Technischen Hochschule Nürnberg Georg Simon Ohm
Fakultät Elektrotechnik Feinwerktechnik Informationstechnik

Studiengang: Elektronische und Mechatronische Systeme
Masterarbeit von
Philipp Renner

Deep Learning basierte Segmentierung und Transformation von Spike-Rauschen in der MR-Bildgebung

Wintersemester 2022/Sommersemester 2023

angefertigt bei
Siemens Healthineers AG
Betreuer
Mario Zeller

Erstkorrektor: Prof. Dr. Oliver Hofmann
Zweitkorrektor: Prof. Dr. Jan Paulus
Abgabedatum: 28.06.2023

Schlagwörter: Künstliche Intelligenz, Bildklassifikation, MRT

Offizielles Deckblatt

Bearbeiter:	Philipp Renner
Matrikelnummer:	3173820
Studiengang:	Elektronische und Mechatronische Systeme
Vertiefungsrichtung:	Informationstechnik
Erstprüfer:	Prof. Dr. Oliver Hofmann
Zweitprüfer:	Prof. Dr. Jan Paulus
Durchgeführt bei:	Siemens Healthineers AG Henkestraße 127 91052 Erlangen
Betreuer der Firma:	Dr. Mario Zeller Telefonnummer: +49 (9131) 84-5216 E-Mail: Mario.Zeller@Siemens-Healthineers.com
Semester:	Wintersemester 2022/Sommersemester 2023
Ausgabedatum:	01.010.2022
Abgabedatum:	28.06.2023
Thema der Arbeit:	Konzeption und Entwicklung einer Bildverarbeitungspipeline zur Reduktion von Spike Rauschen in MRT-Daten.
Arbeit ist frei einsehbar	[X] Ja [ ] Nein

Hinweis: Diese Erklärung ist in alle Exemplare der Abschlussarbeit fest einzubinden. (Keine Spiralbindung)

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Renner Vorname: Philipp Matrikel-Nr.: 3173820

Fakultät: EFI Studiengang: MSY

Semester: 4

Titel der Abschlussarbeit:

Deep Learning basierte Segmentierung und Transformation von Spike-Rauschen in der MR-Bildgebung

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum, Unterschrift Studierende/Studierender

Erklärung der/des Studierenden zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.

Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit ☑ genehmige ich, wenn und soweit keine entgegenstehenden Vereinbarungen mit Dritten getroffen worden sind,

☐ genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgebrachten Sperrvermerks kenntlich gemachten Sperrfrist

von _____ Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigefügt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

Ort, Datum, Unterschrift Studierende/Studierender

Datenschutz: Die Antragstellung ist regelmäßig mit der Speicherung und Verarbeitung der von Ihnen mitgeteilten Daten durch die Technische Hochschule Nürnberg Georg Simon Ohm verbunden. Weitere Informationen zum Umgang der Technischen Hochschule Nürnberg mit Ihren personenbezogenen Daten sind unter nachfolgendem Link abrufbar: https://www.th-nuernberg.de/datenschutz/

Kurzfassung

Die Magnetresonanztomographie (MRT) ist ein bildgebendes Diagnoseverfahren, das auf einer Kombination aus starken Magnetfeldern und sensiblen Detektoren basiert. Durch externe sowie interne Störquellen kann es zu sogenanntem Spike-Rauschen kommen. Dabei wird dem entstehenden Bild ein Muster aus Linien überlagert und somit die Bildqualität negativ beeinflusst. In den Frequenzräumen der betroffenen Bilder wird das Rauschen deutlich sichtbar durch wenige Pixel, die sich durch eine höhere Intensität von ihrer umgebenden Nachbarschaft abheben. Der Qualitätsverlust innerhalb der entstehenden Bilder kann eine neue Aufnahme der Bilder notwendig machen, was wiederum Zeit und Kosten verursacht.

Diese Arbeit beschäftigt sich mit der Frage, wie gut eine Bildverarbeitung mittels neuronalen Netzen das Spike-Rauschen reduzieren kann und somit eine Verbesserung der Bildqualität erzielt. Dazu wird eine Bildverarbeitungspipeline entworfen, die aus drei Teilbereichen besteht: der Detektion, der Segmentierung und der Transformation. Für jeden der drei Teilbereiche werden verschiedene Architekturen neuronaler Netze ausgewählt und ihr Aufbau erläutert. Anschließend werden die Netzwerke auf Basis der L1-Norm, des Betrags und der Real- und Imaginärteile der verfügbaren komplexen Daten trainiert.

Die Ergebnisse zeigen, dass sowohl die Detektion als auch die Segmentierung basierend auf der L1-Norm die besten Ergebnisse liefern. Basierend auf ihrer Effektivität werden für die Detektion eine VGG-Architektur und für die Segmentierung ein UNetPP ausgewählt. Die Transformation erfolgt aus mathematischen Gründen auf Basis der Real- und Imaginärteile. Hier wird aufgrund eines leicht besseren Ergebnisses eine ResNet-basierte Architektur ausgewählt.

Die Ergebnisse der gesamten Pipeline zeigen, dass das Rauschen innerhalb der Bilddaten erheblich reduziert werden kann. Um die Robustheit der implementierten Pipeline sicherzustellen, müssen jedoch in Zukunft noch mehr Tests durchgeführt werden.

Abstract

Magnetic resonance imaging (MRI) is a diagnostic imaging technique based on a combination of strong magnetic fields and sensitive detectors. External as well as internal sources of interference can cause so-called spike noise. In this case, a pattern of lines is superimposed on the resulting image, thus negatively affecting the image quality. In the frequency spaces to the affected images the noise becomes clearly visible by a few pixels, which stand out from their surrounding neighborhood by higher intensity. The loss of quality within the resulting images can make it necessary to retake the images, which in turn takes time and costs money.

This thesis deals with the question of how well image processing using neural networks can reduce spike noise and thus achieve an improvement in image quality. For this purpose, an image processing pipeline is designed, which consists of three sub-areas, detection, segmentation and transformation. Different neural network architectures are selected for each of the three subdomains and their design is explained. Then, the networks are trained based on the L1 norm, the magnitude and the real and imaginary parts of the available complex data.

The results show that both detection and segmentation based on the L1 norm give the best results. Based on the effectiveness, the architecture chosen is ResNet for detection and UNetPP for segmentation. The transformation must take place for mathematical reasons on basis of real and imaginary part. Here a ResNet based architecture is selected due to a slightly better result.

The results of the whole pipeline then show that the noise within the image data can be reduced significantly. However, more testing needs to be done in the future to ensure the robustness of the implemented pipeline.

Inhaltsverzeichnis

Abkürzungsverzeichnis

HF-Impuls	Hochfrequenz Impuls
KI	Künstliche Intelligenz
KNN	Künstliche Neuronale Netze
FC	Fully Connected
ReLU	Rectified Linear Unit
MSE	Mean Squared Error
BCE	Binary Cross Entropy
Adam	Adaptive Moment Estimation
CNN	Convolutional Neural Networks
BN	Batch Normalization
ResNet	Residual Network
IoU	Intersection over Union

1. Einleitung

1.1 Motivation

In Deutschland ist der Einsatz von Kernspintomographen, auch bekannt als Magnetresonanztomographen (MRT), im Vergleich zu anderen europäischen Ländern am weitesten verbreitet. Laut einer Studie von Radtke (2022) gab es im Jahr 2019 durchschnittlich 34,5 Tomographen pro eine Million Einwohner. Diese hohe Verbreitung unterstreicht die bedeutende Rolle, die die MRT in der modernen Medizin in Deutschland einnimmt. Die MRT ermöglicht die nicht-invasive Darstellung von Geweben und Organen mit hoher Auflösung, ohne den Einsatz von ionisierender Strahlung. Dies wird durch die gezielte Anwendung von starken Magnetfeldern und hochfrequenten Radiowellen erreicht. Die Qualität der MRT-Bilder bildet die Grundlage für die diagnostische Anwendung. Allerdings können verschiedene Störeinflüsse während der Aufnahme die Bildqualität erheblich beeinträchtigen. Im schlimmsten Fall kann dies dazu führen, dass die Aufnahme unbrauchbar ist und wiederholt werden muss.

In der vorliegenden Arbeit wird das Spike-Rauschen in seinem Kontext als Störfaktor behandelt. Besonderes Augenmerk wird dabei auf die Wirkung des Rauschens auf die entstehende Bildqualität gelegt. Grundsätzlich werden externe Felder durch das Design der RF-Kabine, als physikalischer Faraday-Käfig, vermieden. Dennoch besteht die Möglichkeit, dass innerhalb der Kabine Felder erzeugt werden. Diese können von elektronischen Geräten, beispielsweise defekten Leuchtstoffröhren, während ihrer Benutzung emittiert werden. Darüber hinaus können sich bewegende metallische Objekte wie Münzen oder Schrauben, lockere Kabelverbindungen oder Funkenüberschläge Störsignale verursachen. Sobald diese Felder mit den Feldern des Tomographen interferieren, führt dies zu Störungen im erzeugten Bild. Das Rauschen beeinträchtigt die Qualität der Bilder, indem es ein Linienmuster überlagert, das die

eigentlichen Bilddaten beeinflusst. Ein Verfahren, das diese Störung reduziert oder vollständig beseitigt, hätte das Potenzial, die Diagnostik zu verbessern und potenzielle Fehlerquellen zu verringern.

1.2 Zielsetzung

Ziel der Arbeit ist es, zu erforschen wie die Qualität von Bildern mit Spike-Rauschen durch Deep Learning Algorithmen verbessert werden kann. Dazu soll das theoretische Funktionsprinzip in folgendem Ablaufdiagramm dargestellt werden.

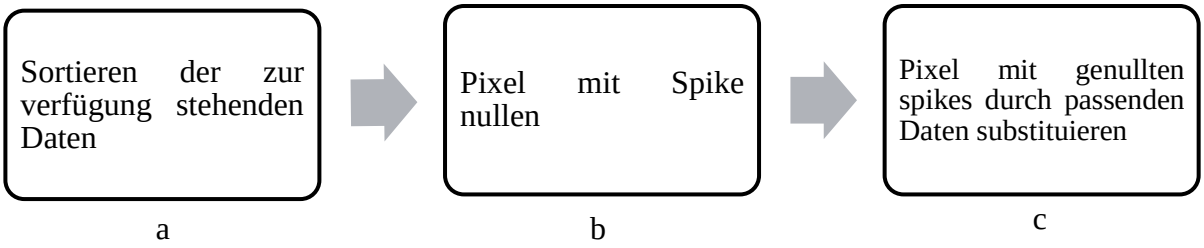


Abbildung 1-1 Ablaufdiagram zur theoretischen Überlegung zur Filterung von Spikes in K-Raum Daten

Die algorithmische Lösung des Problems teilt sich dabei in mehrere Stufen auf, die nacheinander durchlaufen werden sollen. Zunächst soll ein Netzwerk aufgebaut werden, das in der Lage ist, die Existenz von Spikes im Frequenzraum der Bilder zu erkennen (siehe Abbildung 1-1 a). Anschließend soll ein Netzwerk implementiert werden, das die regionale Verteilung der Spikes innerhalb des Frequenzraums erkennt. Danach sollen die betroffenen Pixel auf einen Wert von Null gesetzt werden, um den Beitrag der Frequenz im endgültigen Bild zu entfernen (siehe Abbildung 1-1 b). Die Konsequenz sollte dementsprechend eine Minimierung des durch die Spikes entstehenden Linienmusters sein. Um einen Datenverlust zu vermeiden, sollen in einem letzten Schritt die betroffenen Pixel mittels einer bildgenerativen Methode wieder mit passenden Daten substituiert werden (siehe Abbildung 1-1 c).
Die einzelnen Netzwerke sollen anschließend zu einer Pipeline zusammengesetzt und an realen Spikes getestet werden. Das Ziel ist es dabei, den entstehenden Bildeindruck zu bewerten und

eine Aussage über die Effektivität von Deep Learning Algorithmen für diese Aufgabenstellung zu treffen.

2 Theoretische Grundlagen

2.1 MRT-Grundlagen

Bei der Magnetresonanztomographie handelt es sich um ein nichtinvasives Bildgebungsverfahren, das detaillierte Bilder von Organen und Geweben liefern kann. Wie alle bildgebenden Verfahren basiert es auf der Wechselwirkung externer Felder mit dem menschlichen Körper. Dabei werden starke Magnetfelder eingesetzt, die in der Regel eine Stärke von 0,5 bis 7 Tesla haben, sowie Hochfrequenzimpulse im Radiowellenfrequenzbereich. Im folgenden Kapitel wird für die Erläuterungen das folgende Koordinatensystem in Abbildung 2-1 festgelegt.

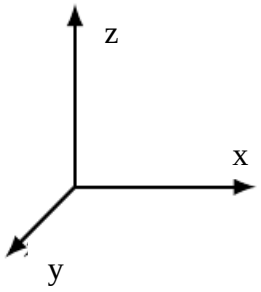


Abbildung 2-1 Festgelgtes Koordinatesystem für folgende Arbeit

2.1.1 Entstehung des MR-Signals

Der Kernspin eines Protons ist eine seiner Basiseigenschaften. Er stellt den Eigendrehimpuls des Protons um seine eigene Achse dar. Die Frequenz dieser Drehung ist bei jedem Element charakteristisch. Das klinische MRT verwendet die Eigenrotation der Wasserstoffatome für die

Bildgebung. Mathematisch wird diese Frequenz durch die sogenannte Larmor-Frequenz beschrieben.

(2-1)

$$\omega_0 = \gamma * B_0$$

Definiert wird die Frequenz dabei durch das gyromagnetische Verhältnis γ und das vorliegende Magnetfeld B_0. Innerhalb eines starken Magnetfeldes (B_0) in z-Richtung des definierten Koordinatensystems richten sich die Wasserstoffprotonen entlang der Magnetfeldlinien aus und präzedieren dabei um die Feldlinien. Dabei kann es zu energieärmeren parallelen oder zur energiereicheren antiparallelen Ausrichtung entlang der Feldlinien kommen, wie in Abbildung 2-2 rechts gezeigt. Ohne anliegendes Magnetfeld, präzedieren die Protonen frei im Raum wie in Abbildung 2-2 links dargestellt.

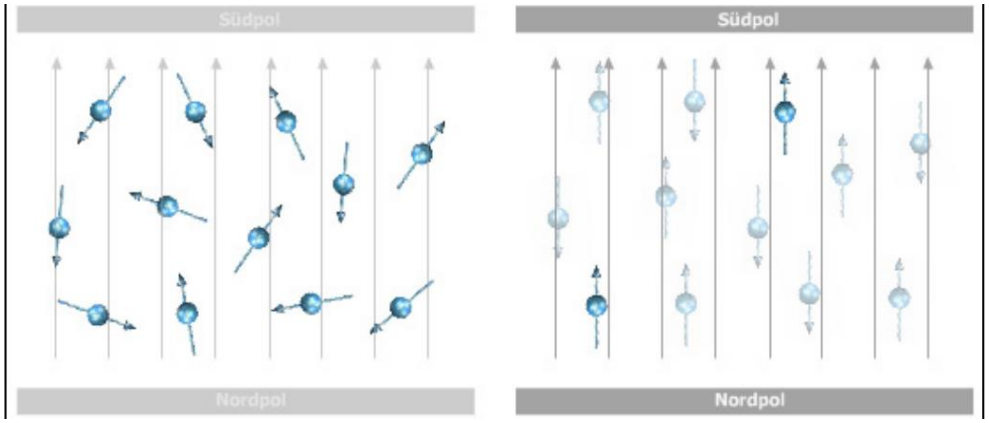


Abbildung 2-2 Parallele und antiparallele Ausrichtung der Spins entlang eines anliegenden Magnetfelds [Pabst13]

Das Verhältnis beträgt hierbei ca. 1.000.007:1.000.000 bei einem äußeren Magnetfeld von 1 Tesla. Aufgrund der großen Anzahl an Protonen im menschlichen Körper reicht dieser Überschuss an parallel magnetisierten Protonen aus, um eine Längsmagnetisierung zu erreichen. Über die Boltzmann-Verteilung kann diese Verteilung mathematisch nachvollzogen werden.

(2-2)

$$\frac{N_-}{N_+} = \exp\left(\frac{\Delta E}{k_B\, T}\right)$$

T definiert dabei die vorliegende Temperatur, ΔE die Energiedifferenz der beiden Niveaus und k definiert die Boltzmann-Konstante. $N-$ bezeichnet die Spins, welche ein kleineres Energieniveau annehmen und parallel zur anliegenden Magnetfeldrichtung zeigen. $N+$ repräsentiert die Spins mit höherem Energieniveau, die antiparallel zur angelegten Magnetfeldrichtung zeigen. Bei dieser Längsmagnetisierung ist zunächst noch kein Vektoranteil in x- oder y-Richtung des Magnetfelds vorhanden.

Wird nun ein Hochfrequenzimpuls (HF-Impuls) eingestrahlt, so kommt zur vorliegenden Längsmagnetisierung M_z eine Quermagnetisierung M_{xy} hinzu. Die Frequenz des HF-Impulses ist dabei identisch mit der vorliegenden Larmorfrequenz. Dadurch kommt es zur Resonanz, und die Spins werden von der Z-Achse in die X-Y-Ebene gekippt. Der Kippwinkel α (siehe Abbildung 2-3) dieser Magnetisierung ist dabei abhängig von der Dauer der Einstrahlung des HF-Impulses. [Dössel16, S. 297-303]

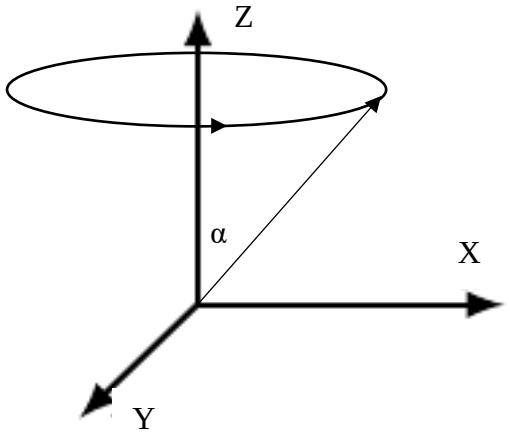


Abbildung 2-3 Kippwinkel α nach einstrahlen des HF-Impulses

Nach Abschalten des HF-Impulses kommt es wieder zur sogenannten Spin-Gitter-Relaxation. Dabei zerfällt die Quermagnetisierung M_{xy} exponentiell, und es wird Energie an die Umgebung abgegeben. Die Spins richten sich wieder nach dem thermischen Gleichgewicht M_0 entlang des

anliegenden B0-Feldes aus. Dieses erneute Ausrichten entlang des anliegenden B0-Feldes wird T1-Relaxation genannt. Es ist definiert über die Zeit, die die Spins benötigen, um sich wieder entlang der Z-Achse auszurichten. Mathematisch wird sie durch die Konstante T1 in folgender Funktion definiert.

(2-3)

$$M_z(t) = M_0 \left(1 - exp^{\left(-\frac{t}{T1}\right)}\right)$$

Ein weiterer Effekt der Einstrahlung des HF-Impulses ist die Phasengleichheit der Spins, die zur Quermagnetisierung M_T führt. Nach Abschalten des HF-Impulses dephasieren die Spins nun wieder exponentiell. Verantwortlich dafür ist nicht, wie bei der T1-Relaxation, das anliegende B0-Feld, sondern die Wechselwirkung zwischen den Spins, die ähnlich wie magnetische Kreisel agieren und zusammenstoßen.

Der Zerfall der Phasengleichheit der Spins wird über die T2-Relaxation beschrieben. Ähnlich der T1-Relaxation folgt auch diese einer exponentiellen Funktion. Aufgrund der unterschiedlichen physikalischen Ursachen der T1- und der T2-Relaxation wird jedoch in dieser Formel eine andere Zeitkonstante verwendet.

(2-4)

$$M_T(t) = M_{T0} \left(1 - exp^{\left(-\frac{t}{T2}\right)}\right)$$

Diese T1- und T2-Relaxationen bewirken eine Änderung des vorliegenden Magnetfelds und folglich ebenfalls eine Wechselspannung, die an einer Spule induziert werden kann. Die induzierte Spannung hängt dabei von der anliegenden Magnetfeldstärke und von der Protonendichte ab, wodurch sich unterschiedliche T1- und T2-Relaxationszeiten ergeben. Diese unterschiedlichen Zeiten können wiederum in unterschiedliche Bildkontraste umgesetzt werden. [Dössel16, S303-305]

2.1.2 K-Raum und Spike Rauschen

Der K-Raum wird definiert als mathematische Repräsentation des Bildraums, also des Frequenzraums der aufgenommenen Bilder. Dabei werden Informationen über die räumliche Verteilung der Signale in Form einer Matrix dargestellt. Durch ein gedachtes Koordinatensystem, das seinen Nullpunkt in der Mitte des K-Raums hat, lassen sich die Frequenzkomponenten darstellen [Elster15]. Ein KRaum mit symbolischen Koordinatensystem ist in Abbildung 2-4 dargestellt

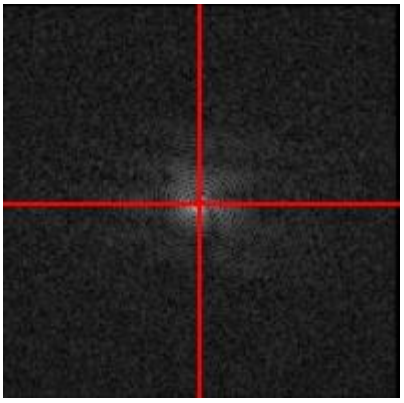


Abbildung 2-4 K-Raum mit imaginären Koordinatensystem

Mittig liegen die kleinsten Frequenzen, mit zunehmendem Abstand steigt diese linear an. Die Intensität der Pixel repräsentiert den Frequenzanteil im Bildraum. Mittig werden folglich die Bildintensitäten definiert, während in den äußeren Bereichen des K-Raums feinere Strukturen des Bildes dargestellt sind. In Abbildung 2-5 ist der originale K-Raum eines Hirnscans dargestellt, während Abbildung 2-6 den identischen Scan mit ausgeblendetem K-Raumzentrum zeigt.

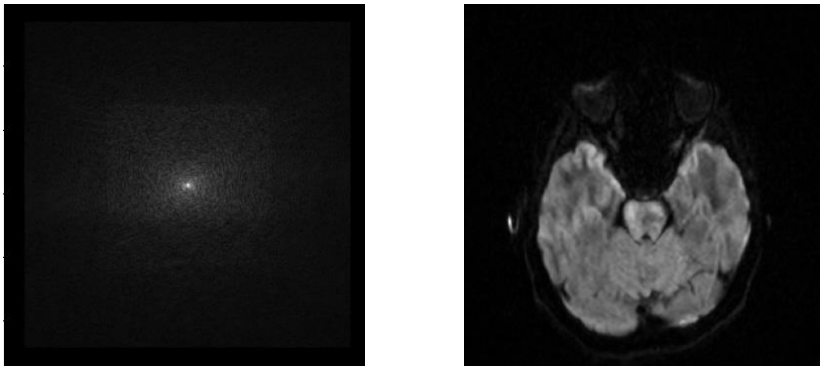


Abbildung 2-5 K-Raum (links) und Bildraum (rechts) eines Hirnscans

In Abbildung 2-6 ist klar zu sehen, dass durch das Entfernen der niedrigen Frequenzanteile größere Flächen mit höherer Bildintensität ausgeblendet werden. Es bleiben die feineren Strukturen zurück, die die Konturen des Scans definieren. In Abbildung 2-7 wird der identische Scan wie in Abbildung 2-5 dargestellt, jedoch mit ausgeblendeten Rändern im K-Raum.

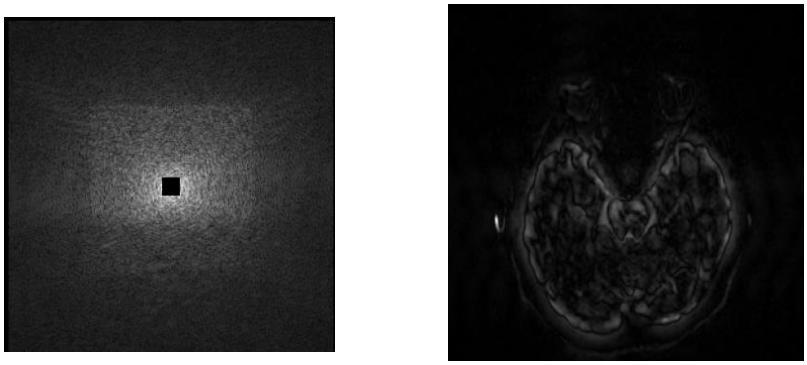


Abbildung 2-6 K-Raum (links) und Bildraum (rechts) eines Hirnscans mit ausgeblendeten K-Raum Zentrum

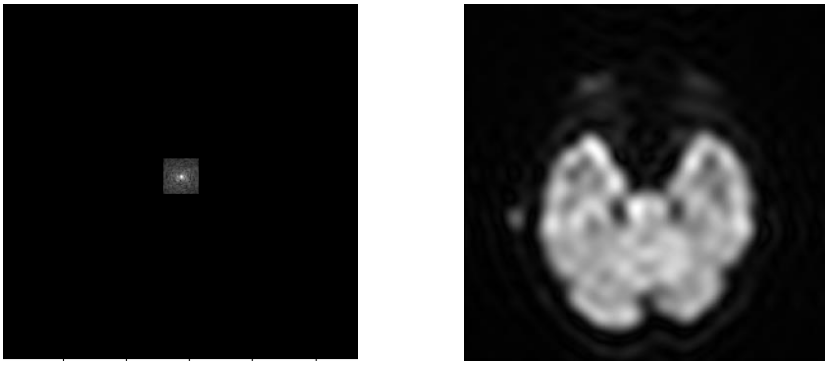


Abbildung 2-7 K-Raum (links) und Bildraum (rechts) eines Hirnscans mit ausgeblendeten Rändern im K-Raum

Durch das Ausblenden der Ränder in Abbildung 2-7 bleiben innerhalb des Bildraums die niedrigen Frequenzanteile zurück. Dadurch werden lediglich größere Flächen dargestellt und es fehlen feinere Strukturen.

Das sogenannte Spike-Rauschen, auch bekannt als Herringbone-Artefakt, ist ein visuell wahrnehmbares Phänomen, das vor allem im K-Raum beobachtet werden kann. Es handelt sich um ein Artefakt, das bei der Magnetresonanztomographie auftritt. Das Spike-Rauschen kann mehrere Ursprünge haben, die im Folgenden erläutert werden. Eine mögliche Ursache für das Spike-Rauschen liegt in Entladungen, die in den synthetischen Fasern von Kleidung auftreten können. Diese Entladungen können durch die Reibung und elektrostatische Aufladung der Kleidung verursacht werden. Aufgrund der Nähe der Kleidung zum Patienten und zur Umgebung des MRT-Geräts können diese Entladungen in Form von Rauschen auf den Bildern sichtbar werden.

Ein weiterer Ursprung des Spike-Rauschens kann in mechanischen Belastungen des Tomographen liegen. Während des MRT-Verfahrens können bestimmte Bewegungen oder Vibrationen des Tomographen auftreten, die durch externe Einflüsse oder interne Faktoren verursacht werden. Insbesondere kann die Belastung des Tomographen zur Vibration der Gradientenspule führen.

Es lässt sich festhalten, dass das Spike-Rauschen multiple Ursachen haben kann, und beispielsweise durch elektrische Entladungen innerhalb der abgeschirmten Kabine oder durch fehlerhafte mechanische Komponenten verursacht werden kann. [Yi-Hsuan00]

Innerhalb des K-Raums zeigen sich die Spikes durch einen oder mehrere Punkte mit höherer Intensität als die umliegende Region. Diese lokal starken Gradienten führen zur Überlagerung des Musters im Bildraum. In Abbildung 2-8 ist eine MRT-Aufnahme eines Gehirns mit überlagertem Spike-Rauschen dargestellt.

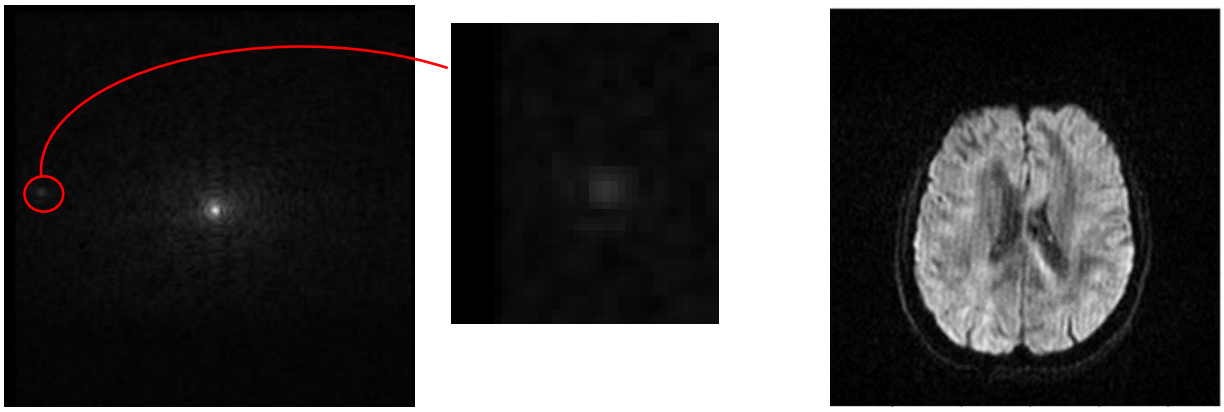


Abbildung 2-8 Spike innerhalb eines K-Raums (links) mit dazugehöriger Bildraumdarstellung (reschts) mit vertikalem streifen Muster überlagert.

In der K-Raum Darstellung des Scans ist mittig links ein Spike zu erkennen. Im rechten Teil der Darstellung ist der daraus Fourier Transformierte Bildraum dargestellt. Hier sind vertikale Streifen zu sehen welche dem eigentlichen Bild überlagert sind.

2.2 Künstliche Intelligenz

Bei Künstlicher Intelligenz (KI) handelt es sich um mathematische Modelle, die durch statistische Wahrscheinlichkeiten in der Lage sind, komplexe Probleme dynamisch zu lösen. Der Aufbau der KI ist vergleichbar mit der Funktionsweise von neuronalen Netzen im Nervensystem von Lebewesen. Neuronen sind durch Verbindungen untereinander in der Lage, Informationen auszutauschen. Ein Neuron erhält in der Regel parallel Informationen durch mehrere Eingänge. Nach Erhalt der Informationen können diese entweder weitergeleitet werden oder nicht. An den Ausgang des Neurons sind wiederum mehrere andere Neuronen angeschlossen, die dann die Informationen dieses und anderer Neuronen erhalten. Durch die dichte Vernetzung und die hohe Anzahl an Neuronen können komplexe Informationen verarbeitet werden. Innerhalb der KI werden diese biologischen Vorgänge numerisch nachgebildet, wobei die Neuronen in ihrer Grundfunktion künstlich simuliert werden.

2.2.1 Perzeptron

Bei den künstlichen Neuronen innerhalb von Neuronalen Netzwerken handelt es sich um sogenannte Perzeptronen. Dabei wird der chemische/elektrische Informationsaustausch zwischen den biologischen Neuronen mathematisch nachgebildet. Das Modell wurde bereits in den 1950er Jahren von Frank Rosenblatt eingeführt [Rosenblatt58]. Das einzelne Perzeptron besitzt mehrere Eingänge und erhält durch diese numerische Werte. Innerhalb des Perzeptrons selbst werden diese eingehenden Werte summiert. Anschließend wird ein Gewicht w multipliziert und dem Ergebnis wird ein Bias b addiert. Gewicht und Bias sind dabei dynamisch und werden innerhalb des Lernprozesses des Netzwerks angepasst. Die realisierte Formel ist in Formel 2-6 abgebildet.

(2-5)

$$y_i = \sum_{i=0}^{n} x_i * w + b$$

Das Ergebnis der Funktion ist zunächst eine einfache lineare Manipulation der eingehenden Daten. Um die bereits erwähnten binären Signalverläufe der biologischen Neuronen nachzubilden, wird eine Aktivierungsfunktion benötigt. Diese Funktion wird bei jedem Knoten einer Schicht verwendet und muss vor Beginn definiert werden. Mit der Aktivierungsfunktion g wird Formel 2-6 wie folgt zu Formel 2-7 erweitert. [Wilmott20, S. 176-179]

(2-6)

$$y_i = g\left(\sum_{i=0}^{n} x_i * w + b\right)$$

Die Aktivierungsfunktion ist dabei frei wählbar. Das Rosenblatt Modell nutzt die einfache Stufenfunktion als Aktivierung nach den Knoten (Formel 2-8)

(2-7)

$$g(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

Die beschriebene Funktion entspricht der biologisch definierten Aktivierungsfunktion. Allerdings ist diese Stufenfunktion für das Training von neuronalen Netzen nicht geeignet. Der Grund dafür liegt darin, dass ihre Ableitung nicht differenzierbar ist, was wiederum die Anwendung des Gradient Descent Algorithmus verhindert. Eine differenzierbare Aktivierungsfunktion ist erforderlich, um den Gradienten berechnen und den Algorithmus effektiv anwenden zu können. Im späteren Verlauf der Arbeit wird der Gradient Descent Algorithmus noch genauer erläutert. Ein weiterer Nachteil dieser rein binären Aktivierungsfunktion besteht in ihrer mathematischen Instabilität. Kleine Änderungen im Eingangssignal können zu erheblichen Veränderungen in der Funktion führen. Darüber hinaus ist es nicht möglich, die Signale probabilistisch abzuschätzen, was den Spielraum für die Gewichtung einzelner Signale einschränkt.

Aufgrund der beschriebenen Probleme werden für die Verwendung in Neuronalen Netzen eher andere Funktionen als Aktivierung verwendet. Verbreitet sind hier beispielsweise die Rectified Linear Unit Funktion (ReLU) [Wilmott20, S181]. (Siehe Abbildung 2-9)

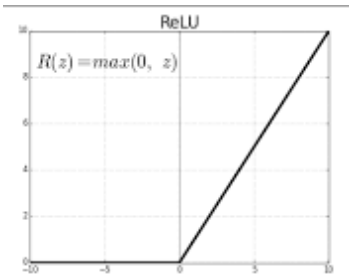


Abbildung 2-9 Rectified Linear Unit Funktionsverlauf

Hier wird das Signal entweder unverändert weitergeleitet oder es bleibt bei einem Wert von 0. (siehe Formel 2-9)

(2-8)

$$g(x) = \max(0, x)$$

Eine weitere mögliche Aktivierungsfunktion ist die Sigmoid Funktion [Wilmott20, S182]. (Siehe Abbildung 2-10)

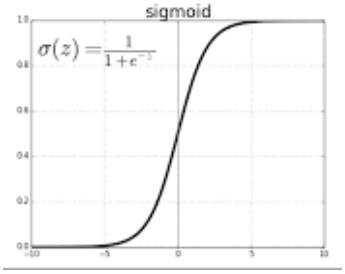


Abbildung 2-10 Sigmoid Funktionsverlauf

Dese ist definiert als eine Stufenfunktion mit differenzierbaren Übergang zwischen den Werten 0 und 1. Eingesetzt werden kann diese Funktion ebenfalls bei der binären Klassifikation von

eingehende Daten. Dabei kann mit einer zusätzlich angewendeten Threshold Funktion unterschieden werden zwischen den Klassen null oder eins. (siehe Formel 2-10)

(2-9)

$$g(x) = \frac{1}{1 + e^{-x}}$$

[Osinga19, S3-4]

2.2.2 Neuronale Netze

Biologische neuronale Netze können durch künstliche neuronale Netze (KNN) nachgebildet werden. Ein einfaches neuronales Netz besteht dabei aus mehreren Schichten oder Layer. Jeder Layer enthält ein oder mehrere Knoten, die die lernenden Neuronen darstellen. In Abbildung 2-11 wird eine einfache Form eines KNN dargestellt.

Eine verbreitete Architektur ist das Fully Connected (FC) Netzwerk. Dabei sind die künstlichen Neuronen innerhalb der Layer parallel geschaltet. Jeder Knoten einer Schicht ist mit jedem Knoten der nachfolgenden Schicht verbunden und gibt somit sein Signal an alle folgenden Knoten weiter. Ein einzelner Knoten erhält als eingehendes Signal die Informationen jedes Knotens der vorherigen Schicht. Aufgrund dieser umfassenden Verbindung der Knoten untereinander werden diese Netzwerke als vollständig vernetzt oder Fully Connected (FC) bezeichnet.

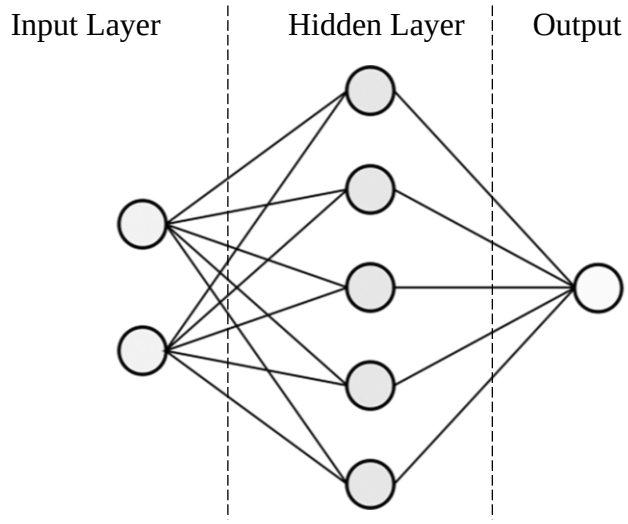


Abbildung 2-11 Darstellung eines einfachen Fully Connected Neuronalen Netzes mit zwei Neuronen als Input fünf als Hidden Layer und einen innerhalb der Output Layer [Tsutis23]

In ihrer einfachsten Ausführung sind die Layer innerhalb des Netzwerks hintereinandergeschaltet, und die Informationen fließen von einer eingehenden Schicht, dem Input Layer, durch die versteckten Schichten, den Hidden Layern, hin zur ausgebenden Schicht, dem Output Layer. Die Anzahl der Knoten in der eingehenden Schicht wird dabei über die Dimensionen der eingehenden Informationen definiert. In Abbildung 2.11 werden zum Beispiel zwei Werte als eingehende Parameter genutzt. Die Anzahl der Hidden Layer insgesamt und die Anzahl der Knoten in diesen Schichten sind frei wählbar. In Abbildung 2.11 ist eine versteckte Schicht mit fünf Knoten dargestellt. Die Output Layer wird über die Anzahl der gewünschten Ausgaben definiert. In Abbildung 2.11 ist ein Knoten realisiert.
Im Kontext des Deep Learning können verschiedene Aufgabenmuster durch die definierten Netzwerke unterschieden werden. Innerhalb dieser Arbeit liegen die Aufgabentypen Regression und Klassifikation vor. Bei der Regression wird versucht, auf Basis der eingehenden Daten kontinuierliche Werte vorherzusagen und anzunähern. Im Gegensatz dazu wird bei der Klassifikation versucht, eingehende Datenpunkte in vorher definierte Kategorien einzuordnen. Jede Kategorie wird dabei durch einen vorher definierten numerischen Wert repräsentiert. [Wilmott20, S173-175]

Nach Definition der Aktivierungsfunktionen innerhalb der Schichten, kann das Netzwerk in einem Training auf definierten Eingangsdaten verwendet werden. Das übergeordnete Ziel ist dabei eine Funktion möglichst gut zu approximieren. Dabei werden die eingehenden Daten x_n mathematisch abgeändert und resultierende Werte $\hat{y}_n$ ausgegeben. Es ist erforderlich, dass für die eingehenden Daten korrespondierende Daten y_n vorhanden sind. Das Ziel des Netzwerks ist es nun das die ausgehenden Daten $\hat{y}_n$bestmöglich die Zieldaten y_n approximieren. Um eine numerische Aussage über die Differenz zwischen $\hat{y}_n$ und y_n zu treffen wird eine sogenannte Kostenfunktion benötigt, welche die Kosten oder auch loss definiert. In der Regression wird als klassische Kostenfunktion der Mean Squared Error (MSE) verwendet welcher in Formel 2-11 definiert ist.

(2-10)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Liegt stattdessen ein Klassifikationsproblem mit zwei Kategorien vor, kann die Binäre Kreuzentropie (BCE) verwenden werden. (Formel 2-12)

(2-11)

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log\big(p(y_i)\big) + (1 - \hat{y}_i) * \log\,(1 - p(y_i))$$

Nach der Berechnung der Differenz zwischen dem Output $\hat{y}_n$ und der Zielgröße y_n kann innerhalb der sogenannten Backpropagation das Netzwerk angepasst werden. Dabei wird das Gewicht w und die der Bias b so angepasst das die Kostenfunktion J minimiert werden kann. Dafür wird zunächst die partielle Ableitung von J gebildet. (siehe Formel 2-12 und 2-13)

(2-12)

$$\frac{\partial J}{\partial w_{i,j'}^{(l)}}$$

(2-13)

$$\frac{\partial J}{\partial b_{j'}^{(l)}}$$

Nach Berechnung des Loss innerhalb der Ausgabeschicht kann der Fehler durch die Hidden Layer propagiert werden. Dieser Vorgang wird als Backpropagation bezeichnet. Mit Hilfe des Gradientenabstiegsverfahrens können die Gewichte w und der Bias b der Layer angepasst werden. Dafür werden w und b zunächst initialisiert. Anschließend werden beide Parameter schrittweise aktualisiert, um die Kostenfunktion in Richtung eines Minimums zu bewegen (siehe Formel 2-14).

(2-14)

$$neues\ w_{i,j'} = altes\ w_{i,j'} - \beta * \frac{\partial J}{\partial w_{i,j'}}$$

Die Größe der Anpassung in einem neuronalen Netz wird durch den Lernfaktor β definiert. Die Auswahl des richtigen β-Werts ist entscheidend für den Erfolg des Trainings. Ist der Lernfaktor zu groß gewählt, besteht die Gefahr, dass das globale Minimum der Funktion übersprungen wird. Ist er hingegen zu klein, kann die Anpassung des Modells sehr lange dauern. Die schrittweise Optimierung der Modellparameter wird fortgesetzt, bis die Funktion gegen ein globales Minimum konvergiert. Beim Training von neuronalen Netzen wird häufig das sogenannte Batch-Gradientenabstiegsverfahren verwendet. Dabei werden die Gewichte nicht nach jedem einzelnen Datenpunkt aktualisiert, sondern basierend auf dem Durchschnittswert einer Gruppe von Datenpunkten, einem sogenannten Batch. Die Aufteilung der Daten in Batches ist deutlich weniger zeitaufwendig als die Aktualisierung für jeden einzelnen Datenpunkt, insbesondere bei großen Datensätzen. Die Größe der Datengruppe wird durch die sogenannte Batchsize festgelegt.

Durch die Verwendung des Batch-Gradientenabstiegsverfahrens und die Anpassung der Batchsize kann das Training effizienter gestaltet werden, insbesondere bei großen Datensätzen. [Wilmott20 S184-188]

Die Initialisierung der Perzeptronen kann unterschiedlich erfolgen. Eine verbreitete Art ist die randomisierte Zuweisung, dabei wird den Perzeptronen zufällig auf Basis einer bestimmten Verteilung ein Wert zugewiesen. Eine weitere Möglichkeit ist die sogenannte Initialisierung von Xavier Glorot und Yoshua Bengio aus dem Jahr 2010 [Glorot10]. Diese Methode berücksichtigt die Größe der Eingabe und Ausgabe eines jedes Neurons. Die Idee hinter der Initialisierung besteht darin, die Gewichte so zu initialisieren, dass der Durchschnitt der Varianz der Eingangs- und Ausgangssignale eines Neurons gleich ist. Durch diese symmetrische Initialisierung wird sichergestellt, dass die Größenordnung der Signale während des Vorwärts- und Rückwärtsdurchlaufs des Netzwerks stabil bleibt. Die Xavier-He-Initialisierung berücksichtigt die Anzahl der Eingangs- und Ausgangsverbindungen eines Neurons, um die geeignete Skalierung der Gewichte zu bestimmen.

Um den Gradientenabstieg effektiver zu gestalten, werden in den meisten Optimierungsalgorithmen zusätzliche Optimierungsschritte implementiert. In dieser Arbeit wird insbesondere der Adam-Algorithmus (Adaptive Moment Estimation) zur Optimierung der Netzwerke verwendet. Dieser Algorithmus wurde von Kingma und Ba auf der ICLR-Konferenz 2015 vorgestellt [Kingma15]. Der Adam-Algorithmus passt den Faktor der Gewichtsaktualisierung basierend auf den berechneten Gradienten an. Er berechnet sowohl einen exponentiell abklingenden Durchschnitt vergangener Gradienten als auch einen exponentiell abklingenden Durchschnitt der vergangenen quadrierten Gradienten.

Zusätzlich wird auch die Variante NAdam des Optimierers verwendet, die von Dozat im Rahmen eines Workshop-Tracks auf der ICLR-Konferenz 2016 vorgestellt wurde [Dozat16]. NAdam kombiniert den Adam-Algorithmus mit der beschleunigten Gradientenmethode von Nesterov. Dabei wird ein zusätzlicher Schritt eingeführt, um den exponentiell gleitenden Durchschnitt vergangener Gradienten zu berechnen. Dadurch kann der Gradient an einem zukünftigen Punkt abgeschätzt werden. Diese Maßnahme beschleunigt die Konvergenz und reduziert Oszillationen.

2.2.3 Faltungs Neuronale Netze

Bild- oder Videodaten bestehen aus Matrizen, die eine Vielzahl von Datenpunkten (Pixeln) enthalten. Aufgrund der enormen Menge an Eingangsdaten ist es nicht effizient, sie mit den zuvor beschriebenen vollständig vernetzten Netzwerken zu trainieren. Stattdessen werden Faltungsneuronale Netze oder Convolutional Neural Networks (CNN) verwendet, die speziell für die Verarbeitung von Bildern entwickelt wurden. Diese Netzwerke arbeiten mit einer deutlich geringeren Anzahl an Parametern und sind daher effektiv für die Verarbeitung von Bilddaten geeignet. Die grundlegenden Bausteine von CNNs sind die sogenannten Convolutional Layer oder Faltungsschichten. Diese Schichten verwenden Perzeptronen, die in zweidimensionalen oder dreidimensionalen Matrizen, auch Kernels genannt, implementiert sind. Ein Kernel wird durch seine Höhe, Breite und sein sogenanntes rezeptives Feld definiert. Durch wiederholtes Verschieben des Kernels über das gesamte Bild und eine Matrixmultiplikation mit den überlappenden Pixeln wird die Eingangsmatrix manipuliert und der zentrale Pixel des Kernels neu berechnet. Dadurch werden Merkmale oder Features aus den Eingangsbildern extrahiert. Eine Veranschaulichung dieses Prozesses ist in Abbildung 2-12 dargestellt [Osinga19, S5].

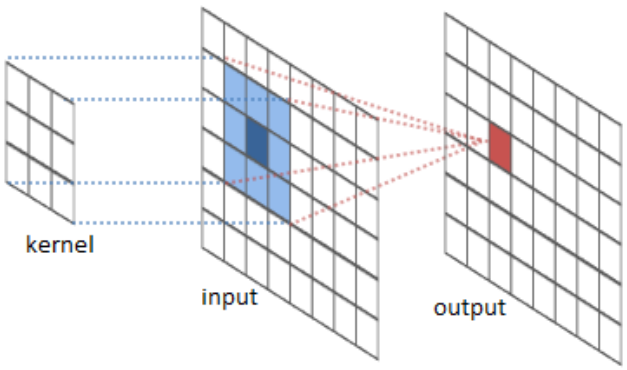


Abbildung 2-12 Beispielanwendung einer 3x3 Convolution auf einer eingehednen Matrix [Pohrkel23]

Dazu werden jeweils alle Pixel des Kernels mit einem Ausschnitt der eingehenden Matrix multipliziert. Das Ergebnis wird summiert und stellt den neuen Wert des Pixels in der eingehenden Matrix dar. Die Schrittweite, mit der der Kernel über die Matrix iteriert, wird als

Stride bezeichnet. Da am Rand der Eingangsmatrix Daten für die Convolution fehlen, werden diese durch das sogenannte Padding künstlich hinzugefügt. Dabei wird die Input-Matrix erweitert, sodass der Kernel auf alle Pixel angewendet werden kann. [Heaton15, S201]
Ein einzelner Kernel stellt einen Filter dar, der eine Feature Map der eingehenden Daten erzeugt. Die Convolutional Layer sind anders als die Fully Connected Layer nicht mit allen Neuronen der vorherigen Schicht verbunden, sondern erhalten die ausgegebenen Feature Maps als Input. [Osinga19, S5]

Durch die Manipulation der Convolutional Layer lassen sich verschiedene Merkmale wie Linien, Kanten oder Punkte aus dem eingehenden Bild extrahieren. Je mehr Filter verwendet werden, desto mehr unterschiedliche Features können extrahiert werden. [Heaton15, S200]
Da jeder Kernel durch sein rezeptives Feld begrenzt ist, können zunächst nur Merkmale innerhalb dieses Felds erkannt werden. Um größere und komplexere Merkmale eines Bildes zu erfassen, wird das sogenannte Subsampling verwendet. Dabei wird die Größe der eingehenden Bildmatrix um einen festgelegten Faktor reduziert. Durch die Reduktion der Filteroperationen auf einer kleineren Matrix können Ressourcen wie Rechenzeit und Rechenkapazität eingespart werden.

Eine verbreitete Methode zur Reduktion der Bildgröße ist die sogenannte strided Convolution. Dabei wird in einer Convolutional Layer ein Stride größer eins festgelegt, sodass einige Pixel innerhalb der Matrixmultiplikation übersprungen werden. Dadurch wird eine Feature Map generiert, deren Größe um den Faktor Stride kleiner ist als die eingehende Matrix.
[Osinga19, S6]
Eine weitere Methode zur Größenreduktion ist das sogenannte Pooling der eingehenden Daten. Dabei handelt es sich um eine separate Schicht in einem Netzwerk, die vor oder nach der Convolutional Layer implementiert werden kann. Die Pooling Layer haben keine zu trainierenden Gewichte. Es muss ebenfalls die Größe des Kernels und der Stride definiert werden. Eine verbreitete Art ist das sogenannte Max-Pooling. Hierbei wird jeweils lediglich der maximale Werte innerhalb der eingehende Matrix für den Output verwendet. [Heaton15, S203-204] (siehe Abbildung 2-13)

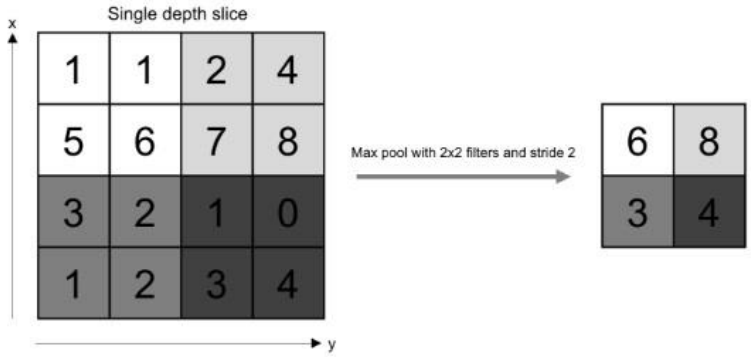


Abbildung 2-13 MaxPooling Operation mit einer 2x2 Matrix und einem stride von zwei [MatlabOne23]

Bei der Verwendung besonders tiefer Neuronaler Netze kann es zum sogenannten Vanishing Gradient Problem kommen. Dabei werden die Gradienten zur Aktualisierung der Gewichte nicht ausreichend durch alle Layer propagiert. Dadurch wird der Gradient und damit die Änderung der Gewichte in den anfangs gelegenen Schichten sehr gering [Wang18]. Die untere Grafik (Abbildung 2-14) zeigt beispielhaft das Training eines tiefen Neuronalen Netzwerks auf dem CIFAR-10 Datensatz. Ein Netzwerk besteht dabei aus 20 Layern, während ein Vergleichsnetzwerk 56 Layer hat.

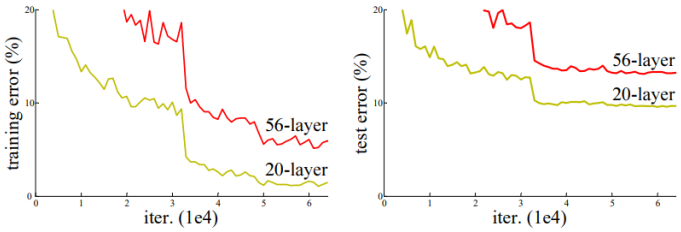


Abbildung 2-14 Beispielhafter Vergleich von zwei Neuronalen Netzwerken mit 56 Layer und 20 Layer. Der Test- und Trainingserror fällt bei den flacheren Neuronalen Netz kleiner aus als bei dem tieferen. [He15 S1]

Klar zu sehen ist dabei, dass das flachere Netzwerk einen geringeren Test- und Trainingsfehler aufweist als das tiefere Netzwerk. Eine mögliche Lösung für das Problem sind die sogenannten Residuals. Diese sind in der Lage, vor allem das Training von tiefen neuronalen Netzen positiv

zu beeinflussen. Dabei wird dem Netzwerk ermöglicht, die Identität der eingehenden Feature Maps zu erlernen und diese weiterzuleiten. Ein möglicher Aufbau eines solchen Residual Blocks ist in Abbildung 2-15 dargestellt.

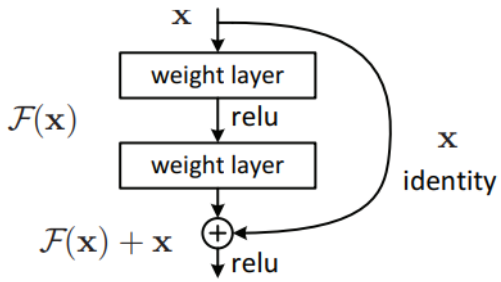


Abbildung 2-15 Aufbau eines beispielhaften residual Blocks. Die Identität der eingehenden Matrix wird dabei parallel die Faltungsschichten geleitet und mit dem Output der zweiten Convolution addiert. Die Summe wird der zweiten Aktivierungsschicht zugeführt.

Dabei wird zunächst von der Annahme ausgegangen, dass mehrere nichtlineare Schichten hintereinander eine Mapping-Funktion $H(x)$ erlernen könnten, wobei x den Input dieser Layer darstellt. Es wäre eine äquivalente These, dass die Funktion ebenfalls die Funktion der Residuals $H(x) = x$ ausreichend gut approximieren könnte. Eine Alternative wäre hierbei das Erlernen einer Restfunktion $F(x) + x = H(x)$. Das Ergebnis wäre hierbei äquivalent zu der ersten Aussage, und beide Funktionen sollten die Funktion der Residuals approximieren können. Die Praxis zeigt uns jedoch, aufgrund des Vanishing Gradients, dass die komplette Approximierung der eingehenden Funktion in der Praxis weniger gut funktioniert als das Erlernen der Restfunktion. Durch die Residuals können somit die Gradienten einfacher durch das Netzwerk propagiert werden, was die Problematik der Optimierung von tiefen neuronalen Netzen erheblich reduziert [He15 S2].

Eine Möglichkeit, das Training von neuronalen Netzen zu beschleunigen und zu stabilisieren, kann durch die sogenannte Batch Normalization (BN) Layer erreicht werden. Die Layer wurden 2015 von Sergey Ioffe und Christian Szegedy von Google vorgestellt [Ioffe15]. Diese Schicht wirkt dem sogenannten Covariate Shift einer lernenden Layer entgegen. Dieser wird definiert als die Veränderung der Verteilung der Aktivierungsfunktion innerhalb dieser Layer. Die BN-

Layer sind als separate Layer nach einer Convolutional Layer oder auch einer Fully Connected Layer zu integrieren. Dabei wird die Aktivierung einer Layer normalisiert, indem über die momentane Batch an Eingangsdaten x die Standardabweichung σ und der Mittelwert μ der Aktivierungen berechnet werden. Anschließend werden die Werte der Aktivierung durch eine lineare Transformation normalisiert, sodass diese einer Normalverteilung folgen (siehe Formel 2-15).

(2-15)

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu}{\sqrt{\sigma^2}}$$

Zusätzlich werden zwei trainierbare Parameter γ und β eingeführt welche die Verteilung transformieren. Über β wird ein Bias spezifiziert, mit γ die Standartabweichung der Verteilung (siehe Formel 2-16).

(2-16)

$$y_i = \gamma * \hat{x} + \beta$$

3 Stand der Technik

3.1 Detektion von Spikes

Ein bildbasierter Ansatz zur Detektion von Spikes wurde bereits im Jahr 1986 von M. Staemmler und K. Gersonde entwickelt [Staemmler86]. Dieser Ansatz bildet die Grundlage für eine Vielzahl nachfolgender Detektionsalgorithmen [Xiadong01, Yi-Hsuan00, Campbell-Washburn16]. Im Echoformanalyse-Verfahren von Staemmler werden für jeden Intensitätswert die umgebenden 20 Werte betrachtet, bestehend aus 10 vorangehenden und 10 folgenden Werten. Diese Werte werden summiert und durch 5 geteilt, um eine Grenze für den Intensitätswert festzulegen. Dieser Werte kann als ein Parameter für die Sensibilität der Detektion verstanden werden. Wenn ein Intensitätswert diese Grenze überschreitet, wird er als Spike identifiziert und entfernt. Es kommt häufig vor, dass nicht nur einzelne Spikes, sondern auch mehrere Spikes in einem Echo-Signal auftreten. In solchen Fällen wird das beschriebene Verfahren sequenziell angewendet, um die mehrfachen Spikes zu erkennen und zu entfernen. Im Gegensatz zur Behandlung einzelner Spikes werden mehrere Spikes durch ihre jeweiligen Grenzwerte ersetzt.
Um die Berechnungszeit zu optimieren, wird die Echoformanalyse durch den Einsatz von zwei Algorithmen für verschiedene Teile des Echo-Signals verbessert. Der zentrale Teil des Echos enthält Intensitätswerte, die signifikant von der Rauschschwelle abweichen, während die Randbereiche des Echos Werte innerhalb der Rauschschwelle darstellen. Durch die Anwendung spezifischer Algorithmen auf diese unterschiedlichen Bereiche kann die Effizienz der Spike-Detektion weiter gesteigert werden.

Ein algorithmischer Ansatz, der auf der statistischen Analyse des Zeitsignals der Daten beruht, wurde 2001 von Xiadong Zhang et al. vorgestellt [Xiadong01]. Der Algorithmus geht zunächst

von der Annahme aus, dass bei der funktionellen Magnetresonanztomographie (fMRT) mehrere Hundert Bilder desselben Schnittes erzeugt werden. Bei diesen Schnittbildern kann davon ausgegangen werden, dass zwischen den aufeinanderfolgenden Bildern nur geringfügige Unterschiede auftreten und dass der Zeitverlauf der k-Raum-Daten relativ stabil ist, solange nur zufälliges Rauschen vorhanden ist. Basierend auf dieser Beobachtung erfolgt die Erkennung von Spikes durch die Punkt-für-Punkt-Analyse des k-Raum-Zeitverlaufs.
Zur Detektion von Spikes innerhalb des K-Raums wird ein Schiebefensteroperator angewendet. Um die Komplexität des Algorithmus zu reduzieren, wird ausschließlich der Absolutwert der eingehenden Daten berücksichtigt. Zunächst wird die Standardabweichung des thermischen Rauschens für jeden Datenpunkt im zeitlichen Verlauf der Schnittbilder geschätzt, wobei das Ergebnis mit dem Symbol σ bezeichnet wird. Für jeden Zeitpunkt i wird der Durchschnitt der vorangegangenen 10 Zeitpunkte berechnet, um eine Schätzung des lokalen Mittelwerts μ_i zu erhalten. Anschließend erfolgt ein Vergleich der Magnitude des betrachteten Zeitpunkts m_i mit dem lokalen Mittelwert. Wenn die folgende Funktion (3-1) erfüllt ist:

(3-1)

$$m_i - \mu_i > (0{,}1\,\mu_i + 3\sigma)$$

Ist die Funktion erfüllt, wird der Punkt i als Spike identifiziert. Dieser Vorgang wird für den nächsten Zeitpunkt wiederholt, bis alle Zeitpunkte verarbeitet wurden. Der Ansatz ist jedoch nur für spezielle MRT-Verfahren (funktionelles MRT) verwendbar. Aus diesem Grund lässt sich hier kein generalisierbarer Ansatz ableiten.

Ein aktuellerer Ansatz aus dem Jahr 2016 [Campbell-Washburn15] nutzt eine Robust Prinicipal Component Analysis (RPCA) zur Detektion von Spikes. Dabei wird eine gemessene K-raum Matrix M in eine niederrangige Matrix L und eine dünne Matrix S aufgeteilt. Basierend auf den beiden entstandenen Matrizen wird anschließend folgendes Optimierungsproblem (siehe 3-2) gelöst.

(3-2)

$$min_{L,S}||L||_* + \lambda||S||_1$$

Dabei wird folgende Annahme getroffen:

(3-3)

$$M = L + S$$

$||.||_*$Stellt dabei die nuklear Norm der der Matrix da und $||.||_1$ die L1-Norm. Im Fall von RF-Spike-Rauschen repräsentiert M die gemessenen Daten, S stellt die starken RF-Spikes dar und L repräsentiert die wiederhergestellten artefaktfreien K-Raum-Daten. Die Methode teilt die Daten in eine dünne Komponente und eine niederrangige Komponente auf. Bei mehreren Bildern enthält die dünne Komponente die Veränderungen von Bild zu Bild, die nicht durch die niederrangige Komponente erklärt werden können. Bei der Analyse eines einzelnen Bildrahmens werden die Daten zeilenweise betrachtet.

3.2 Filterung von Spikes

Bei der Filterung der Spikes innerhalb des ersten Algorithmus wird der als Spike identifizierte Wert durch einen neuen Wert ersetzt. Dabei wird der Durchschnitt der drei vorangegangenen Zeitwerte ersetzt. Bei dem Verfahren von Staemmler wird der identifizierte Spike zunächst gelöscht. Anschließend wird für einen Spike ein neuer Intensitätswert berechnet, der den Durchschnittswert des vorangehenden und folgenden Werts darstellt und den gelöschten Spike ersetzt.

4 Daten

Für diese Arbeit steht ein festes Kontingent an Datensätzen zur Verfügung. Dabei handelt es sich um MRT-Aufnahmen, innerhalb derer Spikes registriert wurden. In dem folgenden Kapitel sollen der Datenbestand sowie die Datenvorverarbeitung behandelt werden. Die Daten stammen aus Studien mit Probanden innerhalb von Siemens Healthineers.

4.1 Datenanalyse

Insgesamt stehen für diese Arbeit 6563 MRT-Einzelschichten zur Verfügung. Die Daten zeigen dabei MRT-Scans der oberen Thorax Region (Abbildung 4-1 links) und des Gehirns (Abbildung 4-1 rechts) verschiedener Probanden. Die Scans stehen dabei als Bildraumdaten zur Verfügung.

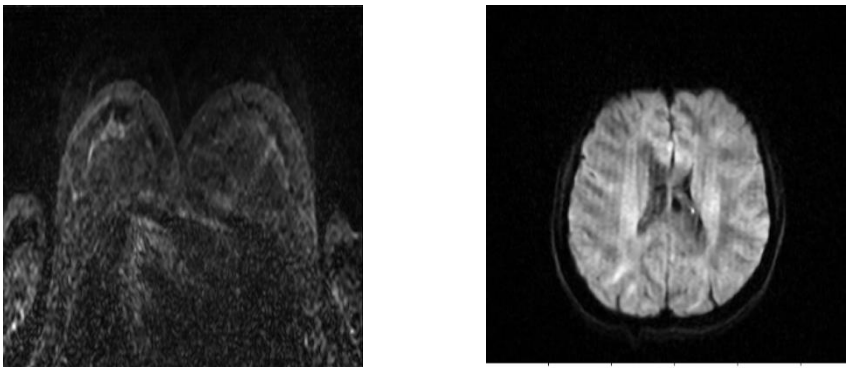


Abbildung 4-1 Beispieldaten von Thorax und Hirn MRT-Scans

Innerhalb der vorliegenden Daten sind zwei Körperregionen als MRT-Scans für das Training verfügbar. Wenn diese Bilddaten als Trainingsdaten verwendet würden, könnte es zu einer mangelhaften Generalisierung kommen, wenn das Netzwerk auf andere Körperregionen angewendet wird. Da der Aufbau des K-Raums jedoch weitestgehend einheitlich ist, sollte die Verwendung der Netzwerke auf anderen Objekten oder Körperregionen kein Problem darstellen. Zur Weiterverarbeitung wird das binäre Label "Spike" und "Clean" verwendet. Die Verteilung der Labels aus dem vorhandenen Datensatz ergibt sich wie folgt (siehe Abbildung 4-2).

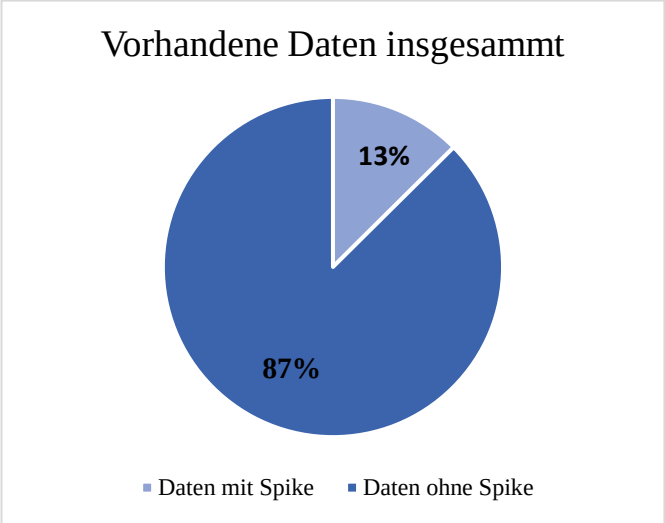


Abbildung 4-2 Numerische Verteilung der Label in Prozent

Das Label "Spike" bezeichnet die Schichten, in denen Spikes erkennbar sind. In den Schichten mit dem Label "Clean" sind optisch keine Spikes zu erkennen. Abbildung 4-2 zeigt deutlich, dass der Datensatz stark unausgeglichen ist. 87% der Daten (5709 Daten) sind als "Clean" gelabelt, während 13% der Daten (853 Daten) einen Spike enthalten. Diese Ungleichgewichtung muss beim Training und der Validierung der folgenden Algorithmen berücksichtigt werden. Aufgrund der großen Anzahl an Basisdaten kann jedoch davon ausgegangen werden, dass genügend Daten für das Training von Neuronalen Netzen zur Verfügung stehen.

Die Höhe und Breite der bereitgestellten Scans variiert stark innerhalb der Daten. Das untere Diagramm (Abbildung 4-3) visualisiert die Verteilung der Matrixgrößen.

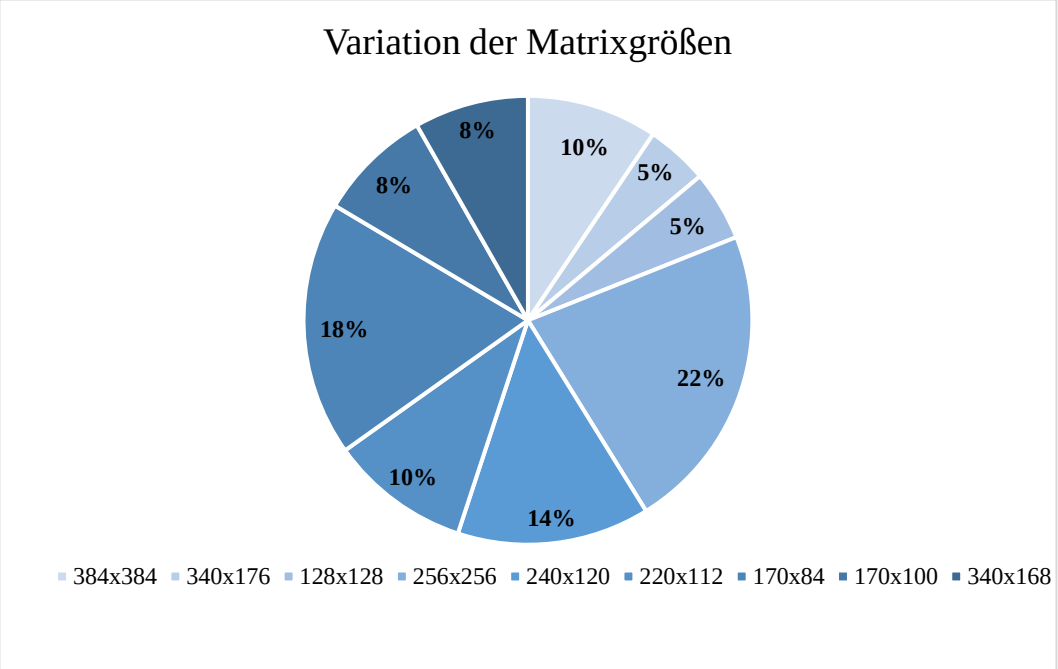


Abbildung 4-3 Übersicht über die Variation der Bildgröße

Abbildung 4-3 zeigt, dass die zur Verfügung stehenden Scans in ihrer Größe variabel ausfallen. Die Bilddaten MRT sind in der Regel ebenfalls frei wählbar. Demzufolge muss auf Basis der Datenlage keine Bildmatrixgröße gewählt werden.

4.2 Datenvorbereitung

Auf Basis der vorangegangenen Datensichtung wird eine Vorverarbeitung eingerichtet, um die Daten für das nachfolgende Training zu konfigurieren. Zu Beginn liegen die Daten als komplexe Bildraumdaten innerhalb eines zweidimensionalen Arrays vor, aufgrund der Vorverarbeitung des MRT-Aufnahmesystems.

Die verwendeten KI-Algorithmen zur Detektion, Segmentierung und Transformation der Spikes sollen mit dem K-Raum der MRT-Scans arbeiten. Daher werden die Bildraumdaten auf Basis der Anforderung durch eine zweidimensionale schnelle Fourier-Transformation in den Frequenzraum transformiert. Durch das komplexe Format der Daten ergeben sich spezifische Datenformate, die als Eingabe für die Algorithmen genutzt werden können. Für diese Arbeit sollen mehrere Datenformate parallel aufgebaut werden, da im Vorfeld nicht eindeutig gesagt werden kann, welches Datenformat innerhalb des Trainings der neuronalen Netze die besten Ergebnisse liefert.

Zum einen wird Real- und Imaginärteil getrennt als erstes Datenformat gesichert. Dabei wird der Imaginärteil der Daten in reelle Daten überführt und separat gespeichert. Damit ergibt sich ein zweidimensionales Array als Eingangsformat (siehe Abbildung 4-4).

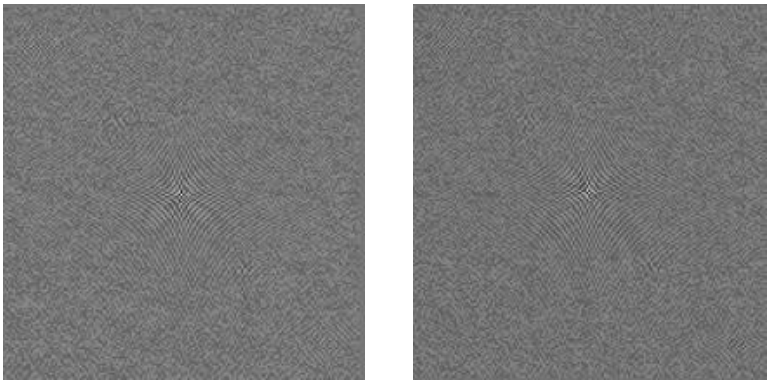


Abbildung 4-4 Beispiel K-Raum als Real (links) und Imaginärteil (rechts) dargestellt

Als zweites Datenformat wird der Absolutwert des K-Raums gesichert. Die Daten werden hierbei als eindimensionale Format gesichert (siehe Abbildung 4-5).

Abbildung 4-5 Beispiel K-Raum als Absolutwert dargestellt

Rein optisch können innerhalb dieser Daten die Spikes nur schwer identifiziert und lokalisiert werden. Dies erschwert das Labeln der Daten. Aus diesem Grund wird eine vierte Datengrundlage eingeführt, die experimentell ermittelt wurde. Dabei wurde ein Datenformat entwickelt, das die visuelle Sortierung der Daten und die Lokalisierung der Spikes vereinfacht. Bei diesem Format wird zunächst die L1-Norm berechnet, also die Summe von Real- und Imaginärteil, und anschließend in den Frequenzraum transformiert. Der Absolutwert des errechneten Frequenzraums wird verwendet (siehe Abbildung 4-6).

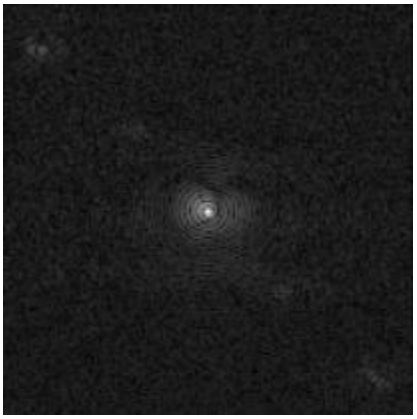


Abbildung 4-6 Beispiel K-Raum als L1 Wert dargestell

Innerhalb des in Abbildung 4-6 dargestellten K-Raums sind die vier vorhandenen Bereiche mit Spikes deutlich zu erkennen. Der Wertebereich der Pixel in den verwendeten K-Räumen ist nicht einheitlich und muss angepasst werden. Durch die Normalisierung der Daten wird sichergestellt, dass Merkmale mit unterschiedlichen numerischen Größen gleich gewichtet werden. Ein weiterer Effekt ist die Verbesserung der Generalisierung des Netzwerks. Der verwendete Wertebereich der K-Räume liegt in dieser Arbeit zwischen 0 und 1.

Zusätzlich müssen die Daten eine einheitliche Matrixgröße besitzen, um sie als Input für die neuronalen Netze nutzen zu können. Dabei wird eine Bildgröße verwendet, die dem Anspruch genügt, mehrfach ohne Rest durch 2 teilbar zu sein, um die Bildtransformationen innerhalb der Segmentierungsalgorithmen zu ermöglichen. Zusätzlich sollen die Matrizen möglichst klein gewählt werden, um die Trainingszeiten der Netzwerke zu minimieren. Die Spikes innerhalb der K-Räume beschränken sich auf eine Region mit einem Abstand von maximal 135 Pixeln vom K-Raumzentrum entfernt. Aus den Anforderungen ergibt sich eine optimale Größe von

288x288 Pixeln. Die Bildgrößen der Schichten mit kleinerem Ausmaß werden hierfür durch Zero Padding auf die entsprechende Größe erweitert. Größere Matrizen werden auf die entsprechende Größe zugeschnitten.
Zu Beginn wird zusätzlich ein Datensplit vorgenommen, um einen Teil der Daten für den Testdatensatz zurückzuhalten. Dieser muss den Anspruch genügen, disjunkt von den Trainingsdaten zu sein. Insgesamt stehen 6563 Daten von K-Räumen zur Verfügung. Als Testset für den Detektionsalgorithmus werden randomisiert 20% der Daten jeder Klasse verwendet.

Um die Segmentierungsalgorithmen zu trainieren, wird eine Grundwahrheit (Maske) als Zielgröße benötigt. Die verwendete Maske und der eingehende K-Raum müssen dabei die identische Größe besitzen. Innerhalb der Maske heben sich bestimmte Regions of Interest (ROI) durch eine erhöhte Pixelintensität vom Rest der Matrix ab. Dazu werden Pixel, die einen Spike beinhalten, mit einer Intensität von 1 gelabelt. Die übrige Matrix besitzt einen Wert von 0. In Darstellung 4-7 wird ein K-Raum mit der dazugehörigen Maske dargestellt.

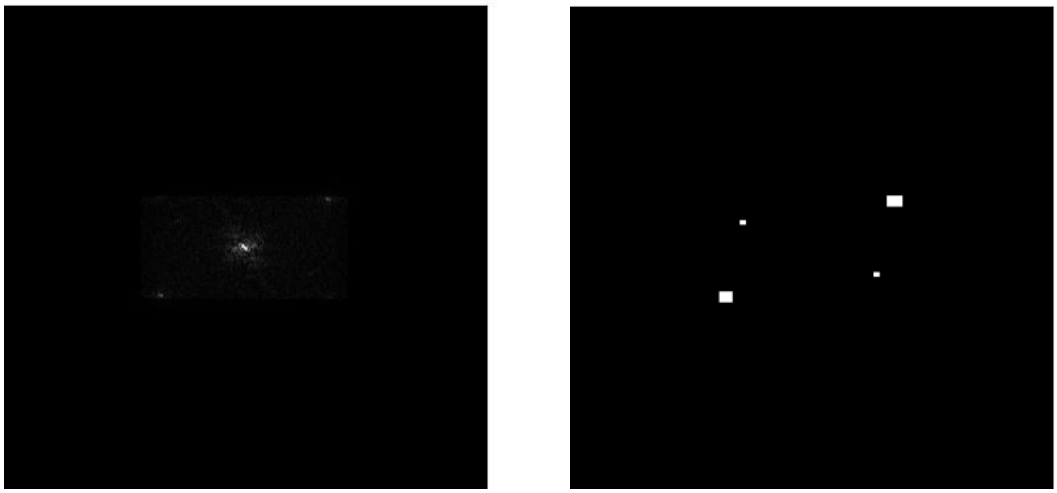


Abbildung 4-7 K-Raum (links) mit dazugehöriger Maske(rechts)

Als Testset werden hier ebenfalls 20% der Daten randomisiert aus dem Datensatz zurückgehalten um als Testdatensatz verwendet werden zu können.

In der Bildtransformation sollen die segmentierten Spikes durch einen passenden Algorithmus mit Daten substituiert werden. Um den Transformationsalgorithmus zu trainieren, werden K-Räume mit Spikes und identische K-Räume ohne Spikes benötigt. Da innerhalb des Datensatzes keine Aufnahmen von identischen K-Räumen mit und ohne Spike existieren, müssen diese künstlich erzeugt werden.

Dazu werden die segmentierten Positionen der Spikes in den betreffenden K-Räumen ermittelt. Die entsprechenden Positionen werden anschließend in K-Räume ohne Spikes projiziert und auf Null gesetzt. Dadurch wird sichergestellt, dass die Position und Größe der künstlich erzeugten Spikes realistisch dargestellt werden. Durch die Projektion der Spikes in die K-Räume entsteht eine Version des K-Raums, in der die betreffenden Positionen genullt sind. Zusätzlich wird eine Version des K-Raums ohne genullte Bereiche beibehalten.

Die K-Räume mit den genullten Daten können anschließend als Eingangsdaten für den Transformationsalgorithmus verwendet werden (siehe Abbildung 4-8 links). Die Version der K-Räume ohne Spikes dient als Zielgröße (siehe Abbildung 4-8 rechts).

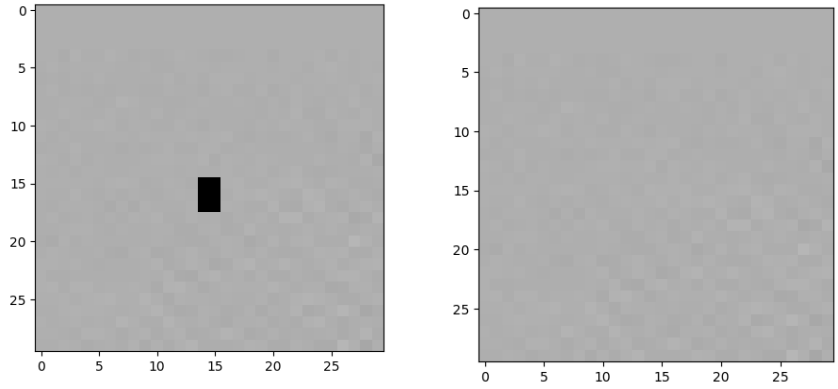


Abbildung 4-8 Links segmentierter Bereich eines Spikes genullt
rechts Grundwahrheit ohne Spike

Da nur ein kleiner Teil des K-Raums mit Daten substituiert werden soll, wird nur ein relevanter Ausschnitt um die Spikes herum als Trainingsdaten verwendet. Als Größe wird hier ein Ausschnitt von 30x30 Pixeln um die betreffende Stelle gewählt. Durch die Reduktion der Ausschnitte kann die benötigte Rechenleistung und Zeit erheblich reduziert werden.

Insgesamt werden 10928 K-Raum Ausschnitte mit dazugehöriger Grundwahrheit erzeugt. Hier werden ebenfalls 20% der Daten randomisiert extrahiert um diese anschließend als Testdatensatz verwenden zu können.

5 *Methoden Analyse*

Um die Spikes innerhalb des K-Raums zu detektieren und zu filtern, wurde bisher stark auf regelbasierte Ansätze gesetzt. Das Ziel dieser Arbeit ist es jedoch, die Effektivität von Deep Learning-Algorithmen bei der Filterung von Spike-Rauschen zu überprüfen. Aus diesem Grund werden im folgenden Kapitel die betrachteten Algorithmen für jeden Teilbereich erläutert. Die Netzwerke werden dabei in die Bereiche Detektion, Segmentierung und Transformation unterteilt. Innerhalb jedes Bereichs wird der Aufbau der Netzwerke beschrieben, die im folgenden Training miteinander verglichen werden.

5.1 Detektion von Spikes

Die Detektion von Spikes erfolgt innerhalb des K-Raums. Als Zielgröße soll der Algorithmus bestimmen, ob in diesem K-Raum Spikes identifiziert, werden können oder nicht. Demnach handelt es sich um einen Algorithmus aus dem Bereich der Bildklassifikation. Die eingehenden Daten werden nach verschiedenen vordefinierten Klassen sortiert. Innerhalb dieser Arbeit werden die Daten nach den Labels "Spike" und "Clean" sortiert, was eine binäre Klassifizierung bedeutet.
Wie bereits in Kapitel 4 "Daten" erörtert, stehen mehrere Datensätze von K-Räumen mit und ohne Spikes zur Verfügung. Da Daten beider Klassen vorhanden sind, kann auf Algorithmen mit einem Supervised Learning Ansatz zurückgegriffen werden. Ein entscheidender Faktor für den Trainingserfolg des Algorithmus ist die sogenannte Netzwerkarchitektur. Diese definiert den Aufbau des Netzwerks und die Signalübertragung innerhalb des Algorithmus. Sie hat direkten Einfluss auf die Performance des Algorithmus in Bezug auf die ihm gestellten Aufgaben.

Jedoch lässt sich aufgrund der Komplexität der Netzwerkzusammensetzungen und der verfügbaren Parameter im Vorfeld nicht sagen, welche Netzwerkarchitektur am besten geeignet ist. Daher muss empirisch ermittelt werden, welche Architektur die beste Performance für die vorgesehene Aufgabe liefert. Innerhalb dieser Arbeit sollen vier verschiedene Algorithmen getestet werden.

1. VGG basiertes Network
2. ResNet basiertes Network
3. Densenet
4. Shufflenet

Die Unterschiede zwischen den folgenden Architekturen soll im Folgenden erläutert werden.

5.1.1 VGG basiertes Netzwerk

Als erstes Netzwerk wird ein Feed Forward Convolutional Network erstellt. Dabei wurde sich an einer VGG Netzwerk Architektur orientiert [Simonyan15] welche 2015 von Simonya und Zisserman im Ramen der ICLR-Konferenz vorgestellt wurde. Das erstellte Netzwerk ist als Ablaufdiagramm in der unteren Tabelle dargestellt.

Tabelle 1 Aufbau VGG Netwerk zur Detektion von Spikes

LAYER	OUTPUT SIZE	KERNEL SIZE	STRIDE	REPEAT
INPUT	288x288			
CONV1	144x144	3x3	2	1
CONVBLOCK1	72x72	3x3	2	1
			1	1
CONVBLOCK2	36x36	3x3	2	1
			1	1
CONVBLOCK3	18x18	3x3	2	1
			1	1
CONVBLOCK4	9x9	3x3	2	1
			1	1
CONVBLOCK5	4x4	3x3	2	1
			1	1
FLATTEN				
FC	16			
CLASSIFICATION	1			

Als Input Layer wird ein Convolutional Layer mit einem Stride von zwei verwendet, gefolgt von Batchnormalization und der ReLU-Aktivierungsfunktion. Innerhalb eines Convolutional Blocks werden jeweils zwei Convolutional Layer eingesetzt, wobei ausschließlich eine Kernelgröße von 3x3 verwendet wird. Durch die Kombination dieser beiden Kernel wird ein effektives rezeptives Feld von 5x5 erreicht. Diese Herangehensweise ermöglicht eine Reduktion der benötigten Parameter und Gewichte im Vergleich zur Verwendung eines einzigen 5x5 Kernels. Zusätzlich ermöglicht die nicht-lineare Aktivierungsfunktion eine komplexere Entscheidungsfunktion des Netzwerks. [Simonyan15, S3]

Nach den Convolutional Layern wird eine Batchnormalization eingefügt, gefolgt von der ReLU-Aktivierungsfunktion. Der erste Convolutional Layer verwendet einen Stride von eins, während der anschließende Layer einen Stride von zwei verwendet, um die resultierende Matrixgröße zu halbieren. Die Anzahl der Filter innerhalb jedes Blocks wird verdoppelt, wobei im ersten Block 8 Filter verwendet werden.

Nach fünf Convolutional Blöcken wird die Bildmatrix durch eine Flatten-Layer in einen Vektor umgewandelt. Darauf folgt eine Fully Connected Schicht mit 16 Neuronen, gefolgt von Batchnormalization und der ReLU-Aktivierungsfunktion. Als Output Layer wird ein Neuron mit der Sigmoid-Aktivierungsfunktion verwendet.

5.1.2 ResNet

Als zweites Netzwerk wird eine Architektur verwendet welche auf der Residual Network (ResNet) Architektur von He et. al. von Microsoft aus dem Jahr 2015 basiert [He15]. Diese verwendet zusätzliche Residual Layer, um das Training von tiefen Neuronalen Netzen zu verbessern. Die verwendete Netzwerkarchitektur ist in Tabelle 2dargestellt.

Tabelle 2 Aufbau ResNet zur Detektion von Spikes

LAYER	OUTPUT SIZE	KERNEL SIZE	STRIDE	REPEAT
INPUT	288x288			
CONV1	144x144	3x3	2	1
MAXPOOLING	72x72	1		
RESBLOCK1	72x72	3x3	2	1
			1	1
RESBLOCK2	36x36	3x3	2	1
			1	1
RESBLOCK3	18x18	3x3	2	1
			1	1
RESBLOCK4	9x9	3x3	2	1
			1	1
FLATTEN				
FC	16			
CLASSIFICATION	1			

Bei dem ResNet handelt es sich ebenfalls um eine Feed Forward Netzwerk. Als Input Layer wird ein Convolutional Layer mit einem Kernel von 3x3 und einem stride von zwei verwendet, gefolgt wird diese Layer durch eine Batchnormalization und eine ReLU-Aktivierung verwendet. Als Hidden Layer werden Residual Blöcke verwendet. Dabei werden jeweils zwei Convolutional Layer mit einem Kernel von 3x3 mit anschließender Batchnormalization und ReLU-Aktivierung verwendet. Die Größe der eingehenden Feature Maps wird dabei durch

einen stride von zwei in der ersten Convolutional Layer jeweils halbiert. Die Anzahl an Filtern wird innerhalb jedes Blocks verdoppelt. Der erste Block realisiert 8 Filter.
Parallel zu den Convolutional Layer werden innerhalb jedes Blocks parallele Residuals implementiert. Diese verbinden die eingehende Matrix eines Blocks mit dem output der zweiten Batchnormalization. Die kombinierte Matrix wird anschließend an die zweite ReLU-Aktivierungsfunktion weitergeleitet. Durch den verwendeten Stride von zwei muss die Matrixgröße innerhalb des Residual Pfades ebenfalls halbiert werden um die Matrixaddition z ermöglichen. Der Aufbau des Residual Blocks zur Realisierung eines strides von zwei ist in Grafik 5-1 dargestellt.

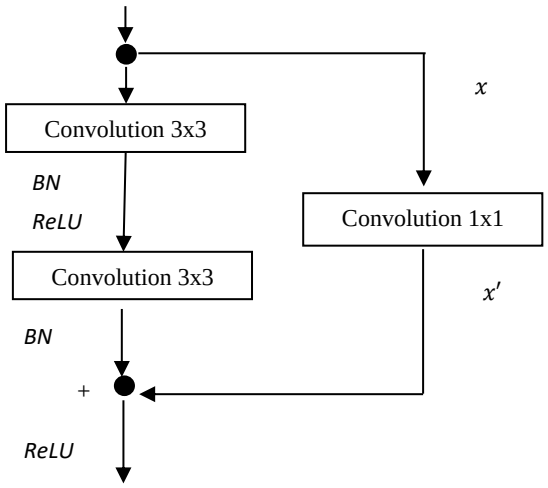


Abbildung 5-1 Aufbau des Residual Pfades mit stride zwei

Dazu wird ein Convolutional Layer mit einen Kernelgröße von 1x1 verwendet und einem stride von zwei. Dadurch wird eine lineare Transformation der gesamten Matrix erreicht bei gleichzeitiger Reduktion der Matrixgröße.

Nachdem die resultierende Matrix durch fünf Residualblöcke gegangen ist, wird sie an einen Flatten-Layer weitergeleitet, der die Matrix in einen Vektor umwandelt. Anschließend folgt eine Fully Connected Schicht mit 16 Neuronen, die mit Batch Normalization und ReLU-Aktivierung aktiviert ist. Für die binären Labels wird ein Neuron mit Sigmoid-Aktivierung verwendet, das die binären Labels repräsentieren kann.

5.1.3 Shufflenet v2

Als drittes Netzwerk wird ein ShuffleNet V2 verwendet, das von Zhang et al. im Jahr 2017 vorgestellt wurde [Xiangyu17]. Dieses Netzwerk wurde speziell für Computer mit geringer Rechenleistung im Bereich von 10 bis 150 MFLOPs entwickelt. Trotz der geringeren Rechenleistung bleibt die Leistungsfähigkeit vergleichbar mit größeren Netzwerken.
Um diese Effizienz zu erreichen, werden zwei neue Operationen verwendet: eine Pointwise-Groupconvolution und ein Channel-Shuffle. Durch die allgemeine Groupconvolution kann die Anzahl der benötigten Operationen reduziert werden. Dazu werden die Eingangskanäle in mehrere Gruppen aufgeteilt und die Faltungen separat auf jeder Gruppe durchgeführt. Dadurch werden nicht alle Convolutional Layer auf allen Kanälen angewendet, was die Anzahl der Operationen verringert. Nach der Groupconvolution werden die Ergebnisse der Gruppen wieder zusammengeführt, um den endgültigen Ausgabewert der Schicht zu generieren. Die Pointwise-Groupconvolution im ShuffleNet verwendet eine Groupconvolution mit einem 1x1-Faltungskern, um die Anzahl der Ausgabekanäle flexibel anzupassen. Die Aufteilung der Kanäle in Gruppen führt jedoch dazu, dass der Informationsfluss zwischen den einzelnen Gruppen nicht mehr gewährleistet ist. Um dieses Problem zu lösen, wird innerhalb der ShuffleNet-Architektur ein Channel-Shuffle-Layer hinzugefügt. Die Funktionsweise wird in Abbildung 5-2 dargestellt.

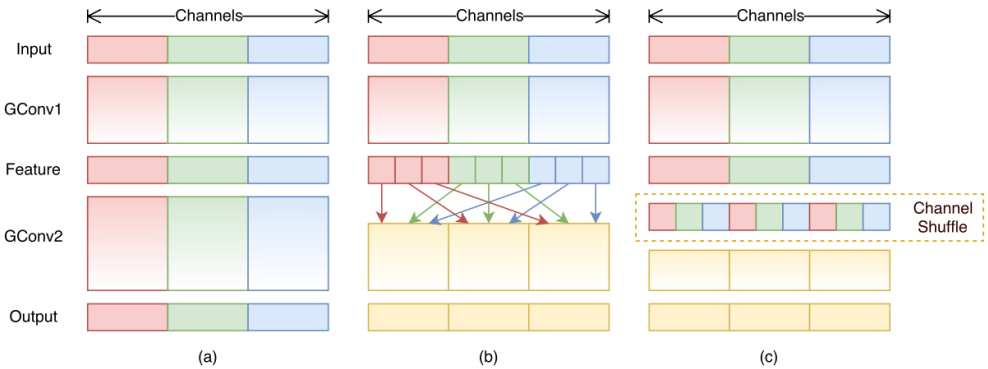


Abbildung 5-2 Funktionsweise der Channel Shuffle Operation [Xiangyu17 S2]

In Abbildung 5-2a sind die Groupconvolutions separat voneinander implementiert. Durch die Organisation der Kanäle in neuen Gruppen (Abbildung 5-2b) wird sichergestellt, dass alle eingehenden und ausgehenden Kanäle vollständig miteinander verbunden sind. Dies wird durch den Einsatz von Channel-Shuffle-Layern realisiert (Abbildung 5-2c). Dabei werden die Kanäle einer Gruppe in mehrere Untergruppen aufgeteilt. Die neu organisierten Gruppen erhalten jeweils Untergruppen aus allen vorherigen Gruppen. Die Implementierung des gesamten Netzwerks ist in der unteren Tabelle dargestellt.

Tabelle 3 Architektur Shufflenet zur Detektion von Spikes

LAYER	OUTPUT SIZE	KERNEL SIZE	STRIDE	REPEAT
INPUT	288x288			
CONV1	144x144	3x3	2	1
MAXPOOL	72x72	3x3	2	
STAGE2	36x36		2	1
	36x36		1	3
STAGE3	18x18		2	1
	18x18		1	7
STAGE4	9x9		2	1
	9x9		1	3
AVERAGE GLOBAL POOL	1x1	9x9		
FC	1000x1000			
CLASSIFICATION	1x1			

Als Eingangsschicht des ShuffleNet wird ein 3x3 Convolutional Layer mit Stride 2 verwendet, gefolgt von einem 3x3 MaxPooling mit Stride 2. Anschließend ist der Netzwerkaufbau in drei Stages organisiert, wobei jede Stage aus mehreren ShuffleNet Units besteht. Eine Unit nutzt dabei den prinzipiellen Aufbau einer Bottleneck Unit als Grundlage. Der prinzipielle Aufbau ist in Abbildung 5-3 dargestellt.

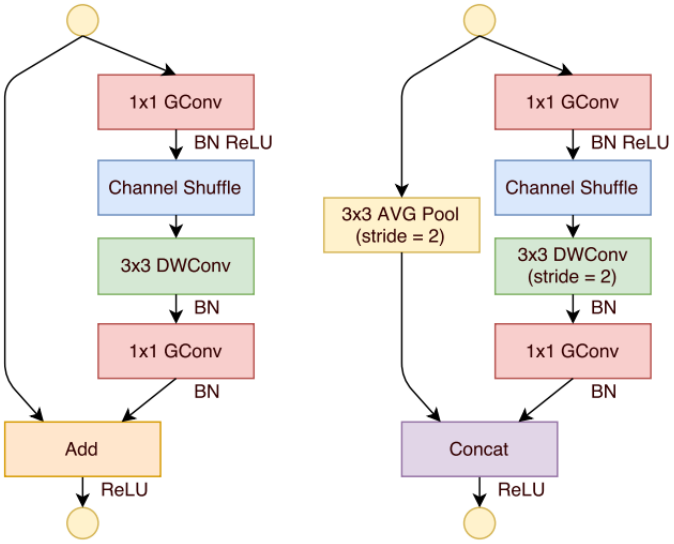


Abbildung 5-3 Shufflenet Unit mit stride eins (links) und stride zwei (rechts)

[Xiangyu17 S3]

Die weiterentwickelten ShuffleNet-Units implementieren eine 1x1-Group-Convolution-Layer, einen Channel-Shuffle-Layer, eine 3x3-Group-Convolution-Layer und eine 1x1-Group-Convolution-Layer. Parallel zu den implementierten Convolution-Layern wird ein Residual-Pfad implementiert, der den Eingang mit dem Ausgang einer Unit verknüpft (Abbildung 5-3 links). Um einen Stride von zwei zu implementieren, wird innerhalb der 3x3-Group-Convolution ein Stride von 2 eingefügt. Um die Matrixgröße des Residual-Pfades anzupassen, wird hier zusätzlich ein 3x3-Average-Pooling mit einem Stride von zwei eingefügt (Abbildung 5-3 rechts). Nach vier Units wird ein Global Pooling implementiert, gefolgt von einer Fully-Connected-Layer mit 1000 Neuronen. Als Classification Layer wird ein Neuron mit Sigmoid-Aktivierung verwendet.

5.1.4 Densenet

Als letzte Architektur wird ein Densenet verwendet. Die Architektur wurde 2018 von Huang et al an der Cornell Universität in Zusammenarbeit mit Facebook AI Research entwickelt [Huang18]. Bei dem Netzwerk handelt es sich um eine Weiterentwicklung der ResNet Architektur. Die verwendete Architektur ist in folgender Tabelle dargestellt.

Tabelle 4 Aufbau Densenet zur Klassifikation von Spikes

LAYER		OUTPUT SIZE	KERNEL SIZE	STRIDE	REPEAT
INPUT		288x288			
CONV		144x144	7x7	2	
MAX POOLING		72x72	3x3	2	
DENSE BLOCK 1		72x72	1x1	2	6
			3x3	1	
TRANSITION LAYER 1	Conv	36x36	1x1	1	
	Average Pooling		2x2	2	
DENSE BLOCK 2		36x36	1x1	2	12
		36x36	3x3	1	
TRANSITION LAYER 2	Conv	18x18	1x1	1	
	Average Pooling		2x2	2	
DENSE BLOCK 3		18x18	1x1	2	24
			3x3	1	
TRANSITION LAYER 3	Conv	9x9	1x1	1	
	Average Pooling		2x2	2	
DENSE BLOCK 4		9x9	1x1	2	16
			3x3	1	
AVERAGE GLOBAL POOL		1x1	9x9		
FC		1000x1000			
CLASSIFICATION		1x1			

Durch die direkte Verbindung der Layer innerhalb einer Architektur lassen sich tiefere Netzwerke effizienter trainieren. Ein Beispiel hierfür ist die ResNet-Architektur, die bei einer erhöhten Anzahl an Layern bessere Ergebnisse als herkömmliche CNNs erzielt. Die DenseNet-Architektur baut auf dieser Idee auf und erweitert sie. Das Netzwerk implementiert Residuals zwischen jedem der implementierten Layer. Jeder Layer erhält somit den Output aller vorangegangenen Layer als Input. Dadurch können spezifische Features einfacher

wiederverwendet und ausgetauscht werden. Innerhalb der Architektur kann so die maximale Vernetzung der Schichten untereinander erreicht werden.

Im Input-Layer ist eine 7x7 Convolutional Layer mit einem Stride von zwei und ein Max Pooling mit einem Kernel von 3x3 und einem Stride von zwei implementiert. Nach dem Input sind vier Dense Blöcke implementiert. Innerhalb eines Blocks sind jeweils 1x1 und 3x3 Kernel realisiert. Der erste Block verwendet 6 Filter, der zweite Block 12, der dritte 24 und der letzte Block 16 Filter. Zwischen den Dense Blöcken sind Transition Layer eingefügt. Die Transition Layer beinhalten jeweils eine 1x1 Convolutional Layer gefolgt von einem 2x2 Average Pooling Layer mit einem Stride von zwei. Als ausgebende Schicht wird ein 7x7 Average Pooling implementiert, gefolgt von einer Fully Connected Layer mit 1000 Neuronen und Softmax-Aktivierung. Das Netzwerk wird für die binäre Klassifikation noch erweitert um eine Schicht mit einem Neuron und Sigmoid-Aktivierung. Die Residuals sind jeweils in einer feet forward Struktur aufgebaut. Die Verbindungen werden in Abbildung 5-4 dargestellt.

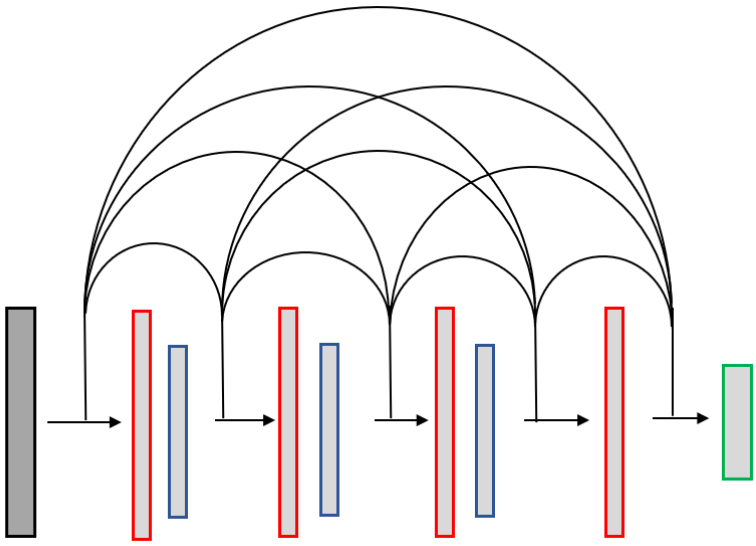


Abbildung 5-4 Residual Verbindungen innerhalb von Densenet

Das bedeutet das jede Layer den Output an jede darauffolgende Layer weitergibt. Zusätzlich wird anders als bei ResNet die Informationen der Residuals nicht addiert, sondern durch Concatenation zusammengefügt.

5.2 Segmentierung von Spikes

In der Segmentierung werden den einzelnen Pixeln des K-Raums Labels zugewiesen. Innerhalb dieser Arbeit soll unterschieden werden zwischen Pixeln, die Teil von Spike-Rauschen sind, und Pixeln, die den regulären K-Raum darstellen. Demzufolge wird eine binäre Segmentierung durchgeführt. Im Bereich der KI spricht man dabei von semantischer Bildsegmentierung. Als Eingangsgröße wird der gesamte K-Raum verwendet. Die Zielgröße stellt eine Maske dar, die die Pixel identifizierbar macht, welche Spikes enthalten. Die Grundwahrheit enthält inhärent beide Klassen, daher kann hier ein supervisierter Lernansatz genutzt werden.

Auch bei der Segmentierung kann die verwendete Netzwerkarchitektur einen entscheidenden Unterschied für den Trainingserfolg darstellen. Aus diesem Grund werden experimentell verschiedene Netzwerkarchitekturen miteinander verglichen. Innerhalb dieser Arbeit werden folgende vier Architekturen verwendet.

1. ResNet
2. UNet
3. UNet mit Residuals
4. UNetPP

5.2.1 ResNet zur Segmentierung

Als erste Architektur wird ein ResNet verwendet. Dabei handelt es sich um ein Feed Forward Netzwerk wie es bereits in seiner Grundfunktion in Kapitel 5.1.2 ResNet Detektion erläutert wurde. Innerhalb der Segmentierung wird ein Fully Convolutional Network verwendet. Dabei handelt es sich um ein Netzwerk, welches ausschließlich aus Convolutional Layern besteht. Das verwendete Netzwerk ist in der unteren Tabelle visualisiert.

Tabelle 5 Aufbau ResNet zur Segmentierung von Spikes

LAYER	OUTPUT SIZE	KERNEL SIZE	STRIDE	REPEAT
INPUT	288x288			
CONV1	288x288	3x3	1	1
RESBLOCK1	288x288	3x3	1	2
RESBLOCK2	288x288	3x3	1	2
RESBLOCK3	288x288	3x3	1	2
RESBLOCK4	288x288	3x3	1	2
RESBLOCK5	288x288	3x3	1	2
CLASSIFICATION	288x288	1x1	1	

Innerhalb des ResNets wird auf eine Reduzierung der eingehenden Bildgröße verzichtet. Die ausgehende Feature Map benötigt innerhalb der Segmentierung die gleiche Größe wie die eingehende Bildmatrix. Die Input Layer ist implementiert als eine Convolutional Layer mit anschließender Batch Normalization und ReLU-Aktivierung. Innerhalb der Convolution wird dabei ein Kernel von 3x3 verwendet, mit acht Feature Maps.
Innerhalb jedes ResBlocks werden zwei Convolutional Layer mit jeweils einem Kernel von 3x3 verwendet. Anschließend folgt eine Batch Normalization, und als Aktivierungsfunktion wird ReLU verwendet. Innerhalb des ersten Blocks werden 16 Feature Maps verwendet. Die Anzahl dieser Feature Maps verdoppelt sich dabei innerhalb jedes Blocks.

Parallel zu den Convolutional Layern werden zusätzliche Residuals verwendet, die innerhalb jedes Convolutional Blocks parallel implementiert werden. Dabei werden die eingehenden Matrizen eines Blocks mit dem Output der letzten Batch Normalization Layer zusammengeführt. Das Ergebnis wird dann an die letzte Aktivierungsfunktion weitergeleitet.

Als Output wird ein Convolutional Layer mit einem Kernel von 1x1 verwendet und eine Sigmoid Aktivierung, um einen Wertebereich zwischen null und eins innerhalb der Pixel darzustellen.

5.2.2 UNet

Als zweites Netzwerk soll ein UNet verwendet werden. Dieses wurde 2015 von Ronneberger et. al. vorgestellt [Ronnenberger15]. Dabei handelt es sich um ein Netzwerk mit Encoder und Decoder Struktur. Die realisierte Architektur ist in der unteren Grafik dargestellt.

Tabelle 6 Aufbau UNet zur Segmentierung von Spikes

	OUTPUT SIZE	**KERNEL SIZE**	**STRIDE**	**REPEAT**
INPUT	288x288			
CONVBLOCK1	144x144	3x3	1	2
MAXPOOLING1		2x2	2	1
CONVBLOCK2	72x72	3x3	1	2
MAXPOOLING2		2x2	2	1
CONVBLOCK3	36x36	3x3	1	2
MAXPOOLING3		2x2	2	1
CONVBLOCK4	18x18	3x3	1	2
MAXPOOLING4		2x2	2	1
CONVBLOCK5	9x9	3x3	1	2
MAXPOOLING5		2x2	2	1
UPSAMPLE1	18x18	2x2	2	1
CONVBLOCK6		3x3	1	2
UPSAMPLE2	36x36	2x2	2	1
CONVBLOCK7		3x3	1	2
UPSAMPLE3	72x72	2x2	2	1
CONVBLOCK8		3x3	1	2
UPSAMPLE4	144x144	2x2	2	1
CONVBLOCK9		3x3	1	2
UPSAMPLE5	288x288	2x2	2	1
CONVBLOCK10		3x3	1	2
CLASSIFICATION	288x288	1x1	1	

Innerhalb jedes Blocks werden zwei Convolutional Layer realisiert. Als Kernelgröße wird innerhalb jeder Convolution 3x3 verwendet. Nach den Convolutional Layern wird eine Batch Normalization und eine ReLU-Aktivierung verwendet. Die Anzahl der verwendeten Filter erhöht sich jeweils um den Faktor zwei im Vergleich zum vorangegangenen Block. Die Größe der Bildmatrix wird mittels MaxPooling reduziert. Dabei wird eine Kernelgröße von 2x2 und ein Stride von zwei verwendet. Im Encoder-Teil des Netzwerks werden die eingehenden Featuremaps durch die Verwendung von Upsampling-Layern jeweils um den Faktor zwei in der Größe erhöht. Durch den symmetrischen Aufbau des Netzes sind die Größen der eingehenden und ausgehenden Featuremaps identisch. Hierfür wird ein Upsampling-Layer mit bilinearer Interpolation verwendet. Anschließend sind zwei Convolutional Layer mit einem Kernel von 3x3 realisiert. Nach jeder Convolutional Layer wird eine Batch Normalization und eine ReLU-Aktivierungsfunktion verwendet.

Als Output des Netzwerks wird eine Convolutional Layer mit einem Kernel von 1x1 verwendet, und eine Sigmoid-Aktivierung wird angewendet, um einen Pixelbereich zwischen 0 und 1 darzustellen. Zwischen den Blöcken im Encoder- und Decoder-Teil, die die gleiche Featuremap-Größe haben, werden Residuals verwendet. Der Aufbau der Residual-Verbindungen wird in Abbildung 5-5 dargestellt.

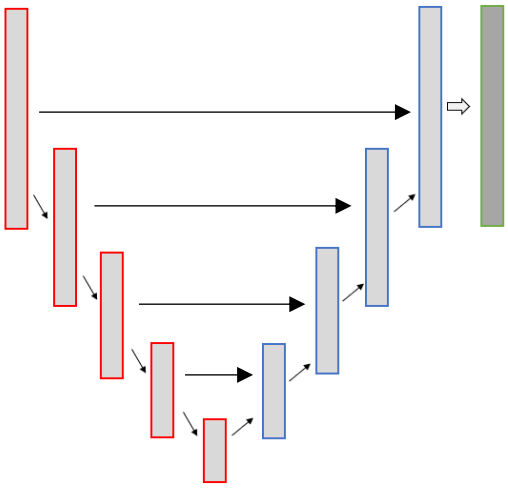


Abbildung 5-5 Verwendete UNet Architektur zur Spike Segmentierung.

Die Residuals verbinden jeweils den Output der Encoder Blöcke mit den Input der Decoder Blöcke (Abbildung 5-5).

5.2.3 UNet mit Residuals

Innerhalb der UNet Architekturen werden die Convolutional Layer ohne zusätzliche Residuals verwendet. Innerhalb dieser Architektur soll die Architektur durch diese erweitert werden. Als Basisarchitektur wird das in 5.2.2 beschriebene UNet verwendet. Die Architektur wird sich in folgender Tabelle dargestellt.

Tabelle 7 Aufbau der UNet Architektur mit zusätzlichen Residuals

	OUTPUT SIZE	**KERNEL SIZE**	**STRIDE**	**REPEAT**
INPUT	288x288			
RESBLOCK1	144x144	3x3	1	2
MAXPOOLING1		2x2	2	1
RESBLOCK2	72x72	3x3	1	2
MAXPOOLING2		2x2	2	1
RESBLOCK3	36x36	3x3	1	2
MAXPOOLING3		2x2	2	1
RESBLOCK4	18x18	3x3	1	2
MAXPOOLING4		2x2	2	1
RESBLOCK5	9x9	3x3	1	2
MAXPOOLING5		2x2	2	1
UPSAMPLE1	18x18	2x2	2	1
RESBLOCK6		3x3	1	2
UPSAMPLE2	36x36	2x2	2	1
RESBLOCK7		3x3	1	2
UPSAMPLE3	72x72	2x2	2	1
RESBLOCK8		3x3	1	2
UPSAMPLE4	144x144	2x2	2	1
RESBLOCK9		3x3	1	2
UPSAMPLE5	288x288	2x2	2	1
RESBLOCK10		3x3	1	2
CLASSIFICATION	288x288	1x1	1	

Die verwendete Netzwerkarchitektur wurde in seiner Konfiguration dem UNet aus Kapitel 5.2.2 UNet entnommen. Anders als bei dem bereits implementierten Netzwerk wird hier innerhalb der Encoder und Decoder Struktur keine Convolutional Blöcke, sondern Residual Blöcke verwendet. Innerhalb jedes Blocks sind dabei zusätzlichen Residuals implementiert, wie sie bereits innerhalb der Technischen Grundlagen in Kapitel 2 definiert wurden.

5.2.4 UNetPP

Innerhalb von UNet gibt es zwei Limitierungen, die bei der Verwendung der Architektur entstehen. Zunächst kann die optimale tiefe des UNet nicht von Beginn an festgelegt werden, da diese nach Komplexität der gestellten Aufgabe variiert.
Zusätzlich ist die ausschließliche Verbindung von Feature Maps mit identischer Größe limitierend. Bei der Realisierten UNetPP Architektur werden die genannten Nachteile von UNet durch zusätzliche Verbindungen und Convolutional Layer umgangen Das Netzwerk wurde 2020 von Zhou et. al. eingeführt [Zhou20]. Es handelt sich also um eine Erweiterung des in Kapitel 5.2.2 realisierten UNet. Das Netzwerk wird innerhalb der unteren Tabelle 8 dargestellt.

Tabelle 8 Aufbau der UNetPP Architektur

	OUTPUT SIZE	KERNEL SIZE	STRIDE	REPEAT
INPUT	288x288			
CONV0_0	288x288	3x3	1	2
CONV1_0	144x144	3x3	1	2
MAXPOOLING		2x2	2	1
CONV0_1	288x288	3x3	1	2
UPSAMPLING			2	1
CONV2_0	72x72	3x3	1	2
MAXPOOLING		2x2	2	1
CONV1_1	144x144	3x3	1	2
UPSAMPLING			2	1
CONV0_2	288x188	3x3	1	2
UPSAMPLING			2	1
CONV3_0	36x36	3x3	1	2
MAXPOOLING		2x2	2	1
CONV2_1	72x72	3x3	1	2
UPSAMPLING			2	1
CONV1_2	144x144	3x3	1	2
UPSAMPLING			2	1
CONV0_3	288x288	3x3	1	2
UPSAMPLING			2	1
CONV4_0	18x18	3x3	1	2
MAXPOOLING		2x2	2	1
CONV3_1	36x36	3x3	1	2
UPSAMPLING				1
CONV2_2	72x72	3x3	1	2
UPSAMPLING				1
CONV1_3	144x144	3x3	1	2
UPSAMPLING				1
CONV0_4	288x288	3x3	1	2
UPSAMPLING			2	1
CLASSIFICATION	288x288	1x1	1	

Bei der UNetPP Architektur handelt es sich um ein Fully Convolutional Network. Diese realisiert identisch zu UNet eine Encoder und Decoder Architektur. Der Aufbau der Architektur ist in Abbildung 5-6 dargestellt.

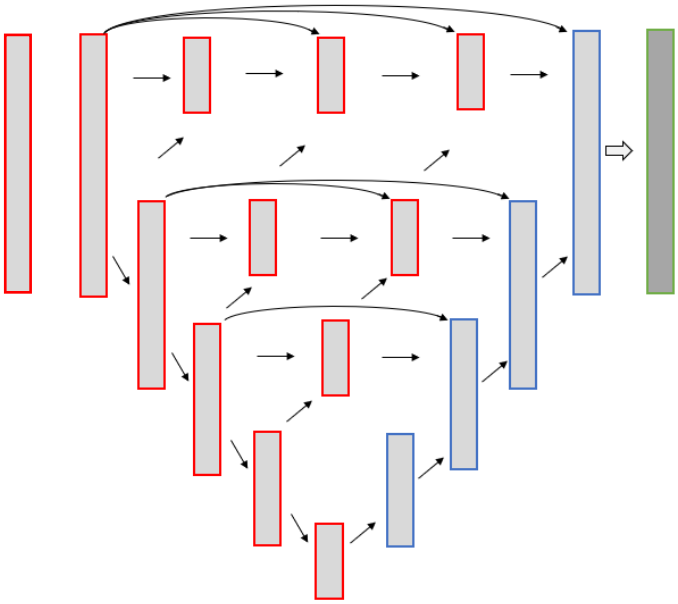


Abbildung 5-6 Verwendete UNetPP Architektur zur Segmentierung von Spikes

Innerhalb des äußeren Pfads im Encoder-Teil des Netzwerks kommt es pro Layer zu einer Reduzierung der Bildgröße um den Faktor zwei. Dabei werden innerhalb jedes Blocks zwei Convolutional Layer mit anschließender Batch Normalization und ReLU-Aktivierungsfunktion verwendet. Die Convolutional Layer werden mit einem Kernel von 3x3 realisiert. Die Anzahl der verwendeten Filter verdoppelt sich bei jeder Reduzierung der Bildgröße.
Zur Reduzierung wird in jeder Schicht ein MaxPooling Layer mit einem Kernel von 2x2 und einem Stride von zwei verwendet. Die Layer mit Downsampling sind innerhalb der Grafik durch einen Pfeil gekennzeichnet, der von links oben nach rechts unten verläuft.
Um die ursprüngliche Featuremap-Größe wiederherzustellen, wird im Decoder-Teil ein Upsampling verwendet. Innerhalb jedes Upsampling-Layers wird die Bildgröße durch bilineare Interpolation verdoppelt. Die Upsampling-Layer sind innerhalb der Grafik durch Pfeile von links unten nach rechts oben gekennzeichnet.

Innerhalb des äußeren Pfads sind identisch zum UNet Convolutional Layer implementiert. Bei gleicher Featuremap-Größe werden die Convolutional Layer des Encoders und des Decoders zusätzlich verbunden. Innerhalb der UNetPP Architektur werden zusätzliche Convolutional Layer zwischen den äußeren Pfaden des Netzwerks eingefügt. Diese verwenden den Input der Layer mit gleicher Featuremap-Größe und zusätzlich die darunterliegenden Featuremaps mit

halbierter Größe. Durch diese zusätzlichen Convolutional Layer sind alle Featuremaps miteinander verbunden. Dadurch ergeben sich innerhalb des gesamten Netzwerks variable UNet-Größen, die sich autonom an die Komplexität der Aufgabe anpassen können. Zusätzlich wird jede Featuremap als Input für die darüberliegende Schicht genutzt, was zu einer variablen Merkmalsfusion führt und das Trainingsergebnis verbessern kann.

5.3 Transformierung von K-Raum Ausschnitte

Ziel der Transformation ist es, die segmentierten Bereiche innerhalb des K-Raums durch passende Daten zu substituieren. Die Zielgröße ist eine transformierte Form des eingehenden K-Raums. Daher handelt es sich um eine generative Modellierung durch einen Deep Learning Algorithmus.

Um den Algorithmus zu trainieren, werden die in Kapitel 4 beschriebenen Datensätze verwendet. Da hier eine eindeutig zugewiesene Grundwahrheit für jede Eingangsgröße vorhanden ist, handelt es sich um einen Supervised Learning Algorithmus. Es werden zwei Architekturen verglichen: ein einfaches VGG-basiertes Netzwerk und eine vergleichbare Architektur mit zusätzlichen Residuals.

5.3.1 VGG

Zunächst soll ein einfaches VGG basiertes Netzwerk verwendet werden, um die fehlenden Daten der segmentierten Bereiche zu transformieren. Dabei soll ein Fully Convolutional Network verwendet werden. Die Architektur ist in der unteren Abbildung dargestellt.

Tabelle 9 Aufbau VGG zur Transformation von Segmentierten Spikes

LAYER	OUTPUT SIZE	KERNEL SIZE	STRIDE	REPEAT
INPUT	30x30			
CONVOLUTION	30x30	3x3	1	1
CONVBLOCK1	30x30	5x5	1	2
CLASSIFICATION	30x30	1x1	1	

Die Eingangsschicht implementiert ein Convolutional Layer mit einem Kernel der Größe 3x3 und einem Stride von eins. Auf den ersten Layer folgen eine Batch Normalization und eine ReLU-Aktivierungsfunktion. Als Hidden Layer werden innerhalb eines Blocks jeweils zwei Convolutional Layer realisiert. Der erste Layer verwendet einen Kernel der Größe 5x5, während der zweite Layer einen Kernel der Größe 3x3 implementiert. Nach jeder Convolution folgen eine Batch Normalization und eine ReLU-Aktivierungsfunktion. Als Output wird ein

Convolutional Layer mit einer Kernelgröße von 1 verwendet und eine Sigmoid-Aktivierung. Dadurch kann der Wertebereich zwischen 0 und 1 dargestellt werden.

5.3.2 ResNet

Als zweites Netzwerk wird ein Fully Convolutional Network mit zusätzlichen Residuals verwendet. Im unteren Verlaufsdiagramm wird das ResNet zur Transformation der KRaum ausschnitte dargestellt.

Tabelle 10 Aufbau ResNet zur Transformation von Segmentierten Spikes

LAYER	OUTPUT SIZE	KERNEL SIZE	STRIDE	REPEAT
INPUT	30x30			
CONVOLUTION	30x30	3x3	1	1
RESBLOCK1	30x30	5x5	1	2
CLASSIFICATION	30x30	1x1	1	

Die Aufbau ist dabei identisch zu der in Abschnitt 5.3.1 realisierten Architektur. Zusätzlich werden innerhalb dieser Architektur Residuals verwendet, die in den beiden mittleren Hidden Layern implementiert sind. Die Residuals verbinden den Input der Blöcke mit dem Output der letzten Batch Normalization. Die L1-Norm beider Matrizen wird anschließend an die letzte Aktivierungsfunktion des Blocks weitergeleitet.

6 Experimente

Um einen Benchmark für die verschiedenen implementierten Netzwerke in den drei Teilbereichen Detektion, Segmentierung und Transformation aufzustellen, werden sie im Rahmen des Trainings und anschließenden Tests miteinander verglichen. Im folgenden Abschnitt wird genauer beschrieben, wie das Training der Netzwerke aufgebaut war. Außerdem werden die verwendeten Metriken zur Evaluierung der Ergebnisse beschrieben.

6.1 Aufbau Training Spike Detektion

Die implementierten Netzwerke (ResNet, VGG, Shufflenet und Densenet) werden mit dem in Kapitel 4 beschriebenen Datensatz trainiert. Zu Beginn wurden bereits 20% der Daten als Testdatensatz extrahiert. Aus den verbleibenden Trainingsdaten werden 20% zufällig als Validierungsdatensatz ausgewählt. Um sicherzustellen, dass die Trainings- und Validierungsdatensätze in jedem Training identisch sind, wird ein einheitlicher Seed für die Zufallsauswahl verwendet.

Wie bereits in Kapitel 4 erwähnt, handelt es sich um einen unbalancierten Datensatz. Das Verhältnis zwischen den Daten mit Spike und denen ohne Spike beträgt etwa 1:8. Diese Ungleichgewichtung muss beim Training berücksichtigt werden. Dazu werden zusätzliche Faktoren (Klassenwichtungen) eingeführt, die während der Backpropagation abhängig von der Klasse multipliziert werden. Durch die Gewichtung der Kostenfunktion nach Klasse kann die unterrepräsentierte Klasse besser erkannt werden. Die Faktoren werden mittels folgender Funktionen invers proportional zur Verteilung ermittelt.

(6-1)

$$\alpha_{mit\ spike} = \frac{nsamples_{\ mit\ spike}}{nsample_{gesammt}}$$

(6-2)

$$\alpha_{ohne\ spike} = \frac{nsamples_{\ ohne\ spike}}{nsample_{gesammt}}$$

Wobei α die Vorfaktoren darstellen, die mit den Klassen multipliziert werden. Für Daten mit Spike wird ein Faktor von 8,74 verwendet, und für Daten ohne Spike wird ein Faktor von 1,25 verwendet. Dadurch können so zunächst der numerische Unterschied in der Klassenhäufigkeit ausgeglichen werden. Zur Detektion der Spikes ist jedoch eine erhöhte Sensetivität in Beug auf die Detetion von Spikes ausschlaggebend. Aus diesem Grund wird zusätzlich ein Faktor von 1,6 zur bestehenden Gewichtung der Spikes hinzugefügt. Dadurch soll sichergestellt werden dass die Klassen mit dem Label Spike sicherer erkannt werden kann.

Da es sich um eine binäre Klassifikation handelt, wird als Verlustfunktion die Binäre Kreuzentropie verwendet. Jedoch liefert diese Kostenfunktion kein lesbares Feedback zur korrekten Klassifikation.Aus diesem Grund wird eine zusätzliche Metrik verwendet, um die Leistung des Netzwerks zu bewerten. Aufgrund der ungleichen Verteilung der Daten sind die klassischen Metriken, die von einer gleichmäßigen Verteilung der Labels ausgehen, nicht geeignet. Basierend auf den Anforderungen wird der F1-Score als Metrik ausgewählt. Dieser Score ist ein maßgebliches Kriterium zur Beurteilung der Netzwerke. Er wird aus der Präzision (Precision) und dem Rückruf (Recall) der Ergebnisse berechnet. Die Präzision wird durch folgende Formel ermittelt.

(6-3)

$$Prec = \frac{True\ Positives}{True\ Positive + \ False\ Positives}$$

Dadurch werden alle Samples ermittelt welche unter den Positiv klassifizierten Werten auch ein positives Label besitzen. Eine hohe Precision ist ein Indikator dafür, dass positiv klassifizierte Instanzen tatsächlich ein positives Label besitzen. Als zweiter Teil wird der Recall berechnet dieser wird über folgende Formel ermittelt.

(6-4)

$$Rec = \frac{True\ Positives}{True\ Positive + \ False\ Negative}$$

Hier werden alle Samples gewertet, die als positiv erkannt wurden und tatsächlich positiv sind. Der Recall kann als Maß für die Anzahl der korrekt gefundenen positiven Instanzen verstanden werden. Ein hoher Recall deutet darauf hin, dass alle vorhandenen positiven Instanzen gefunden wurden. Der F1-Score setzt sich aus den beiden oben beschriebenen Metriken, Präzision und Recall, gemäß folgender Formel zusammen.

(6-5)

$$F1 = 2 * \frac{Prec * Rec}{Prec + Rec}$$

Der Score kann Werte zwischen null und eins annehmen, wobei eins ein perfektes Ergebnis darstellt. Durch die Kombination der beiden Metriken Präzision und Recall lässt sich beurteilen, wie gut das Netzwerk positive Labels korrekt klassifizieren kann und wie gut es die tatsächlich positiven Klassifizierungen erkennt. In der angestrebten Klassifizierung ist sowohl der Recall als auch die Präzision wichtig, daher kann die symmetrische Grundkonfiguration der Metrik beibehalten werden.

Die optimalen Metaparameter können für jedes Netzwerk unterschiedlich sein. Daher werden sie innerhalb des Experiments durch eine Metaparameteroptimierung ermittelt und festgelegt. Dafür wird das Optuna Framework von Akiba et. al. verwendet [Akiba19]. Der Algorithmus verwendet dabei einen Bayersian Optimierungsalgorithmus zur Optimierung der Metaparameter. Jedes Netzwerk wird in 5 Durchgängen über 150 Epochen trainiert. Während

jedes Durchgangs werden die Metaparameter zufällig aus einer Liste verfügbarer Kombinationen ausgewählt. Der Rahmen, innerhalb dem die Metaparameter variieren können, ist in Tabelle 11 unten dargestellt.

Tabelle 11 Mögliche Metaparameterkombinationen innerhalb des Trainings der Netzwerke zur Klassifikation von Spikes in K-Raum Daten

Metaparameter	**Werte**
Optimizer	Adam, NAdam
Learningrate	1e-3, 5e-4, 3e-4, 1e-5
Batchsize	16, 32

Die Metaparameter Learning Rate und Batch Size können Werte aus den numerischen Bereichen in Tabelle 1 annehmen. Zusätzlich kann der Optimizer zwischen Adam und NAdam variiert werden. Die Eingabeformate der Daten für das Netzwerk werden nach jeder Metaparameteroptimierung verändert. Dabei wird zwischen absolutem Datenformat, L1-Norm, Realteil und Imaginärteil variiert. Die Datentypen wurden in Kapitel 4 genauer beschrieben.
Um Overfitting des Netzwerks zu reduzieren, wird eine Bildaugmentationspipeline eingerichtet. Diese manipuliert die eingegebenen Matrizen während des Trainings automatisch und zufällig. Die Pipeline wird in der folgenden Tabelle mit den definierten Wahrscheinlichkeiten und Parametern dargestellt.

Tabelle 12 Image Augmentation Operationen für die Klassifikation von Spikes in K-Raum Daten

Augmentation Operation	**Werte**
Horizontal Spiegelung	Wahrscheinlichkeit = 0,5
Vertikal Spiegelung	Wahrscheinlichkeit = 0,5
Gaus bluring	Kernel = 3,5 Sigma = 0.3-1.5

Der eingehende K-Raum wird zunächst mit einer Wahrscheinlichkeit von jeweils 50% über beide Achsen gespiegelt. Anschließend wird ein Gauß-Kernel eingesetzt um die eingehenden Matrizen zu glätten. Dabei wird randomisiert zwischen einer kernel Größe zwischen 3 und 5

sowie ein Sigma zwischen 0,3 und 1,5 ausgewählt. Dadurch wird die Intensität des bluring variabel definiert.

6.2 Aufbau Experiment Spike Segmentierung

Innerhalb der Segmentierung sollen die zuvor maskierten Bereiche der Spikes durch ein neuronales Netzwerk extrahiert werden. Standardmäßig wird zur Optimierung von Segmentierungsalgorithmen der Mean Squared Error (MSE) als Kostenfunktion verwendet. Der MSE misst die pixelweise Übereinstimmung der Grundwahrheit mit dem Output des segmentierenden Netzwerks. Bei der Segmentierung von Spikes ist der Bereich, der erkannt werden soll, in der Regel erheblich kleiner als der umgebende K-Raum. Dadurch ergibt sich ein Optimierungsproblem bei der Verwendung des MSE als Kostenfunktion. Die geringe Größe des zu segmentierenden Bereichs führt dazu, dass der MSE bereits minimiert werden kann, ohne dass Spikes segmentiert werden. [VanBeers19 S1]

Um dieses Problem zu adressieren, wird der Intersection over Union (IoU) Loss zur Optimierung der Netzwerke verwendet. Der Loss wird durch die folgende Funktion definiert.

(6-6)

$$IOU = \frac{A \cap B}{A \cup B}$$

Der Aufbau der Intersection over Union wird in folgender Abbildung 6-1 dargestellt.

$$IOU = \frac{Überscheidung}{Vereinigung}$$

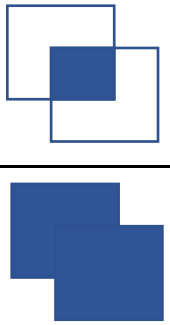


Abbildung 6-1 Intersection over Union

Der Zähler der Funktion repräsentiert den Bereich der Überschneidung zwischen der Grundwahrheit A und dem segmentierten Bereich B. Der Nenner hingegen umfasst den gesamten Bereich der Grundwahrheit und den vom Netzwerk segmentierten Bereich. Das Ergebnis der IoU-Funktion kann als prozentualer Anteil der korrekt erkannten segmentierten Bereiche interpretiert werden. Die ursprüngliche IOU-Verlustfunktion ist nicht differenzierbar und kann daher nicht direkt als Kostenfunktion zur Optimierung der Netzwerke verwendet werden. Um dieses Problem zu lösen, wird die IOU-Funktion in eine differenzierbare Form umgewandelt, die Wahrscheinlichkeiten repräsentiert. Die modifizierte Formel wird in Abschnitt 6-7 beschrieben.

(6-7)

$$IOU' = \frac{|A * B|}{|A + B - (A * B)|}$$

In der modifizierten Formel bleiben A und B unverändert, aber $A * B$ ist das elementweise Produkt von A und B. Im Zähler wird eine Approximation des Schnitts erstellt, indem die Wahrscheinlichkeit von B gegeben wird, wenn A 1 ist, andernfalls wird 0 gegeben. Dadurch ist der Schnitt am größten, wenn B 1 ist, wo auch immer A 1 ist, genau wie erwartet. Der Nenner ist eine Addition von A und B mit einer Abzug des Schnitts, genau wie in einer regulären Berechnung für die Vereinigung, um den Effekt des doppelten Zählens des Schnittbereichs zu verringern. [VanBeers19 S4-5]

Die optimalen Metaparameter für jedes Netzwerk werden während des Trainings in einer Metaparameteroptimierung festgelegt. Dabei wird jedes Netzwerk über 120 Epochen mit einer randomisierten Kombination von Metaparametern trainiert. Das Training wird viermal wiederholt, wobei jedes Mal unterschiedliche Parameter verwendet werden. Die möglichen Variationen der Parametereinstellungen sind in der unten stehenden Tabelle dargestellt.

Tabelle 13 Mögliche Metaparameterkombinationen zur Segmentierung von Spikes in K-Raum Daten.

Metaparameter	**Werte**
Optimizer	Adam, NAdam
Learningrate	1e-3, 5e-4, 3e-4, 1e-5
Batchsize	16, 32

Die Metaparameter Learningrate und Batchsize können zwischen den in Tabelle 13 definierten numerischen Werten variieren. Zusätzlich kann der Optimizer zwischen Adam und NAdam variiert werden.

Um eine mögliche Überanpassung an die begrenzten Spike-Daten im K-Raum zu reduzieren, wird eine Image Augmentation Pipeline eingesetzt, die die Daten randomisiert und automatisch anpasst. In der folgenden Tabelle werden die Transformationen innerhalb der Pipeline in der Reihenfolge ihres Ablaufs dargestellt.

Tabelle 14 Image Augmentation Pipeline für die Segmentierug der Spikes

Augmentation Operation	**Werte**
Horizontal Spiegelung	Wahrscheinlichkeit = 0,5
Vertikale Spiegelung	Wahrscheinlichkeit = 0,5
Gaus bluring	Kernel = 3,5, Sigma = 0.3-1.5

Die K-Räume werden automatisiert mit einer Wahrscheinlichkeit von 0,5 jeweils um eine der Achsen gespiegelt. Anschließend wird ein gaussian bluring angewendet. Dabei wird randomisiert zwischen einer kernel Größe zwischen 3 und 5 sowie ein Sigma zwischen 0,3 und 1,5 ausgewählt.

6.3 Aufbau Experiment Spike Transformation

Innerhalb der Transformation sollen die segmentierten Bereiche durch passende Daten substituiert werden. Da es sich um ein Netzwerk handelt, dessen Zielgröße und Grundwahrheit Matrizen sind, kann der Mean Squared Error (MSE) als Kostenfunktion verwendet werden. Beim MSE wird die quadrierte Differenz aller Pixel summiert und als Kosten verwendet. Als Input müssen sowohl der Real- als auch der Imaginärteil verwendet werden, um eine anschließende Transformation der Daten in den Bildraum durch die inverse Fourier-Transformation zu ermöglichen.
Um die optimalen Metaparameter zu finden, werden diese während des Trainings optimiert. Dafür wird das Training jedes Netzwerks fünf Mal wiederholt. In jedem Durchlauf wird das Netzwerk über 250 Epochen trainiert. Die Variationen der Metaparameter sind in der folgenden Tabelle dargestellt.

Tabelle 15 Mögliche Metaparameterkombinationen zur Transformierung von Spikes in K-Raum Daten.

METAPARAMETER	WERTE
OPTIMIZER	Adam, NAdam
LEARNINGRATE	1e-3, 5e-4, 3e-4, 1e-5
BATCHSIZE	256, 512, 1024

Aufgrund der geringen Größe der eingehenden Matrizen können während der Transformation große Batchsizes zwischen 256 und 1024 verwendet werden. Als Optimizer wird entweder Adam oder NAdam verwendet. Die Learningrate variiert zwischen 1e-3 und 1e-5. Zusätzlich wird eine variable Learningrate eingesetzt. Diese halbiert sich, wenn nach 10 Epochen keine Verbesserung des Loss erreicht wurde. Die Learningrate kann dabei minimal einen Wert von 1e-6 erreichen.

7 Ergebnisse und Diskussion

Im folgenden Kapitel werden die Ergebnisse der vorangegangenen Experimente dargestellt und diskutiert. Die Ergebnisse der Validierung und des Tests wurden dabei direkt zusammengefasst. Die Ergebnisse sind nach den drei Bereichen Detektion, Segmentierung und Transformation gegliedert.

7.1 Ergebnisse Spike Detektion

In den folgenden Grafiken ist der Trainingsverlauf der Netzwerke zur Spike-Detektion dargestellt. Nach der Metaparameteroptimierung wurden die Netzwerke ausgewählt, die den geringsten Loss aufweisen. Innerhalb der folgenden vier Grafiken (7-1 - 7-4) ist der Verlauf des Trainings mit absoluten Daten als Eingabewert dargestellt.

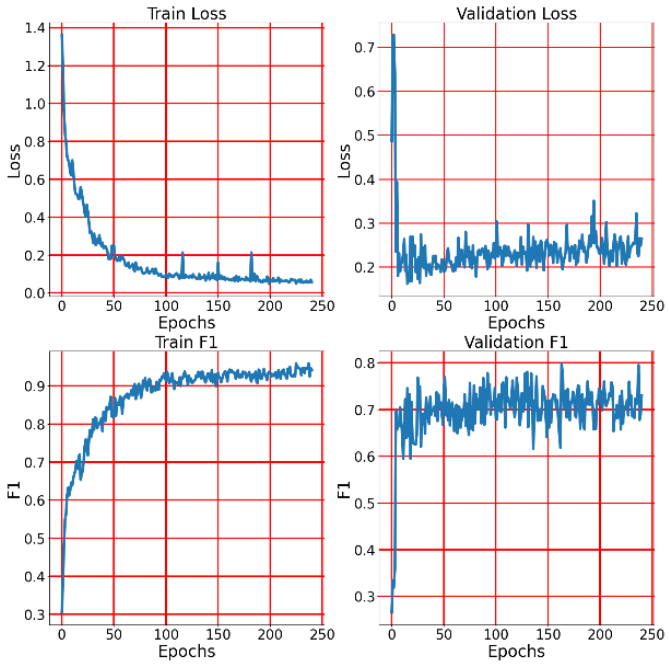


Abbildung 7-1 Trainingsverlauf Detektion mittels VGG mit Absoluten Eingangsdaten

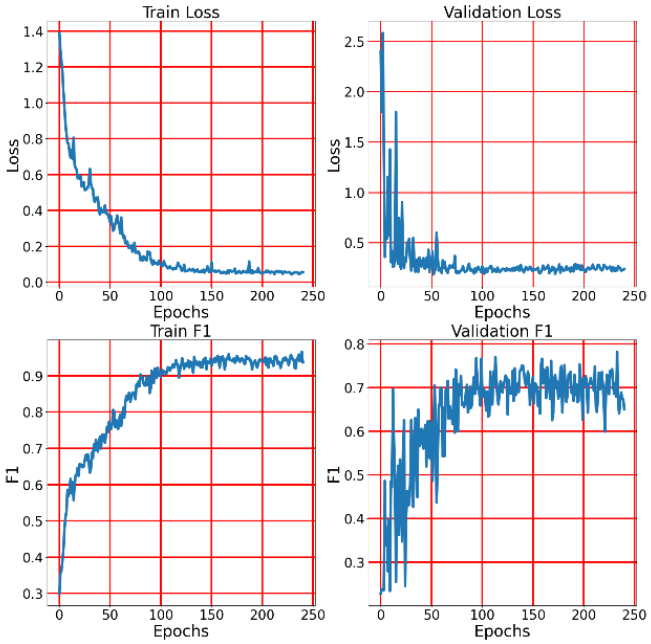


Abbildung 7-2 Trainingsverlauf Detektion mittels ResNet mit Absoluten Eingangsdaten

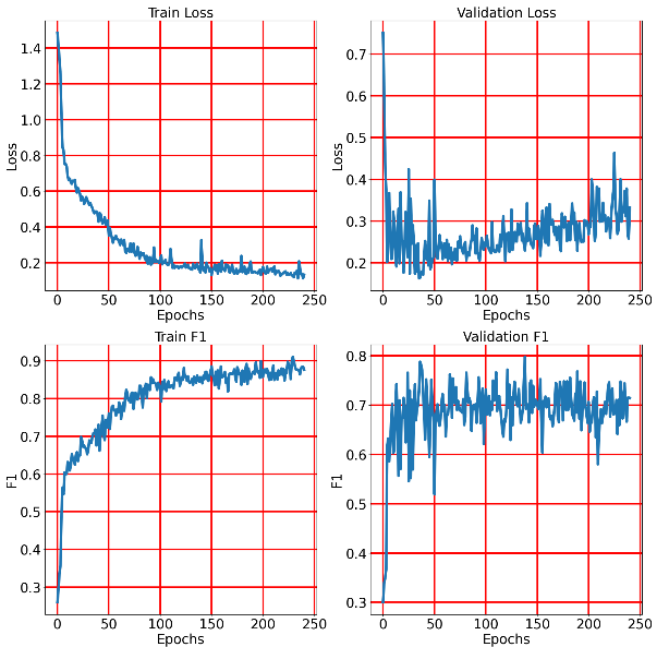


Abbildung 7-3 Trainingsverlauf Detektion mittels Shufflenet mit Absoluten Eingangsdaten

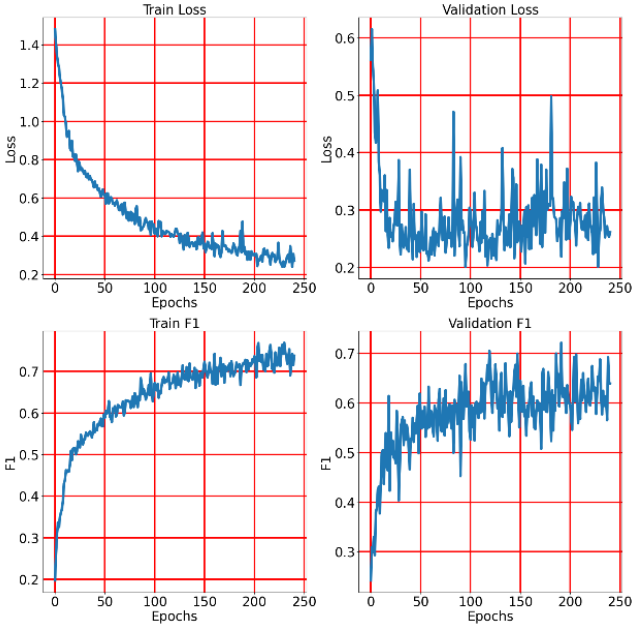


Abbildung 7-4 Trainingsverlauf Detektion mittels Densenet mit Absoluten Eingangsdaten

Die Trainingsverläufe in den Abbildungen 7-1 bis 7-4 zeigen, dass in allen Fällen ein klares Overfitting der Netzwerke vorliegt. Der erreichte F1 score innerhalb des Trainings liegt bei allen vier Netzwerken klar über den erreichten score innerhalb der Validierung.

In den folgenden Abbildungen (7-5 – 7-8) wird der Trainingsverlauf von der Detektion auf Basis von Real und Imaginärteil dargestellt.

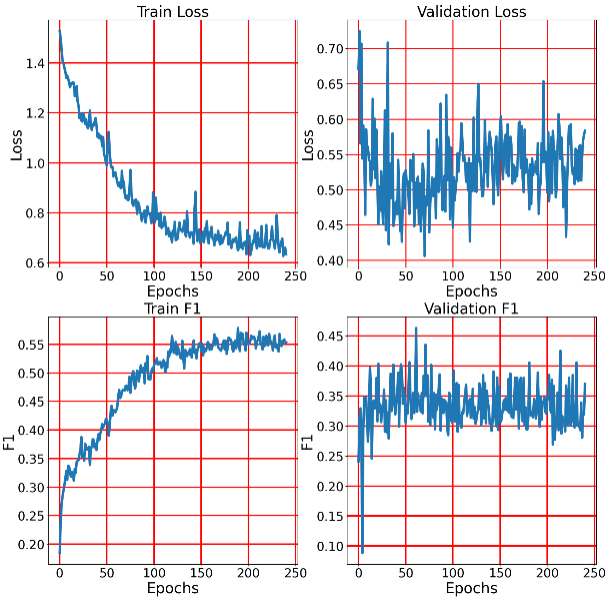


Abbildung 7-5 Trainingsverlauf Detektion mittels VGG mit Real- und Imaginärteil als Eingangsdaten

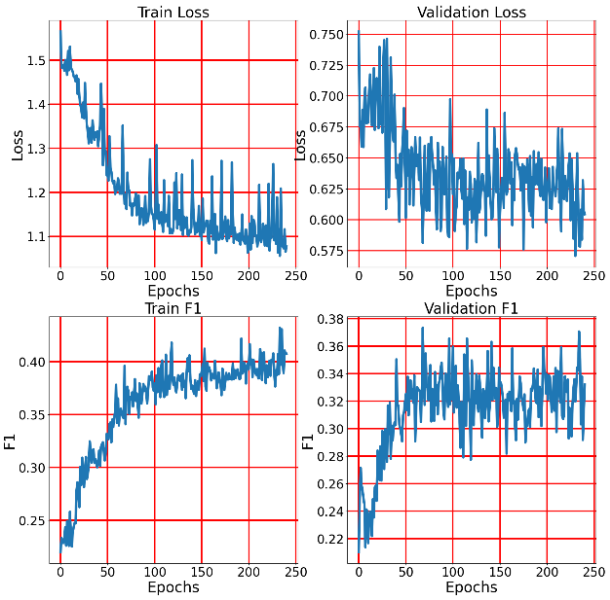


Abbildung 7-6 Trainingsverlauf Detektion mittels ResNet mit Real- und Imaginärteil als Eingangsdaten

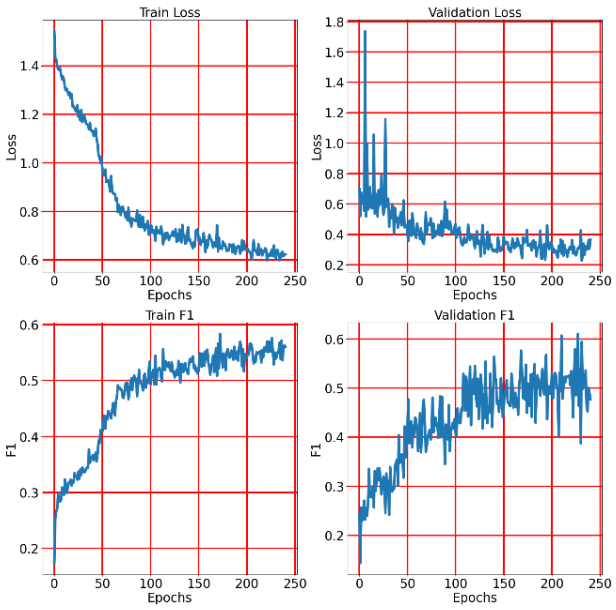


Abbildung 7-7 Trainingsverlauf Detektion mittels Shufflenet mit Real- und Imaginärteil als Eingangsdaten

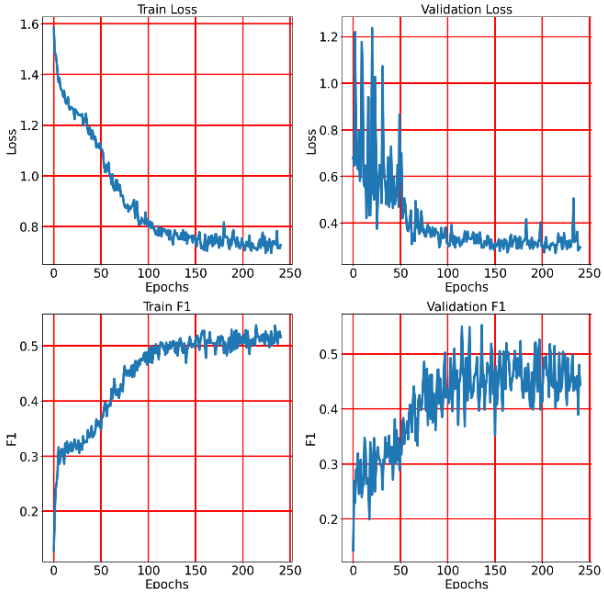


Abbildung 7-8 Trainingsverlauf Detektion mittels DenseNet mit Real- und Imaginärteil als Eingangsdaten

Der Verlauf des Trainings auf Basis der Real und Imaginärteile zeigen, dass ein erheblich weniger starkes Overfitting als bei den Absoluten Daten. Der F1 score liegt im Training nur leicht über den Validierungsdatensatz.

Die nächsten vier Abbildungen (7-9 – 7-12) zeigen das Ergebnis der Detektion auf Basis der L1 Norm als Eingangsdaten.

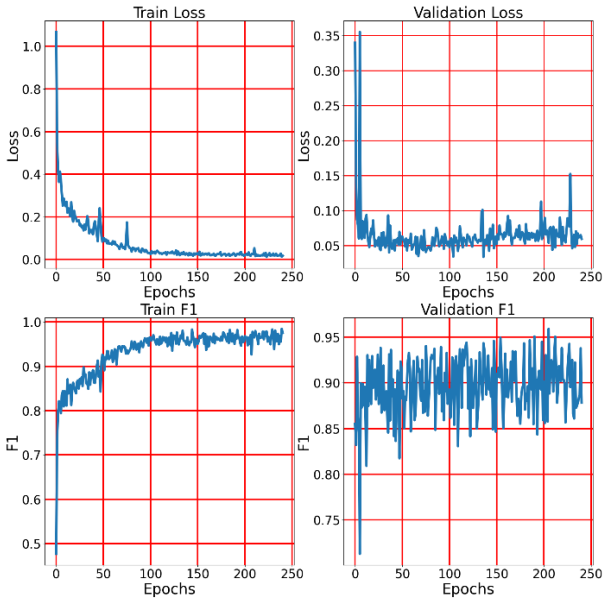


Abbildung 7-9 Trainingsverlauf Detektion mittels VGG mit L1 Norm als Eingangsdaten

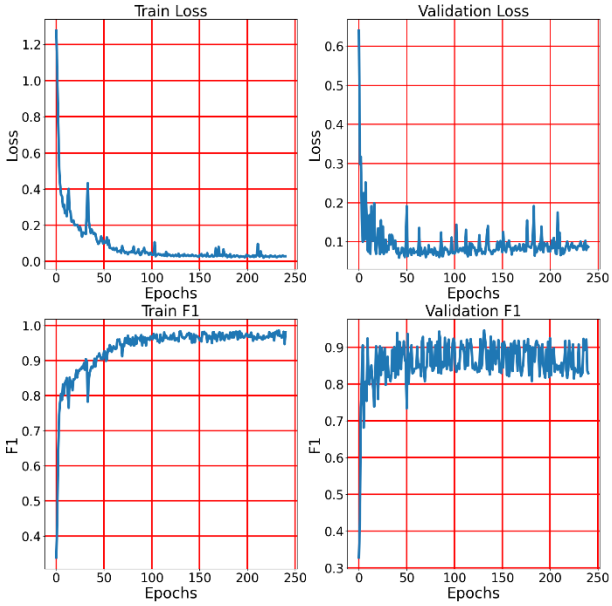


Abbildung 7-10 Trainingsverlauf Detektion mittels ResNet mit L1 Norm als Eingangsdaten

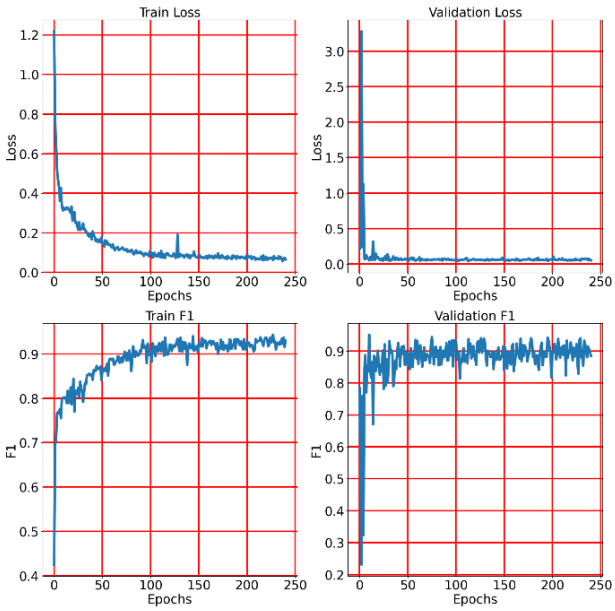


Abbildung 7-11 Trainingsverlauf Detektion mittels Shufflenet mit L1 Norm als Eingangsdaten

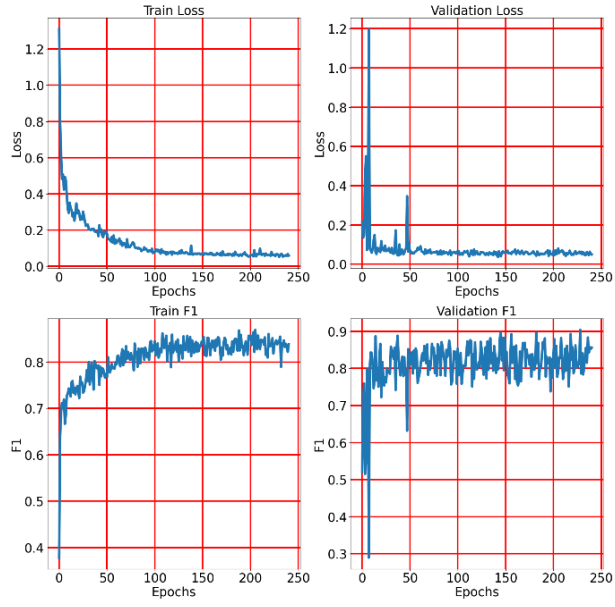


Abbildung 7-12 Trainingsverlauf Detektion mittels Densenet mit L1 Norm als Eingangsdaten

Der Verlauf des Trainings auf Basis der L1-Norm zeigt, dass der verwendete Loss gut optimiert werden konnte. Der F1-Score liegt in allen Fällen innerhalb des Trainings ähnlich wie in der Validierung, was auf ein geringes Overfitting hinweist.

In der folgenden Tabelle sind die Ergebnisse des Trainings und der Validierung der Detektionsalgorithmen dargestellt. Zusätzlich wurden die Netzwerke auf den zuvor extrahierten Testdatensatz getestet. Die Ergebnisse sind in die Tabelle eingetragen.

Tabelle 16 Ergebnisse des Experiments zur Detektion von Spikes mit VGG, ResNet, shufflenet und densenet Architektur

	VGG			***ResNet***		
Input	**L1**	**Abs**	**RealImag**	**L1**	**Abs**	**RealImag**
Optimizer	Adam	Adam	NAdam	Adam	Adam	NAdam
Learning Rate	0,0005	0,0003	0,001	0,0003	0,001	0,0005
Batchsize	32	256	64	64	32	32
Performance ms/Slice	15,5	16,4	8,975	27,64	37,7	25
Validation F1	0,959	0,799	0,31	0,934	0,7818	0,3063
Train F1	0,96	0,944	0,2577	0,971	0,9532	0,249
Test Recall	0,9032	0,7016	0,4839	0,9113	0,3548	0,5161
Test F1	0,8924	0,7039	0,2308	0,8726	0,6667	0,2503
	shufflenet_v2_x1_0			**densenet121**		
Input	**L1**	**Abs**	**RealImag**	**L1**	**Abs**	**RealImag**
Optimizer	Adam	NAdam	Adam	NAdam	Adam	Adam
Learning Rate	0,001	0,0003	0,0003	0,001	0,0001	0,001
Batchsize	32	32	16	16	16	16
Performance ms/Slice	731,9	760	523	937	691,5	907
Validation F1	0,937	0,875	0,594	0,797	0,6782	0,403
Train F1	0,933	0,731	0,545	0,858	0,754	0,544
Test Recall	0,898	0,6694	0,542	0,789	0,8	0,618
Test F1	0,8816	0,6919	0,523	0,807	0,7510	0,536

Bei der L1 Norm als Datengrundlage erreicht die VGG Architektur die besten Ergebnis mit einem F1 Score von 0,8924. Bei den Absolutwerten als Datengrundlage erreicht das Densenet mit einem score von 0,75 das Beste. Mit Real- und Imaginärteil als Datenformat erreicht das Densenet die Besten Ergebnisse mit einer L1-Norm von 0,536 auf Basis des Testdatensatzes.

7.2 Diskussion zur Spike Detektion

Das Experiment liefert eindeutige Ergebnisse, die darauf hinweisen, dass der Real- und Imaginärteil keine geeignete Grundlage für die Detektion von Spikes darstellen. Im Test wurde festgestellt, dass der F1-Score maximal 0,25 erreicht. Ein solcher Wert ist für eine zuverlässige Anwendung des Algorithmus unzureichend. Im Vergleich dazu erzielt der Absolutwert bessere Ergebnisse, wobei mithilfe des Densenets ein F1-Score von 0,75 erreicht werden kann.

Bei Verwendung der L1-Norm als Grundlage konnten wie erwartet die besten Ergebnisse erzielt werden. Das Shufflenet erreichte hier einen F1-Score von 0,878. Für die Anwendung des Algorithmus sind diese Ergebnisse ausreichend gut. Die experimentellen Ergebnisse zeigen deutlich, dass die L1-Norm als Datengrundlage für die Detektion von Spikes am besten geeignet ist und daher in der endgültigen Implementierung verwendet werden sollte. Eine mögliche Erklärung für die bessere Leistung der L1-Norm als Datengrundlage könnte in den größeren Intensitätsunterschieden zwischen den Spikes und dem umgebenden K-Raum liegen. Aufgrund dieser deutlicheren Gradienten können diese auch leichter von den Netzwerken erkannt werden. Basierend auf der Performance und den Ergebnissen wird das VGG Netzwerk für die endgültige Implementierung gewählt. Die F1 Score Ergebnisse sind hier am besten zudem benötigt das Netzwerk lediglich 15,5 ms zur Bearbeitung eines einzelnen Frames.

.

7.3 Ergebnisse Spike Segmentierung

In der folgender Kapitel ist der Trainingsverlauf der Netzwerke zur Spike Segmentierung dargestellt. Nach der Metaparmeteroptimierung wurden die Netzwerke ausgewählt, welche den größten IOU aufgewiesen haben. Innerhlab der fogenden Grafiken (7-13 - 7-16) ist der Verlauf des Trainings mit Absoluten Daten als Eingangswert dargestellt.

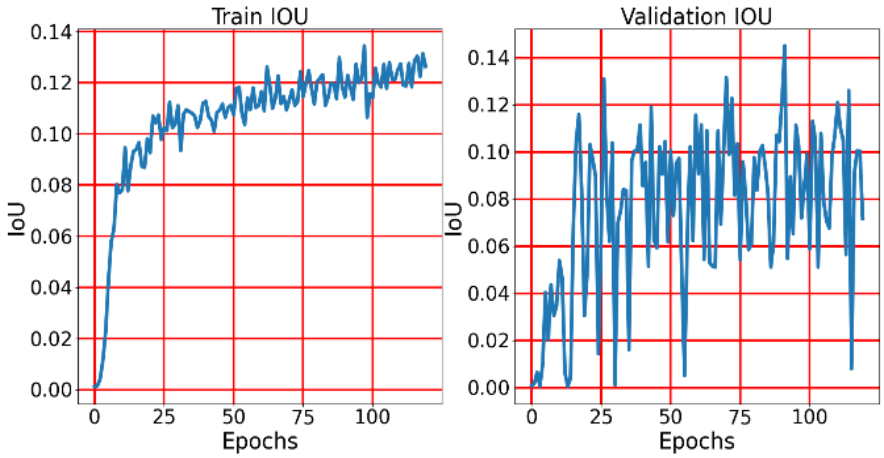


Abbildung 7-13 Trainingsverlauf Segmentierung mittels ResNet mit Absoluten Eingangsdaten

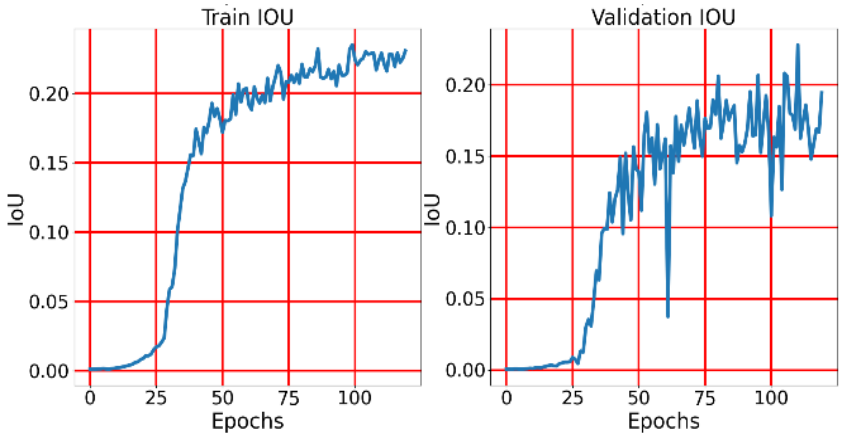


Abbildung 7-14 Trainingsverlauf Segmentierung mittels UNet mit Absoluten Eingangsdaten

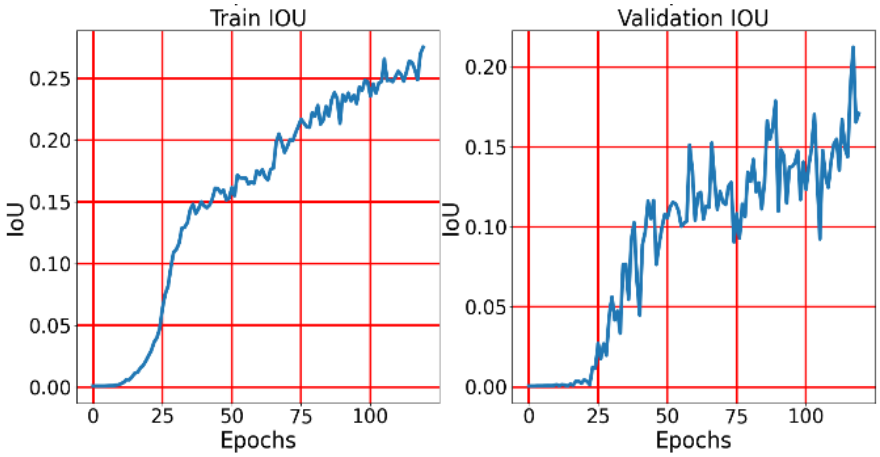


Abbildung 7-15 Trainingsverlauf Segmentierung mittels UNet Residuals mit Absoluten Eingangsdaten

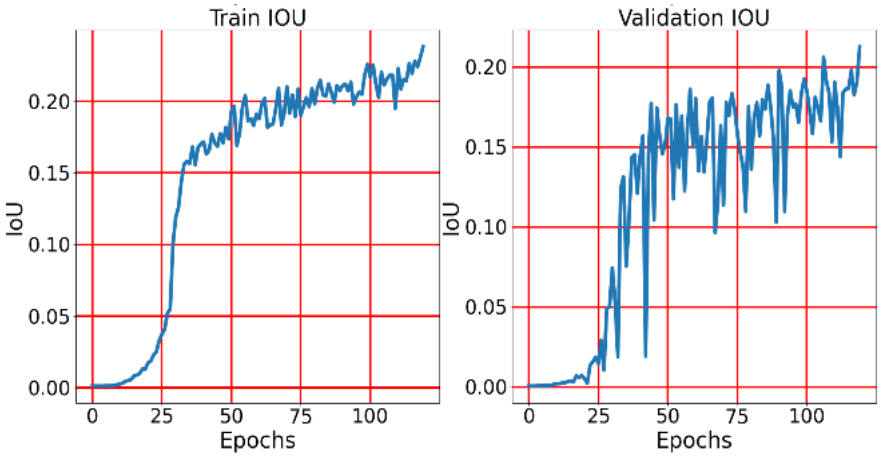


Abbildung 7-16 Trainingsverlauf Segmentierung mittels UNetPP mit Absoluten Eingangsdate

Aus den Trainingsverlauf lässt sich innerhalb jedes Netzwerks ein leichtes Overfitting erkennen. Die IOU-Ergebnisse der Trainingsdaten sind tendenziell höher als die der Validierungsdaten. Der IOU loss der Validierungsdaten liegt hier bei einem maximalen Wert von 0,24 innerhalb des UNet Residuals.

Die nachfolgende Darstellung (7-17 – 7-19) zeigt den Trainingsverlauf der Architekturen mit den getrennten Real- und Imaginärteilen als Eingangsdaten.

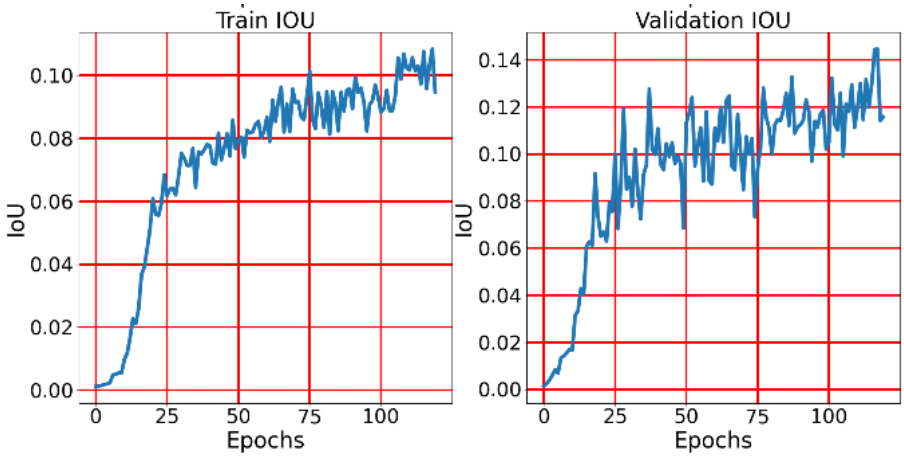


Abbildung 7-17 Trainingsverlauf Segmentierung mittels ResNet mit Real und Imaginärteil als Eingangsdaten

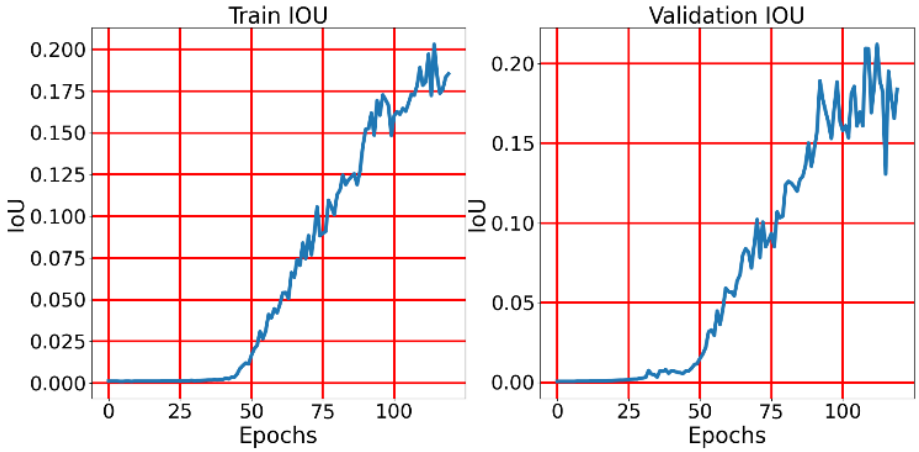


Abbildung 7-18 Trainingsverlauf Segmentierung mittels UNet mit Real und Imaginärteil als Eingangsdaten

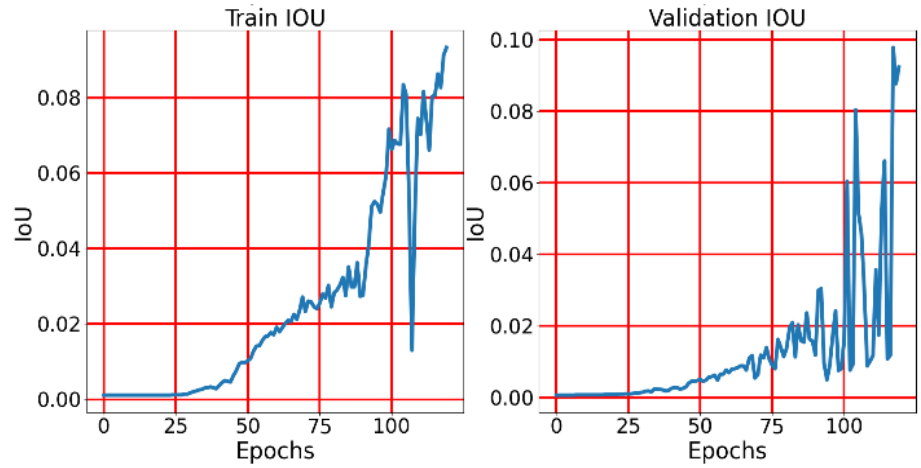


Abbildung 7-19 Trainingsverlauf Segmentierung mittels UNet Residuals mit Real und Imaginärteil als Eingangsdaten

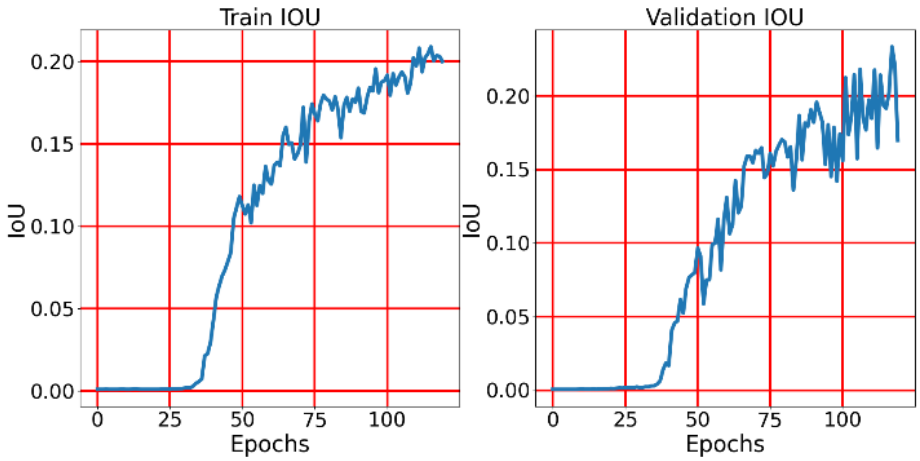


Abbildung 7-20 Trainingsverlauf Segmentierung mittels UNetPP mit Real und Imaginärteil als Eingangsdaten

Innerhalb des Trainings jedes Netzwerks lässt sich kein Overfitting feststellen, die Ergebnisse der Architekturen ist auf dem Validierungsdatensatz ähnlich gut wie bei den Trainingsdatensatz.

Die nachfolgende Darstellungen zeigt den Trainingsverlauf der Architekturen mit der L1-Norm als Eingangsdaten.

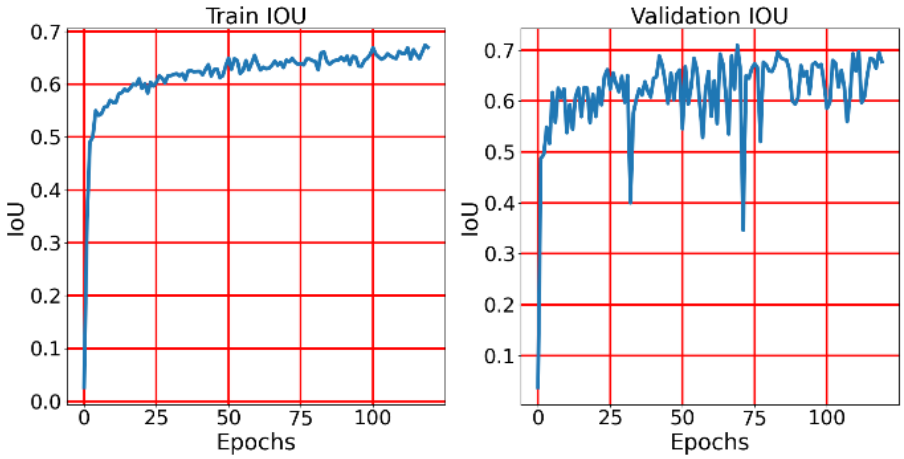


Abbildung 7-21 Trainingsverlauf Segmentierung mittels ResNet mit L1 Norm als Eingangsdaten

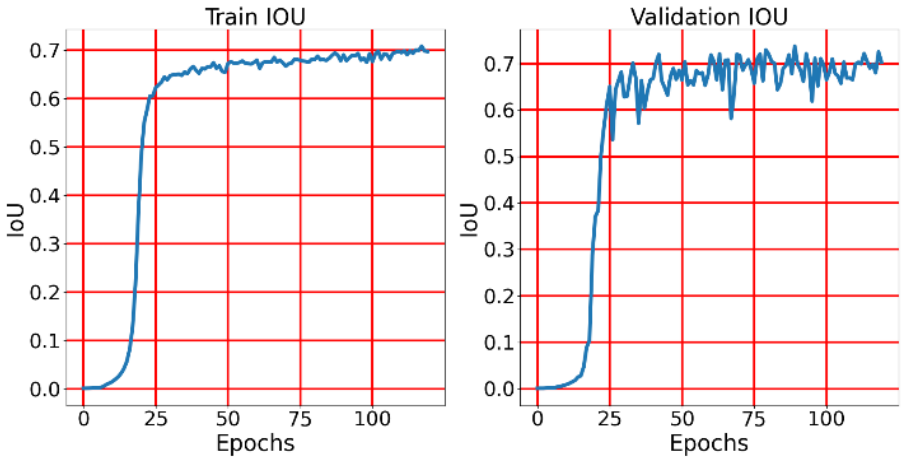


Abbildung 7-22 Trainingsverlauf Segmentierung mittels UNet mit L1 Norm als Eingangsdaten

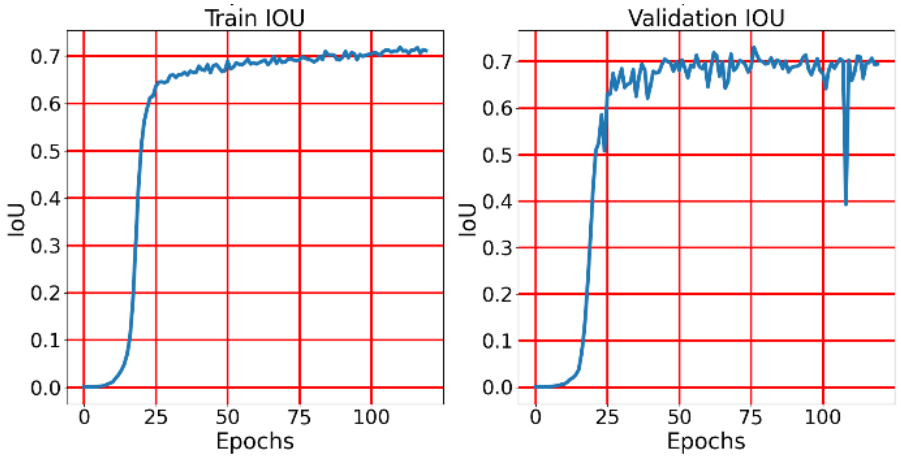


Abbildung 7-23 Trainingsverlauf Segmentierung mittels UNet Residuals mit L1 Norm als Eingangsdaten

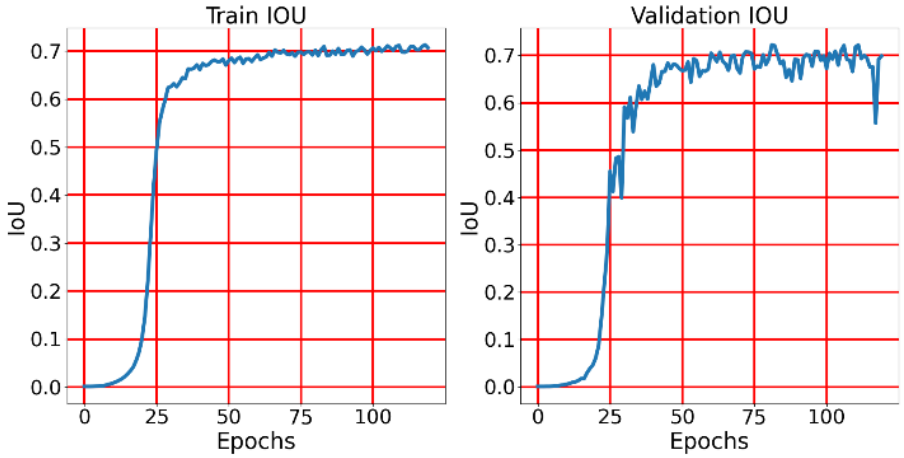


Abbildung 7-24 Trainingsverlauf Segmentierung mittels UNet PP mit L1 Norm als Eingangsdaten

Innerhalb keines der gezeigten Netzwerke lässt sich aus dem Trainingsverlauf ein Overfitting feststellen. Die Ergebnisse der Netzwerke auf Basis des Trainingsdatensatzes sind vergleichbar mit den Ergebnissen des Validierungsdatensatzes. Das UNetPP erzielt das beste Ergebnis mit einer IOU von 0,73 innerhalb der Validierung.

In der folgenden Tabelle sind die Ergebnisse des Trainings mit den dazugehörigen Metaparametern zusammengefasst. Zusätzlich wurden die Netzwerke mit den besten Validierungsergebnissen auf den Testdatensätzen angewendet. Die Ergebnisse des Tests sind ebenfalls innerhalb der unteren Tabelle vermerkt.

Um die Effektivität der Netzwerke zu ermitteln, wurde zudem die Verarbeitungszeit der Daten gemessen. Als Richtwert wurde dabei die Zeit ermittelt, die eine Architektur benötigt, um eine einzelne Matrix zu verarbeiten.

Tabelle 17 Ergebnisse des Experiments zur Segmentierung von Spikedaten mit ResNet, UNet, UNet Residuals und UNetPP Architektur.

	ResNet			**UNet**		
Input	**L1**	**Abs**	**RealImag**	**L1**	**Abs**	**RealImag**
Optimizer	Adam	NAdam	Adam	NAdam	NAdam	NAdam
Learning Rate	0,001	0,001	0,001	0,0005	0,0005	0,0005
Performance ms/Slize	259	285	181	108	92	102
Batchsize	4	8	4	4	4	4
Validation IOU	0,709	0,1176	0,1445	0,7375	0,2291	0,2121
Train IOU	0,631	0,1597	0,1033	0,6784	0,2279	0,1971
Test IOU	0,6877	0,0689	0,267	0,6787	0,1441	0,1029
	UNet Residuals			**UNet PP**		
Input	**L1**	**Abs**	**RealImag**	**L1**	**Abs**	**RealImag**
Optimizer	NAdam	NAdam	NAdam	NAdam	Adam	NAdam
Learning Rate	0,0005	0,001	0,001	0,001	0,001	0,0005
Performance ms/Slize	101	92	96	176	181	177
Batchsize	4	8	12	4	4	4
Validation IOU	0,7308	0,2488	0,0978	0,7329	0,2144	0,2339
Train IOU	0,6987	0,2123	0,08259	0,6895	0,217	0,2036
Test IOU	0,6717	0,169	0,04267	0,7005	0,203	0,153

Bei der L1 Norm als Datengrundlage erreicht das UNetPP die besten Ergebnis mit einer IOU von 0,7. Bei den Absolutwerten als Datengrundlage erreicht das UNetPP mit 0,20 erneut das Beste Ergebnis danach folgt das UNet Residuals mit einer IOU von 0,16, dass UNet mit 0,14

und schließlich das ResNet mit 0,06. Mit Real- und Imaginärteil als Datenformat erreicht das ResNet die Besten Ergebnisse mit einer IOU von 0,26 auf Basis des Testdatensatzes. Darauf folgt das UNetPP mit 0,15 das UNet mit 0,1 und letztlich das UNetPP mit einer IOU von 0,04.

7.4 Diskussion zur Spike Segmentierung

Die in Tabelle 17 präsentierten Ergebnisse zeigen deutlich, dass es innerhalb jeder Architektur eine starke Variation der Ergebnisse in Bezug auf die trainierten Datenformate gibt. Unter Verwendung des absoluten Werts als Eingabe sind die Testergebnisse vergleichbar mit den Trainingsergebnissen. Jedoch konnte keine der ausgewählten Architekturen die Spikes innerhalb der K-Räume erfolgreich segmentieren. Das beste Ergebnis zeigt, dass lediglich 20% der zu segmentierenden Fläche korrekt erkannt werden. Dieses Ergebnis reicht nicht aus, um eine signifikante Verbesserung der Bilddaten zu erzielen. Ähnlich ist das Ergebnis bei Verwendung der Real- und Imaginärdaten. Die Testergebnisse, dass diese Daten als Grundlage für die Spike-Segmentierung ungeeignet sind. Hier konnte maximal 26% der Spikes korrekt erkannt werden. Die Anwendung der L1-Norm auf die Real- und Imaginärdaten im Bildraum liefert sowohl während des Trainings als auch im Test die besten Ergebnisse. Das UNetPP kann dabei 70% der zu segmentierenden Fläche korrekt erkennen. Die benötigte Zeit zur Segmentierung beträgt 176 ms, was im Vergleich höher ist als beim UNet und UNetPP, die jeweils etwas mehr als 100 ms zur Segmentierung des K-Raums benötigen. Dennoch überwiegen die besseren IOU-Ergebnisse den zeitlichen Vorteil der beiden anderen Netzwerke. Das Ergebnis dieses Experiments verdeutlicht, dass die Verwendung der L1-Norm als Datengrundlage besser geeignet ist als der Absolutwert und der getrennte Real- und Imaginärteil. Eine mögliche Ursache für die bessere Leistung liegt in der höheren Intensitätsdifferenz der L1-Norm zwischen Spikes und dem umgebenden K-Raum. Dadurch wird der Gradient im K-Raum erhöht und ermöglicht es den Netzwerken, die gewünschten Bereiche korrekt zu segmentieren. Basierend auf diesen Erkenntnissen wird die L1-Norm als Datengrundlage gewählt und das UNetPP als Architektur implementiert, aufgrund der besseren IOU-Ergebnisse basierend auf dem Testdatensatz.

7.5 Ergebnisse Spike Transformation

In folgenden Grafiken (7-26 – 7-27) sind die Ergebnisse des Trainings verlauf der Netzwerke zur Spike Transformation dargestellt.

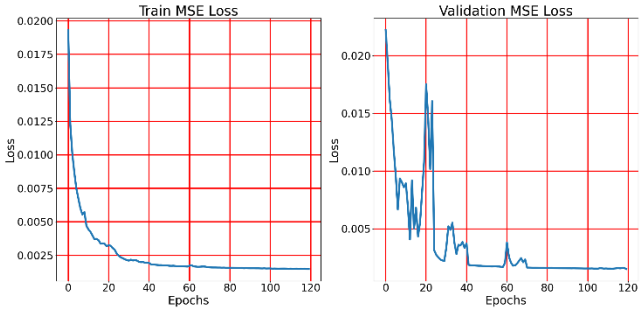


Abbildung 7-25 Trainingsverlauf Transformation mittels VGG

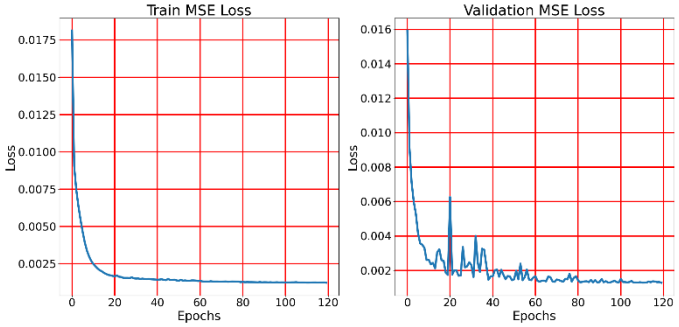


Abbildung 7-26 Trainingsverlauf Transformation mittels ResNet

Aus dem Trainingsverlauf der Transformation lässt sich keine Überanpassung (Overfitting) innerhalb der VGG-Architektur feststellen. Die Ergebnisse zwischen VGG und ResNet sind vergleichbar.

In der folgenden Tabelle sind die Ergebnisse des Trainings mit den verwendeten Metaparametern nochmals aufgelistet. Zusätzlich sind die Ergebnisse beider Netzwerke auf dem Testdatensatz dargestellt. Es wird auch speziell der Mean Squared Error (MSE) innerhalb der segmentierten Bereiche berechnet. Dieser ist unter der Metrik "Spezifischer MSE" dargestellt. Da nur die segmentierten Bereiche ersetzt werden sollen, ist dieser besonders relevant.

Tabelle 18 Ergebnisse des Experiments zur Transformation von Spikedaten der VGG und ResNet Architektur .

	VGG	**ResNet**
Optimizer	NAdam	NAdam
Learning Rate	0,01	0,005
Batchsize	512	1024
Performance ms/Slize	1,5	1,4
Validation MSE	0,001326	0,00128
Test MSE	0,001304	0,001267
Spezifischer MSE	0,0015668	0,00145

Das Ergebnis im Test MSE zeigt, dass es keinen klaren Unterschied zwischen den beiden Architekturen gibt. Das ResNet basierte Netzwerk erreicht die leicht besseren Ergebnisse mit einem MSE von 0,001267 gegenüber dem VGG basieren Netzwerk mit einem MSE von 0,001304. Der Spezifische MSE liegt bei beiden Netzwerken etwas über dem des gesamten Ausschnitts. Das VGG-Netzwerk erreicht hier einen loss von 0,00156, dass ResNet liegt bei 0,00145.

Zur Abschätzung der Netzwerke sollen die Ergebnisse nochmals optisch dargestellt werden. die Ergebnisse des Netzwerks mit der Grundwahrheit verglichen werden. Dazu werden randomisiert zwei Beispiele aus dem Testdatensatz ausgewählt. Die Ergebnisse sind in den unteren Darstellungen gezeigt.

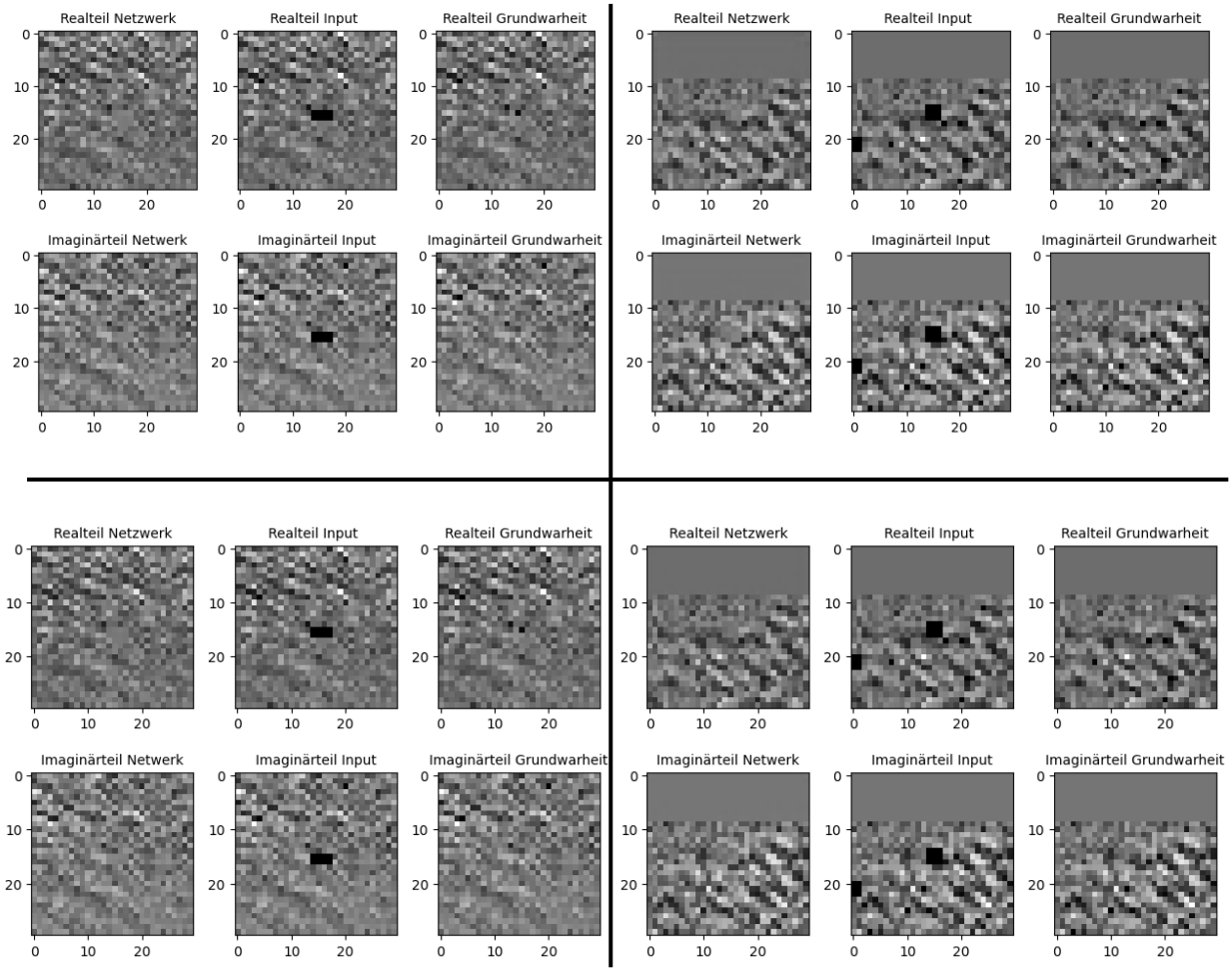


Abbildung 7-27 Vergleich von zwei Grundwahrheiten mit Output des ResNet (unten) und des VGG basierten Netzwerks

Die oberen zwei Beispiele zeigen die Ergebnisse des VGG basierten Netzwerks zur Spike Transformation. Die unteren zwei Beispiele zeigen das Ergebnis des ResNet basierten Netzwerks. Links sind dabei jeweils die Ausgaben von Realteil und Imaginärteil des Netzwerks dargestellt. Mittig ist der Input abgebildet mit genullten Spike welcher schwarz dargestellt ist. Rechts befindet sich die Grundwahrheit, also die Darstellung des K-Raum ohne Spike.

7.6 Diskussion zur Spike Transformation

Innerhalb der Validierung und des Tests lassen sich keine klaren Unterschiede in den Ergebnissen zwischen den beiden Netzwerken feststellen. Der Mean Squared Error (MSE) fällt für beide Netzwerke ähnlich aus. Bei direktem Vergleich des Outputs mit der Grundwahrheit ist gut zu erkennen, dass beide Netzwerke in der Lage sind, fehlende Daten gut zu substituieren. Die schwarzen gekennzeichneten Bereiche werden dabei mit vergleichbaren Daten zur Grundwahrheit substituiert. Demnach lässt sich auch hier kein deutlicher Unterschied zwischen den Architekturen feststellen. Um die Beurteilung der Ergebnisse zu vervollständigen, müssen jedoch auch die entstandenen K-Räume im Bildraum überprüft werden. Die Ergebnisse der Transformation werden daher im nächsten Kapitel im Bildraum dargestellt. Als Implementierung wurde das ResNet-basierte Netzwerk gewählt, da es einen leicht geringeren MSE aufweist als das VGG-basierte Netzwerk.

8 Implementierung und Test der gesamten Pipeline

Um den entstehenden Bildeindruck nach der Verarbeitung der MRT-Daten einschätzen zu können, werden die entwickelten Netzwerke innerhalb einer Bildverarbeitungs-Pipeline zusammengeführt. Die trainierten Modelle werden sequenziell geschaltet, um dem Algorithmus komplette Datensätze zuzuführen und die in Frage kommenden Daten zu filtern und zu bearbeiten. Der letzte Bearbeitungsschritt soll dabei variabel sein, sodass drei verschiedene Bildeindrücke entstehen.

Um die Wirkung der Algorithmen numerisch bewerten zu können, wurden MRT-Daten sowohl mit als auch ohne künstliche Spikes von Phantomen aufgenommen. Phantome sind wassergefüllte Behältnisse, die für Forschungszwecke innerhalb eines MRT-Geräts verwendet werden können. Dabei entsteht ein rundes Schnittbild des Behältnisses mit gleichmäßiger Intensität.

8.1 Implementierung der Pipeline

Zur Implementierung der Pipeline sollen komplette Datensätze verwendet werden. Diese enthalten einen Anteil an Bildern welche Spikes beinhalten. Die gesamte Pipeline ist als Verlaufs Diagramm in der unteren Darstelleng gezeigt.

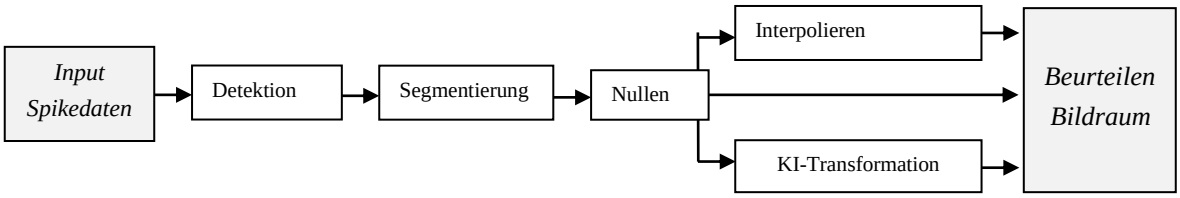


Abbildung 8-1 Gesamte Bildverarbeitungspipeline mit variabler Endstufe

Diese Datensätze werden zu Beginn der identischen Datenvorverarbeitung unterzogen, welche für das Training der KI-Netzwerke festgelegt wurde. Dabei werden zwei Versionen der Daten erstellt. Die Parallele Datenverarbeitung ist in folgender Abbildung visualisiert.

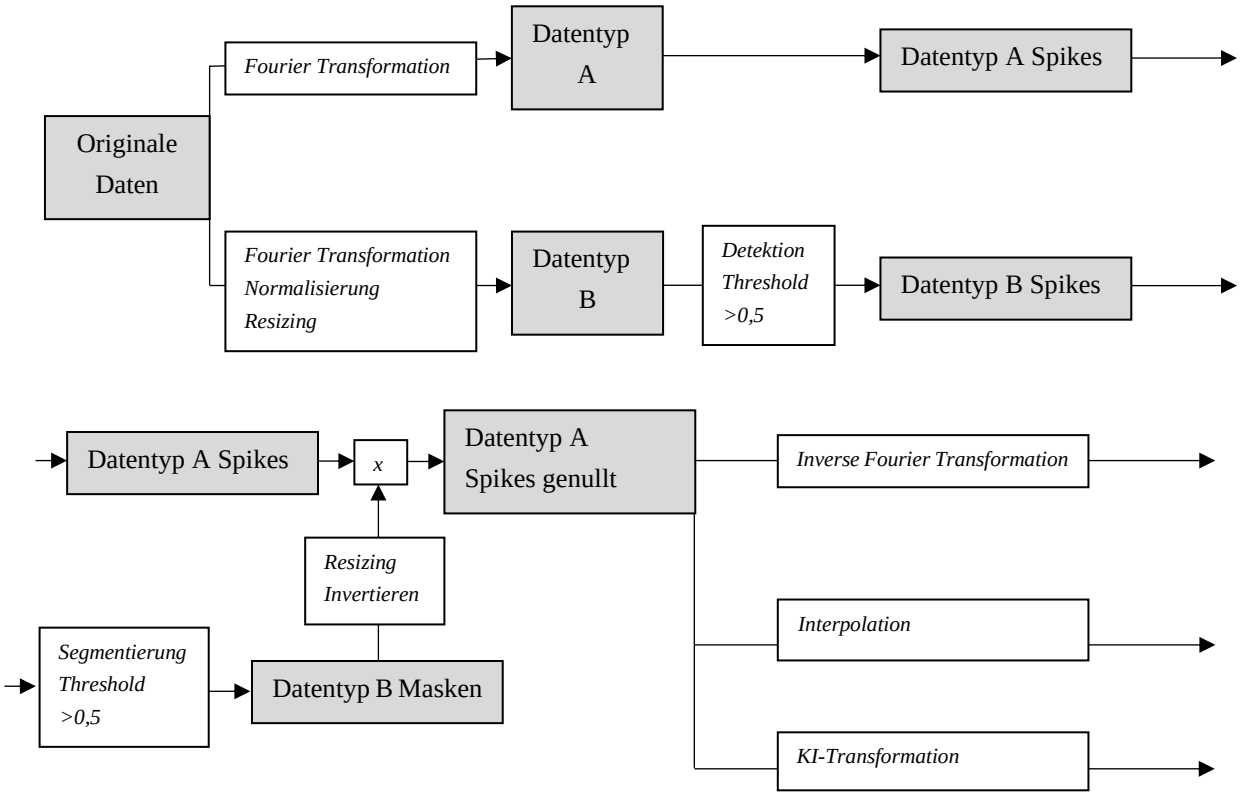


Abbildung 8-2 Ablaufdiagramm inklusive paralleler Datenverarbeitung der implementierten Pipeline

Die bearbeitete Version der Daten (Datentyp B) wird für die Detektion und Segmentierung der Spikes innerhalb des K-Raums verwendet. Die zweite Version der Daten (Datentyp A) wird lediglich durch Fourier Transformation in den K-Raum transformiert.

Zunächst sollen die K-Räume mit Spikes aus dem bereitgestellten Satz an Daten gefiltert werden. Dafür wird das im Kapitel 5.1.2 beschriebene VGG Netzwerk verwendet. Zur Detektion wird dabei der normalisierte Datentyp A verwendet. Wenn das Detektionsergebnis der Daten über 0,5 liegt, wird angenommen, dass der verwendete K-Raum Spikes enthält. In diesem Fall werden die entsprechenden Daten von Datentyp A und Datentyp B aussortiert, da sie keine Spikes enthalten.

Zur Segmentierung werden die Daten vom Typ B, die Spikes enthalten, an das im Kapitel 5.2.2 beschriebene UNetPP-Modell weitergeleitet. Bei der Segmentierung wird ein Schwellenwert von 0,5 verwendet. Die Pixel mit einer höheren Intensität innerhalb der erzeugten Maske werden als Spike bewertet. Die entstandene Maske wird gegebenenfalls auf die ursprüngliche Größe der Daten vom Typ A durch Resize-Operation transformiert. Zusätzlich wird die Maske invertiert, sodass die segmentierten Bereiche mit Spikes den Wert null haben und der restliche K-Raum den Wert eins hat. Anschließend wird die Maske mit dem originalen K-Raum vom Typ A multipliziert. Durch die Invertierung werden die segmentierten Bereiche mit Spikes auf den Wert null reduziert.

Durch das Entfernen des Rauschens wird der Frequenzanteil eliminiert, der das Muster im Bildraum erzeugt hat. Dadurch wird das Muster aus dem Bildraum entfernt. Die genullten K-Räume werden durch inverse Fourier Transformation in den Bildraum transformiert. Damit ist die erste Datenbasis für die Analyse des Bildraums definiert. Ein Nachteil des Entfernens der entsprechenden Daten ist der Verlust von Details im Bildraum. Es ist anzunehmen, dass die entfernten Frequenzen einen gewissen Anteil an der Pixelintensität im Bildraum hatten. Abhängig von der Größe und Lage des entfernten Bereichs kann dies zu einem Verlust von Details und Pixelintensität führen. Der Spike überdeckt jedoch die ursprüngliche Intensität. Um die Pixelintensität approximativ zu reproduzieren, werden zwei parallele Verarbeitungsoptionen implementiert.

Die erste Option besteht darin, die betroffenen Stellen durch Interpolation an die umgebenden Daten anzupassen. Dabei werden die entfernten Datenpunkte durch die Verteilung der nächstliegenden Datenpunkte ersetzt. Zur Implementierung wird in der Pipeline ein Medianfilter mit einem Kernel von 3x3 verwendet. Dieser filtert über einen Ausschnitt von

30x30 Pixeln um den segmentierten Spike herum. Das Ergebnis der Interpolation wird ausschließlich innerhalb der segmentierten Bereiche übernommen. Durch die Interpolation bleiben Pixel unverändert, die nicht von Spikes betroffen sind.

Die zweite Option zur Weiterverarbeitung besteht darin, die K-Raum-Ausschnitte an das im Kapitel 5.3.2 beschriebene ResNet weiterzuleiten. Dieses ersetzt die entsprechenden Datenpunkte durch wahrscheinliche Intensitäten. Der Output des Netzwerks ersetzt anschließend die segmentierten Bereiche innerhalb des K-Raums. Sowohl die Interpolation als auch die Transformation werden anschließend durch inverse Fourier Transformation in den Bildraum zurücktransformiert. Die drei entstandenen Bildergebnisse können anschließend analysiert und verglichen werden.

8.2 Analyse der Bildergebnisse

Um die zuvor implementierte Pipeline zu testen, wurden MRT-Aufnahmen mit künstlich erzeugtem Spike-Rauschen verwendet. Zur Erzeugung der Spikes wurde ein Piezo-Feuerzeug verwendet, das während der Aufnahme in der abgeschirmten Kabine des MRTs ausgelöst wurde. Dies führte zu einem elektrischen Feld, das vom Detektor im MRT erfasst wurde und zu Spike-Rauschen führte. Anschließend wurde dieselbe Aufnahme ohne die künstliche Spike-Erzeugung wiederholt, um eine Grundwahrheit der Aufnahme ohne Störungen zu haben. Als Testobjekt wurde ein Phantom verwendet, um eine erste Aussage über die Generalisierungsfähigkeit des Netzwerks treffen zu können, nachdem es auf K-Raum-Daten von Hirn- und Thoraxaufnahmen trainiert wurde. Abbildung 8-3 zeigt den K-Raum des Phantoms zusammen mit dem entsprechenden Bildraum.

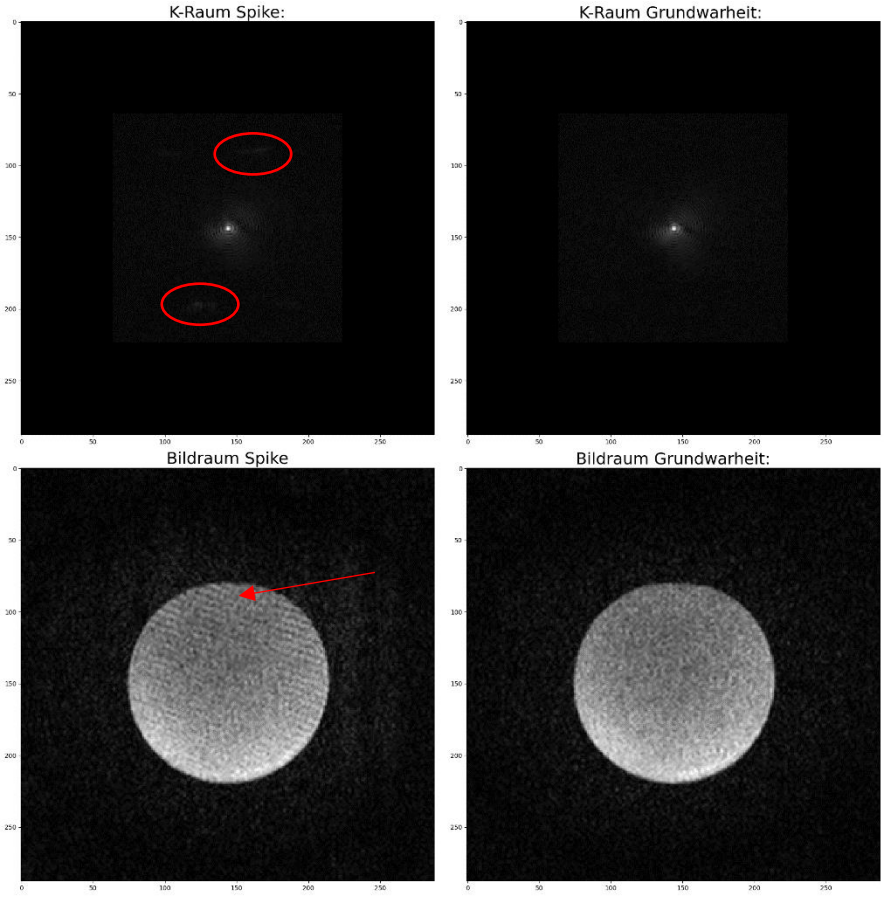


Abbildung 8-3 MRT-Aufnahme mit Spike und ohne Spike

In Abbildung 8-3 ist rechts die Aufnahme des Phantoms als Grundwahrheit ohne Spikes zu sehen. Links ist die identische Aufnahme des Phantoms mit Spikes dargestellt, wobei die Spikes innerhalb des K-Raums rot umrandet sind.

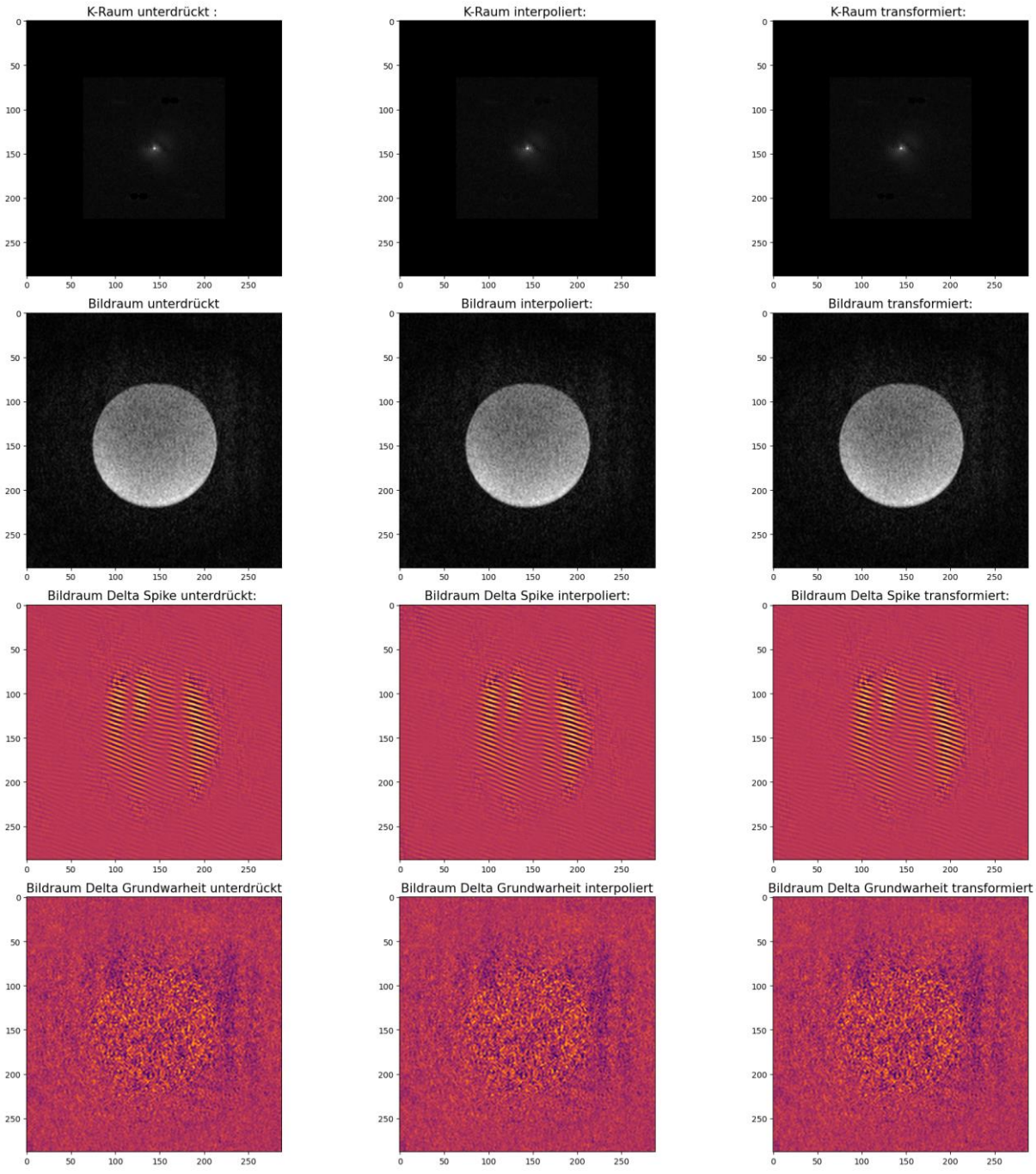


Abbildung 8-4 Vergleich der Bildeindrücke zwischen den künstlich erzeugten Spike und der zur Verfügung gestellten Grundwahrheit.

Deutlich sichtbar ist die Musterung der Streifen, die von rechts unten nach links oben verläuft. Diese Musterung ist im Bildraum durch den roten Pfeil gekennzeichnet. Rechts ist das Schnittbild des Phantoms ohne Spikes zu sehen. Der Datensatz mit Spikes wird durch die Pipeline bearbeitet, wodurch drei verschiedene Bildergebnisse entstehen, wie im vorherigen Kapitel definiert. Um die Leistung der Pipeline zu testen, werden die bearbeiteten Bilder direkt mit der Grundwahrheit ohne Spikes verglichen (Abbildung 8-4). Dadurch kann die Performance der einzelnen Schritte der Pipeline direkt anhand der Grundwahrheit gemessen werden.

In Abbildung 8-4 werden die Ergebnisse der Pipeline dargestellt, wobei die Daten aus Abbildung 8-3 verwendet wurden. In der linken Spalte ist das Ergebnis der direkten Unterdrückung der Spikes im K-Raum zu sehen. In der Mitte sind die Ergebnisse der Interpolation dargestellt und rechts die Ergebnisse der Transformation. Die erste Reihe der Abbildung zeigt die K-Räume der bearbeiteten Daten. In der zweiten Reihe sind die Bildräume zu den bearbeiteten K-Räumen zu sehen. In keinem der drei Bildräume ist eine Musterung erkennbar. In der dritten Reihe wird das Delta zwischen dem Bildraum mit Spikes und der jeweiligen Endbearbeitung der Pipeline dargestellt.

Hier ist die zuvor entfernte Musterung deutlich erkennbar. Daher kann gesagt werden, dass die Bearbeitung der K-Raum-Daten zu einer Verbesserung des Bildeindrucks führt. Die vierte Reihe zeigt das Delta zwischen der Grundwahrheit ohne Spikes und der jeweiligen Bearbeitung innerhalb der Pipeline. Hier zeigt das Delta keine Musterung mehr, was darauf hinweist, dass die gesamte Musterung, die durch die Spikes verursacht wurde, entfernt werden kann. Zur numerischen Einordnung der Ergebnisse wurde zusätzlich der Mittelwert des entstehenden Deltas für jede Endstufe und den K-Raum ohne Spikes berechnet. Es wurde auch der identische Wert für den ursprünglichen K-Raum mit Spikes und den ohne Spikes berechnet. Darüber hinaus wurde der Structural Similarity Index (SSIM) für alle Kombinationen berechnet. Der SSIM wurde 2004 von Zhou Wang et al. eingeführt [Wang04] und misst die Ähnlichkeit zwischen zwei Bildern. Er berücksichtigt Helligkeit, Kontrast und Struktur beider Bilder und liefert ein Ergebnis zwischen -1 und 1, wobei 1 für ein identisches Bild steht. Die Ergebnisse sind in der folgenden Tabelle aufgeführt:

Tabelle 19 Numerischer Vergleich der Pipeline Ergebnisse mit dem K-Raum ohne Spike

Delta	Ergebnis Mittelwert	Ergebnis SSIM
K-Raum mit Spikes – K-Raum ohne Spikes	0.0002274	0,998
K-Raum unterdrückt – K-Raum ohne Spikes	0.00022695	0,9982
K-Raum interpoliert – K-Raum ohne Spikes	0.00022887	0,9982
K-Raum segmentiert – K-Raum ohne Spikes	0.000226962	0,9982

In den Ergebnissen aus Tabelle 17 wird deutlich, dass sowohl der MSE als auch der SSIM keine klare Aussage über die Verbesserung der Bildqualität zulassen. Der geringe Unterschied im SSIM-Wert zwischen den Bildern mit und ohne Spikes deutet bereits darauf hin, dass nur ein geringfügiger numerischer Unterschied durch das Spike-Rauschen im Bildraum entsteht. Der SSIM-Wert der Pipeline-Endstufen ist nur geringfügig höher als der Wert ohne jegliche Bearbeitung. Daher kann die Qualität der Pipeline nur visuell beurteilt werden.

8.3 Vergleich mit Regelbasierten Algorithmus

Im Rahmen der Bewertung der implementierten Pipeline wird zusätzlich ein regelbasierter Algorithmus getestet. Dabei wird der bildbasierte Algorithmus von Staemmler, wie im Kapitel 3 des Standes der Technik beschrieben, verwendet. Als Datengrundlage dient der Testdatensatz mit Phantomaufnahmen. Um den Algorithmus anzuwenden, werden die Real- und Imaginärteile der Schichtbilder separat verwendet. Dadurch kann eine nachträgliche Transformation in den Bildraum erfolgen. Der Algorithmus wurde leicht angepasst, um ihn für den vorliegenden Datensatz zu optimieren. Hierbei wurden die Fenstergröße und die Sensibilität der Pixeldetektion experimentell optimiert. Die besten Ergebnisse wurden mit einer Fenstergröße von 22 Pixeln und einer Sensibilität von 17 erzielt. Aufgrund der erhöhten Pixelschwankungen wurde das Zentrum des K-Raums von der Detektion ausgeschlossen. Die Ergebnisse der regelbasierten Detektion sowie die Ergebnisse des VGG Netzwerks basieren auf dem identischen Datensatz und sind in der folgenden Tabelle aufgeführt.

Tabelle 20 Ergebnisse des Staemmler Detektionsalgorithmus verglichen mit VGG

	Staemmler	**VGG**
Precision	0,1139	1
Recall	0,9	0,8
F1	0,202	0,889

Die Ergebnisse legen nahe das der Staemmler Algorithmus zur Detektion von Spikes nur beschränkt geeignet ist. Der Algorithmus liefert zwar einen Recall von 1 jedoch zeigt die Precision von 0.12 das der Algorithmus sehr sensibel auf die vorliegenden Daten reagiert. Die VGG KI dagegen kann erheblich besser zwischen den K-Räumen mit Spike und denen ohne erkennbaren Spike unterscheiden. Die Precision liegt hier bei 1 und der Recall bei 0,8. In Folgenden Abbildungen sind die Bildergebnisse des Staemmler Algorithmus gezeigt. Dabei soll die entstehende Bildqualität des Staemmler Algorithmus getestet werden.

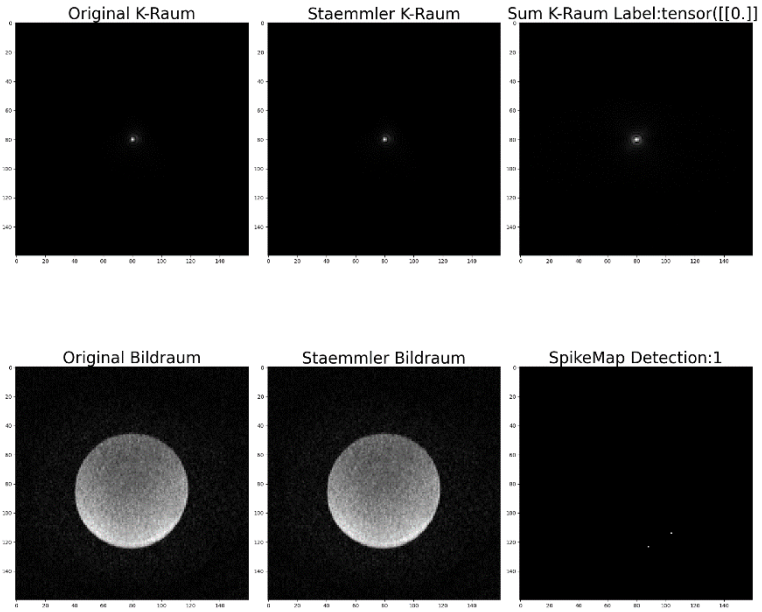


Abbildung 8-5 Ergebnisse des Staemmler Algorithmus auf einem K-Raum ohne Spike Label

Ein Großteil der durch den Staemmler Algorithmus bearbeiteten Daten ohne Spike Label zeigen das lediglich wenige Pixel als Spike durch den Algorithmus detektiert werden wie in Abbildung 8-5 gezeigt wird. Die Verarbeitung der Daten durch den Algorithmus zeigen keine Auswirkungen auf den entstehen Bildraum. Dennoch testet der Algorithmus die Daten Positiv

auf Spikes. In folgenden Darstellung 8-6 und 8-7 werden die Ergebnisse des Staemmler Algorithmus basierend auf zwei K-Räumen mit Spike Label dargestellt.

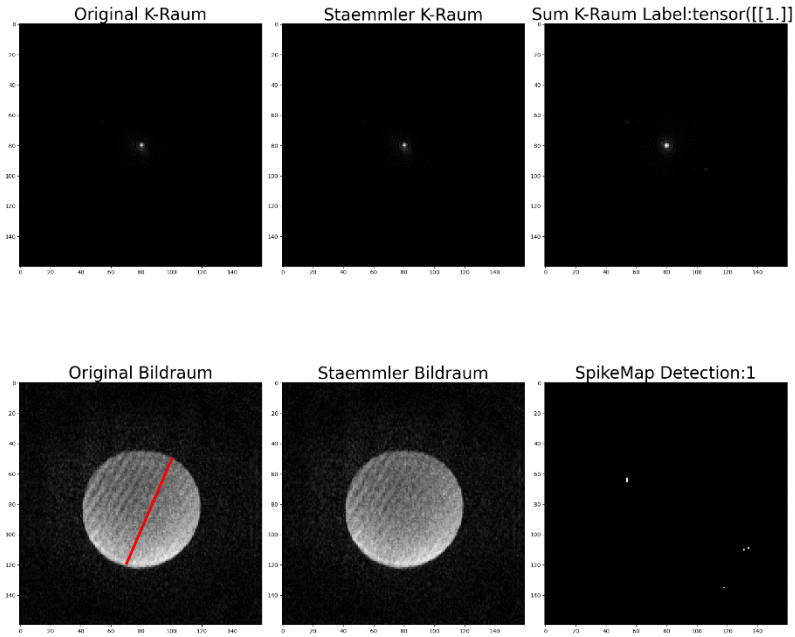
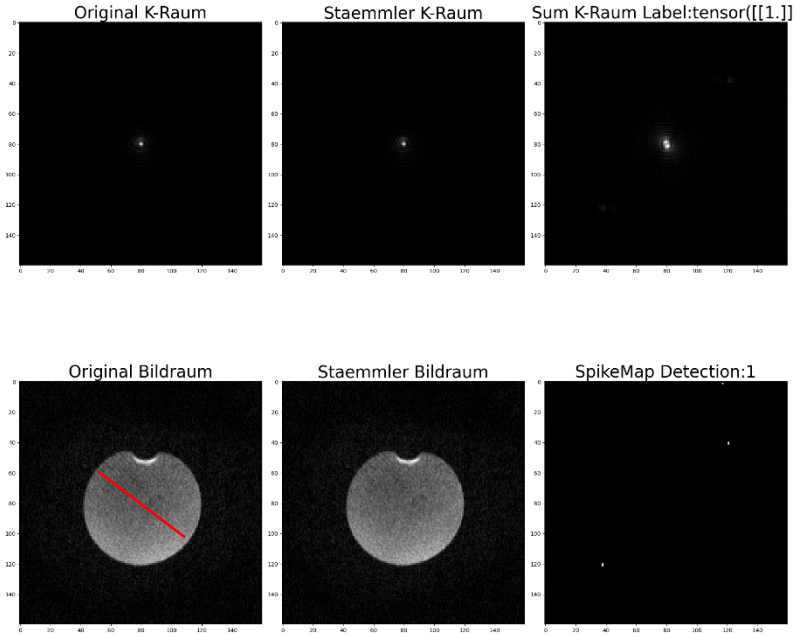


Abbildung 8-6 Ergebnisse des Staemmler Algorithmus auf einem K-Raum mit Spike Label 1



Die Ergebnisse des Algorithmus basierend auf Daten mit Spikes innerhalb das K-Raums zeigen das innerhalb des Bildraums keine Verbesserungen erzielt werden können. Das Streifenmuster

in Abbildung 8-5 und 8-6 ist ebenfalls nach der Detektion und Verarbeitung der Daten stark sichtbar. Das Streifenmuster ist in seinem Verlauf in beiden Abbildungen rot dargestellt.

Der Algorithmus soll nun optimiert werden, um eine Verbesserung in der entstehenden Bildqualität zu erreichen. Dazu wird eine Fenstergröße von 18 Pixel und eine Sensibilität von 12 verwendet werden. In den folgenden Grafiken wird das Ergebnis des Staemmler Algorithmus verglichen mit den Ergebnissen der entwickelten Deep Learning Verarbeitung.

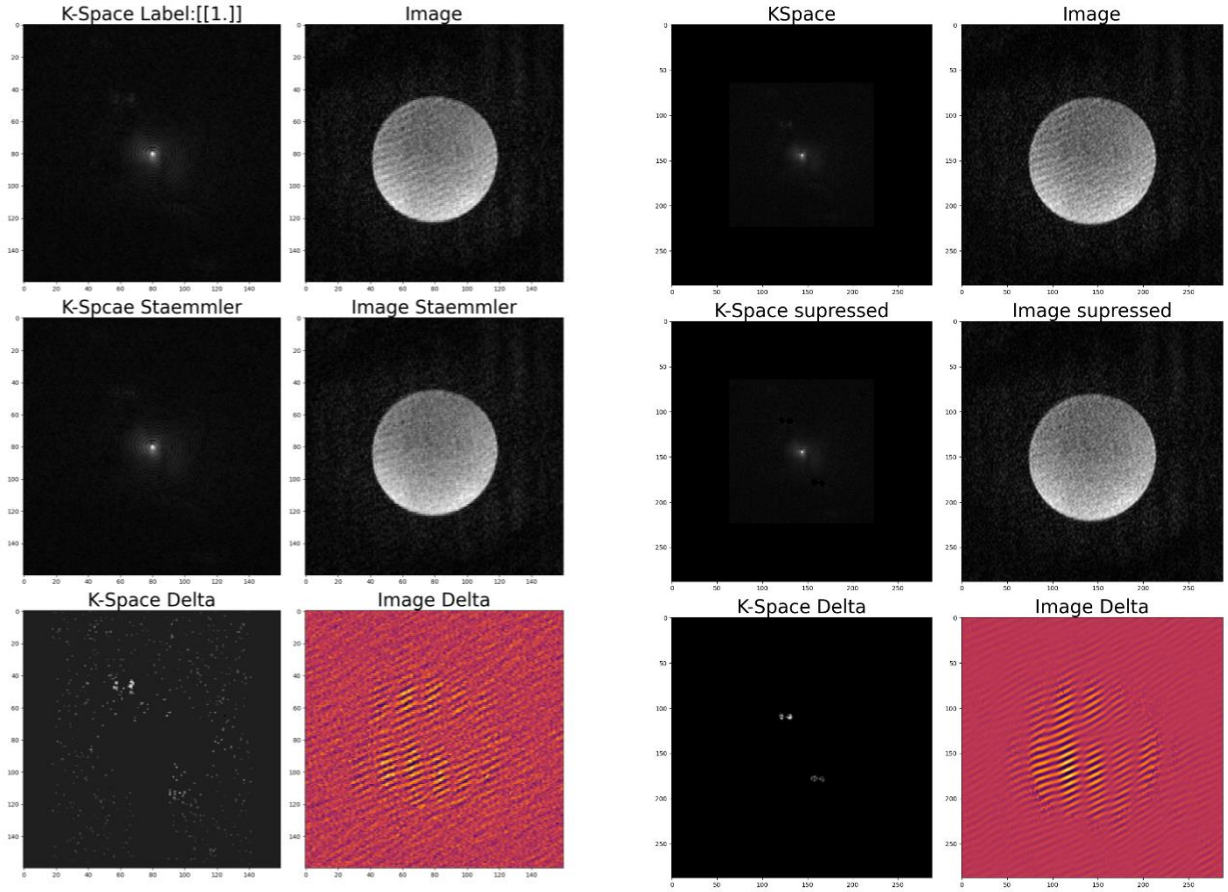
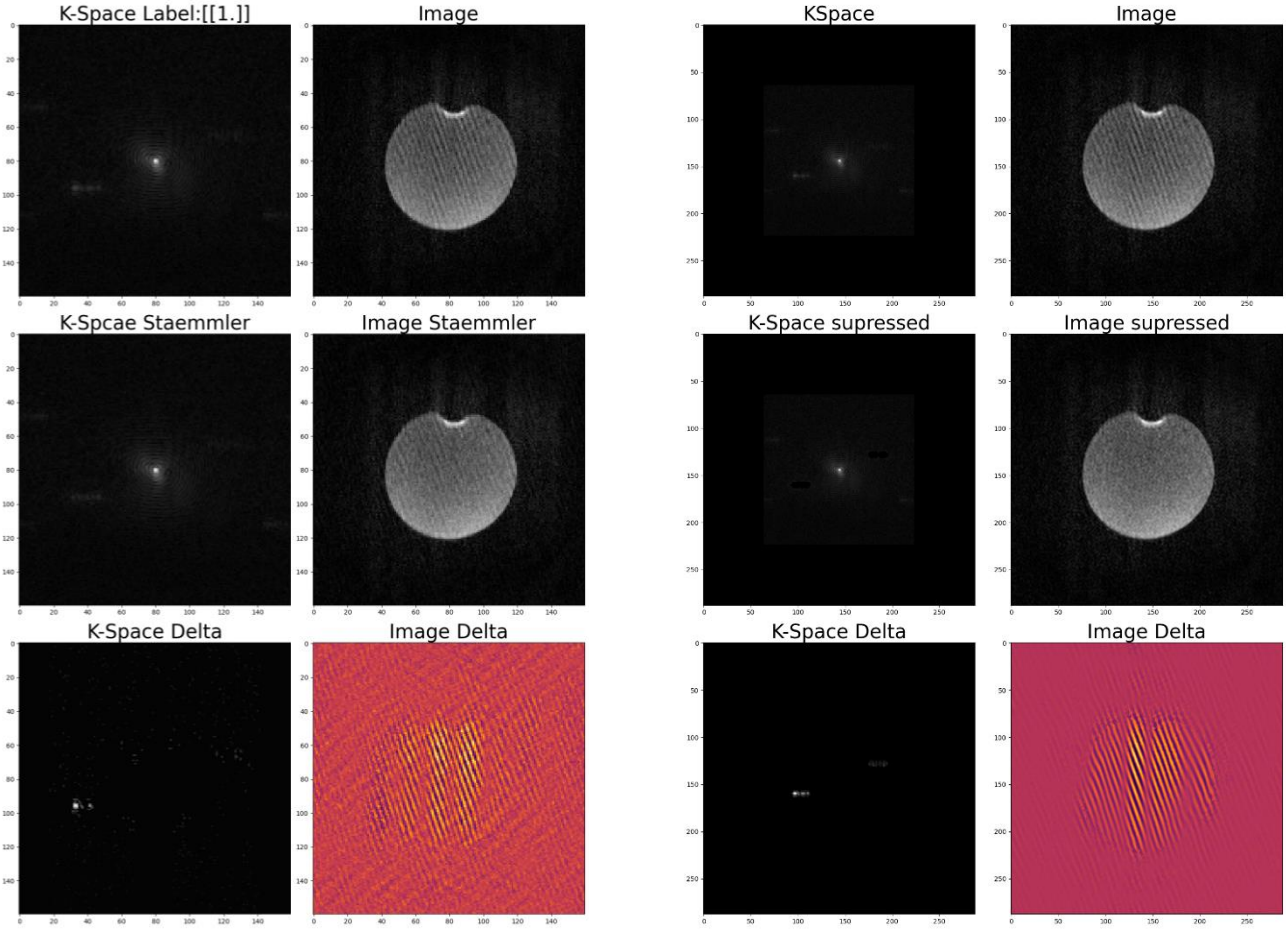


Abbildung 8-7 Erster Vergleich der Verarbeitung des Staemmler Algorithmus (a) mit dem Ergebniss des Deep Learning Algorithmus (b)



a b

Abbildung 8-8 Zweiter Vergleich der Verarbeitung des Staemmler Algorithmus (a) mit dem Ergebniss des Deep Learning Algorithmus (b)

Die Abbildungen 8-7 und 8-8 stellen jeweils eine Schicht des aufgenommenen K-Raums dar, in dem Spikes identifiziert wurden. Abschnitt a zeigt den K-Raum nach der Verarbeitung durch den Staemmler-Algorithmus, während Abschnitt b den K-Raum nach der Verarbeitung durch den Deep-Learning-Algorithmus zeigt. Die erste Zeile jeder Abbildung enthält den ursprünglichen K-Raum sowie den entsprechenden Bildraum. Die zweite Zeile jedes Abschnitts zeigt den bearbeiteten K-Raum und den daraus resultierenden Bildraum. Die dritte Zeile stellt das Delta des K-Raums und des Bildraums dar.

Beide Beispiele verdeutlichen, dass der Staemmler-Algorithmus zur Verbesserung des Bildeindrucks beitragen kann. In beiden Fällen wurde das durch die Spikes erzeugte Linienmuster durch die Anwendung des Algorithmus reduziert. Das Delta des Bildraums zeigt deutlich die verringerte Musterung. Im Delta des K-Raums zeigt sich jedoch, dass auch eine beträchtliche Anzahl von Pixeln fälschlicherweise als Spikes klassifiziert wird. Dies führt dazu, dass auch eine große Anzahl korrekter Daten manipuliert dargestellt wird. Im Delta des Bildraums wird dies durch ein Rauschen deutlich, das dem reduzierten Linienmuster überlagert wird.

Im Gegensatz dazu zeigt die Deep-Learning-basierte Verarbeitung der Daten eine höhere Präzision bei der Bearbeitung der mit Spikes belasteten Pixel. Darüber hinaus kann der Algorithmus die auftretenden Musterungen im Bildraum stärker reduzieren. Der direkte Vergleich legt nahe, dass der entwickelte Deep-Learning-Algorithmus in der Detektion und Verarbeitung von mit Spikes belasteten K-Raum-Daten in diesem Fall überlegen ist. Um jedoch eine allgemeinere Aussage über den direkten Vergleich zu treffen, sollten eine größere Anzahl an heterogen verteilten MRT-Daten getestet werden.

9 Zusammenfassung und Ausblick

9.1 Zusammenfassung

Innerhalb der Masterarbeit wurde untersucht, ob mittels Deep-Learning-Algorithmen das Spike-Rauschen innerhalb von MRT-Daten effektiv entfernt werden kann. Dafür wurde zunächst eine Datenvorverarbeitung implementiert, die als Vergleichsgrundlage für das folgende Training diente. Es wurden drei verschiedene Datenverarbeitungsmethoden entwickelt.

Die Arbeit wurde dann in drei Teilbereiche aufgeteilt: Detektion, Segmentierung und Transformation. Für den Teilbereich der Spike-Detektion wurden vier verschiedene Netzwerke trainiert, basierend auf VGG, ResNet, Densenet und Shufflenet. Diese wurden genauer untersucht und ihr Aufbau definiert. Im Training wurden die vier Netzwerke miteinander verglichen, basierend auf den verfügbaren Datenformaten. Dabei zeigte sich, dass die L1-Norm als Datengrundlage die besten Ergebnisse für die Detektion von Spikes im K-Raum lieferte. In Bezug auf die Architekturen gab es nur geringe Unterschiede. Als Implementierung wurde das ResNet aufgrund seiner schnelleren Bildverarbeitung im Vergleich zu den anderen Netzwerken ausgewählt.

Für die Segmentierung der Spikes im K-Raum wurden erneut vier verschiedene Netzwerke ausgewählt: ein ResNet-basiertes Netzwerk, ein UNet, ein UNet mit Residuals und ein UNetPP. Diese Netzwerke wurden genauer betrachtet, ihr Aufbau und ihre Funktionsweise wurden erläutert. Im Vergleich der Netzwerke zeigte sich erneut, dass die L1-Norm als Datenformat die besten Ergebnisse lieferte. Die UNetPP-Architektur erzielte leicht bessere Ergebnisse als die anderen Netzwerke und wurde daher als Implementierung ausgewählt.

Für den letzten Teilbereich, die Transformation, wurden zwei Netzwerke als Vergleichsgrundlage verwendet: eine stark verkleinerte Version eines VGG-basierten Netzwerks und eine stark verkleinerte Version eines ResNet-basierten Netzwerks. Hier mussten der Real- und Imaginärteil verwendet werden, um eine Rücktransformation in den Bildraum zu ermöglichen. Es zeigte sich, dass beide Netzwerke ähnlich gute Ergebnisse basierend auf dem MSE erzielten. Auch der visuelle Vergleich zwischen den Grundwahrheiten und den Ausgaben der Netzwerke ergab keine klar bessere Performance. Schließlich wurde das ResNet-basierte Netzwerk ausgewählt.

Zuletzt wurde die Komposition aller trainierten Netzwerke innerhalb einer Pipeline getestet. Dazu wurden Phantome innerhalb des MRT aufgenommen und mit einem Piezo-Feuerzeug künstlich Spikes erzeugt. Dabei wurden drei mögliche Endstufen zur Verarbeitung unterschieden: das bloße Nullen der Spikes im K-Raum, die Interpolation der segmentierten Stellen und die Transformation durch das trainierte Netzwerk. Als Vergleichsgrundlage wurden Aufnahmen eines Phantoms mit und ohne Spikes erstellt. Die spike-belasteten Daten wurden anschließend durch die implementierte Pipeline bearbeitet. Dabei zeigte sich, dass alle implementierten Verarbeitungsmethoden die Musterung innerhalb der Bildräume effektiv entfernen konnten. Die resultierenden Bildräume zeigten keinen Unterschied in der Qualität. Es stell. Es zeigte sich, dass das Nullen der Spikes im K-Raum ausreichend war, um eine Verbesserung der Bildqualität zu erzielen. Im ersten direkten Vergleich mit einem regelbasierten Algorithmus zeigen die entwickelten Netzwerke auf dem verwendeten Datensatz bessere Ergebnisse als das regelbasierte Äquivalent.

9.2 Ausblick

Die bisherigen Ergebnisse wurden ausschließlich mit künstlich erzeugten Spikes an einer MRT-Anlage getestet. Um die Effektivität der Pipeline weiter zu evaluieren, wäre es sinnvoll, sie an einer Anlage zu testen, die aufgrund einer Hardware-Fehlfunktion oder äußeren Einflüssen (z. B. Fremdkörper im Scanner) Spikes erzeugt. Dadurch könnte überprüft werden, ob diese Spikes ebenfalls erkannt und entfernt werden können. Zusätzlich könnten verschiedene

Körperregionen innerhalb der MRT-Aufnahmen als Testobjekte vermessen werden, um die Generalisierungsfähigkeit des Netzwerks genauer zu bestimmen.

Um die Leistung des entwickelten KI Algorithmus besser einschätzen zu können, sollten auch andere fortschrittliche Spike-Detektionsalgorithmen mit dem entwickelten Algorithmus verglichen werden. Dazu wäre es ratsam, eine große Anzahl an heterogenen Datensätzen zu verwenden, um eine umfassendere Aussage treffen zu können.

Abbildungsverzeichnis

Formelverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Akiba19] Tajuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, Masanori Koyama1. *Optuna: A Next-generation Hyperparameter Optimization Framework.* arxiv, 2019. S1

[Campbell-Washburn16] Adrienne E. Campbell-Washburn, , David Atkinson,Zoltan Nagy,Rachel W. Chan,Oliver Josephs,Mark F. Lythgoe,Roger J. Ordidge,David L. Thomas. *Using the Robust Principal Component AnalysisAlgorithm to Remove RF Spike Artifacts from MR Images.* Wiley Online Library, 2016. S1-2

[Dössel16] Dössel, Olaf. *Bildgebende Verfahren in der Medizin.* Heidelberg: Springer, 2016.

[Douwe19] Osinga, Douwe. *Deep Learning Kochbuch.* Paderborn: O'Reilly, 2019.

[Dozat16] Dozat, Timothy. *Incorporating Nesterov moment into adam.* stanford: ICLR, 2016. S1

[Elster23] Elster, Allen D. *Questions and answers in MRI.* 2015. https://www.mriquestions.com/index.html (accessed 07 06, 2023).

[Glorot10] Glorot Xavier, Yoshua Bengio. *Understanding the difficulty of training deep feedforward neural networks.* 2010.

[Huang18] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger. *Densly connected convolutional Networks.* Cornell, 2018.

[Heaton15] Heaton, Jeff. *Artificial Intelligence for Humans Volume 3: Deep Learning and Neural Networks.* St. Louis: Heaton Research Inc., 2015.

[He15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Deep residual lLearning for image recognition.* 2015.

[Ioffe15] Sergey Ioffe, Chrisitan Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. S2-3

[Kingma15] Diederik P. Kingma, Jimmy Lei Ba. *Adam: A method of stochastic optimization.* Amsterdam: ICLR, 2015. S1-2

[Simonya15] Karen Simonyan, Andrew Zisserman. *Very deep convolutional networks for large scale image recognition.* Oxford: ICLR, 2015.

[MatlabOne23] MatlabOne. *MatlabOne.* 6 1, 2023. https://matlab1.com/max-pooling-in-convolutional-neural-network/.

[Pabst13] Pabst, Dr. med Christoph. *Magnetresonanz-Tomographie Lernskript für Mediziner S.5.* Universitätsklinikum Giesen Marburg , 2013.

[Phorkel23] Pohrkel, Sabina. *Towards Data Scienense.* 6 1, 2023. https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d.

[Radtke23] Radtke, Rainer. *Statista.* 7 12, 2022. https://de.statista.com/statistik/daten/studie/182664/umfrage/kernspintomographen-anzahl-in-europa/?_sm_vck=q0L5TRbMHfHHQWTbJfRbtDvDRtNPSrbvSvRqjSHjM4TTRRNfL4LR (accessed 6 14, 2023).

[Rosenblatt58] Rosenblatt, Frank. *THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION.* Cornell: Cornell Aeronautical Labratory, 1958.

[Ronneberger15] Olaf Ronneberger, Philipp Fischer, Thomax Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* Freiburg, 2015.

[Stefanos23] Tsoutis, Stefanos. *link niedersachen.* 5 30, 2023. https://www.link-niedersachsen.de/blog/blog_technik_wissenschaft/deep_learning.

[VanBeers19] Floris van Beers, Arvid Lindstrom, Emmanuel Okafor and Marco A. Wiering. *Deep Neural Networks with Intersection over Union Loss for Binary Image Segmentation* . Groningen, 2019. S4-5

[Wang04] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, Eero P. Simoncelli. *Image Quality Assesment: From Error Visibility zo Structural SImilarity.* IEEE, 2004.

[Wang18] Wang, Chi-Feng. *Towards Data science.* 1 8, 2018. https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484 (accessed 6 1, 2023).

[Wilmott20] Wilmott, Paul. *Grundkurs Mashine Learning.* Bonn : Rheinwerk, 2020.

[Xiangyu17] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun. *Shuffle net, an extremly efficient convolutional neural network for mobile devices.* 2017.

[Xiadong01] Xiaodong Zhang, Pierre-Francois Van De Moortel, Josef Pfeuffer, Xiaoping Hu. *Elimination of-kSpace spikes in fMRI data.* Elsevier, 2001.

[Yi-Hsuan00] Yi-Hsuan Kao, James R. MacFall. *Correction of MR k-Space Data Corrupted by Spike Noise.* IEEE, 2000. S1

[Zhou20] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, Jianming Liang. *UNet++, redisigning skip connections to exploit multiscale features in image segmentation.* 2020.