem extra eingefügten Puls in der letzten Lücke. Input Data entspricht der digitalisierten Datensequenz und True Energy der korrekten Ausgabe. Zum Vergleich ist die Rekonstruktion **Conv3**-Netzwerks mitaufgetragen.

Abbildungsverzeichnis

Literatur

[1] G. Aad u. a. „Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC“. In: *Physics Letters B* 716.1 (2012), S. 1–29. ISSN: 0370-2693. DOI: https://doi.org/10.1016/j.physletb.2012.08.020.

[2] *The HL-LHC project*.

[3] Ana Lopes und Melissa Loyse Perrey. „FAQ-LHC The guide“. Jan. 2022.

[4] Joao Pequenao. „Computer generated image of the whole ATLAS detector“. März 2008.

[5] *ATLAS inner detector: Technical Design Report, 1*. Technical design report. ATLAS. Geneva: CERN, 1997.

[6] A. Airapetian u. a. *ATLAS detector and physics performance: Technical Design Report, 1*. Technical design report. ATLAS. Geneva: CERN, 1999.

[7] Joao Pequenao. „Computer generated image of the ATLAS Liquid Argon“. März 2008.

[8] *ATLAS Liquid Argon Calorimeter Phase-II Upgrade: Technical Design Report*. Techn. Ber. Geneva: CERN, Sep. 2017. DOI: 10.17181/CERN.6QIO.YGHO.

[9] Georges Aad u. a. *Artificial Neural Networks on FPGAs for Real-Time Energy Reconstruction of the ATLAS LAr Calorimeters*. Techn. Ber. Geneva: CERN, Juli 2021. DOI: 10.1007/s41781-021-00066-y.

[10] 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.

[11] Ian J. Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[12] Wolfgang Bibel, Steffen Holldobler und Torsten Schaub. *Wissensrepräsentation und Inferenz*. de. 1993. Aufl. Computational Intelligence. Wiesbaden, Germany: Vieweg+Teubner Verlag, Juli 1993.

[13] Brian K. Spears u. a. „Deep learning: A guide for practitioners in the physical sciences“. In: *Physics of Plasmas* 25.8 (Aug. 2018), S. 080901. DOI: 10.1063/1.5020791.

[14] François Chollet u. a. *Keras*. https://keras.io. 2015.

[15] Matthias Feurer und Frank Hutter. „Hyperparameter Optimization“. In: *Automated Machine Learning: Methods, Systems, Challenges*. Hrsg. von Frank Hutter, Lars Kotthoff und Joaquin Vanschoren. Cham: Springer International Publishing, 2019, S. 3–33. ISBN: 978-3-030-05318-5. DOI: 10.1007/978-3-030-05318-5_1.

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

[17] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN: 9780486428093.

[18] James Bergstra und Yoshua Bengio. „Random Search for Hyper-Parameter Optimization“. In: *Journal of Machine Learning Research* 13 (Feb. 2012), S. 281–305. ISSN: 1532-4435.

[19] Jasper Snoek, Hugo Larochelle und Ryan P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms*. 2012. DOI: 10.48550/ARXIV.1206.2944.

[20] Lisha Li u. a. „Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization“. In: *Journal of Machine Learning Research* 18.185 (2018), S. 1–52.

[21] Laith Alzubaidi u. a. „Review of deep learning: concepts, CNN architectures, challenges, applications, future directions“. In: *Journal of Big Data* 8.1 (März 2021), S. 53. ISSN: 2196-1115. DOI: 10.1186/s40537-021-00444-8.

[22] Francesca Cavallari. „Performance of calorimeters at the LHC“. In: *Journal of Physics: Conference Series* 293 (Apr. 2011), S. 012001. DOI: 10.1088/1742-6596/293/1/012001.

[23] Anne-Sophie Reimer. *Performance of the Demonstrator System of the future ATLAS LAr Calorimeter Trigger Readout*. Masterarbeit. Dresden, Juli 2018.

[24] R Bailey und Paul Collier. *Standard Filling Schemes for Various LHC Operation Modes*. Techn. Ber. Geneva: CERN, Sep. 2003.

[25] Nico Madysa. „Simulation Studies of Digital Filters for the Phase-II Upgrade of the Liquid-Argon Calorimeters of the ATLAS Detector at the High-Luminosity LHC“. Presented 30 Sep 2020. Feb. 2020.

[26] Thord Vincent Elst. *Optimierung der Energiemessung im Flüssig-Argon-Kalorimeter des ATLAS-Detektors am LHC mit künstlichen neuronalen Netzen*. Staatsexamen. Dresden, Okt 2021.

[27] Sascha Mehlhase. *Searches for long-lived particles in ATLAS*. Techn. Ber. Geneva: CERN, Aug. 2019. DOI: 10.22323/1.350.0176.

Erklärung

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

Philipp Welle
Dresden, Juni 2022