

TECHNISCHE UNIVERSITÄT DRESDEN
FAKULTÄT INFORMATIK
INSTITUT FÜR SOFTWARE- UND MULTIMEDIATECHNIK
PROFESSUR FÜR COMPUTERGRAPHIK UND VISUALISIERUNG
PROF. DR. STEFAN GUMHOLD

Großer Beleg

Parameterrekonstruktion für Röntgen-Kleinwinkelstreuung mit Deep Learning

Patrick Stiller
(Mat.-Nr.: 3951290)

Betreuer: Dr. Dmytro Schlesinger (TU Dresden), Dr. Heide Meißner (HZDR),
Dr. Michael Bussmann (HZDR)

Dresden, 8. März 2019

AUFGABENSTELLUNG FÜR DIE BELEGARBEIT (INF-D-950)

Name, Vorname des Studenten: Stiller, Patrick

Immatrikulationsnummer: 3951290

Studiengang: Diplom (2010)

Thema (deutsch): **Parameterrekonstruktion für Kleinwinkelstreuung mit Deep Learning**

Zielstellung:

Die Kleinwinkelstreuung (SAXS) ist eine universelle Technik zur Untersuchung von Feststoffen. Dabei liefert SAXS Informationen über die kristalline Struktur, chemische Komposition und die physikalischen Eigenschaften des untersuchten Feststoffes.

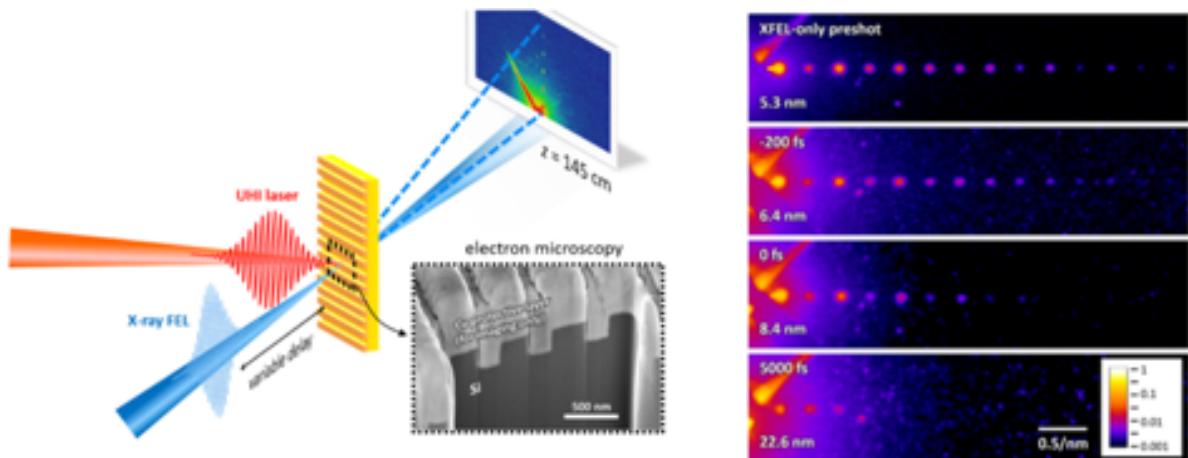


Abbildung 1: Schematischer Aufbau des Experimentaufbaus. Im rechten Teil der Abbildung sind Detektorbilder in Abhängigkeit der Bestrahlungsdauer des Hochintensitätslaser dargestellt.

Im vorliegenden Experiment wird SAXS (Abbildung 1) genutzt um die Elektrodynamik von Plasma zu untersuchen. Dazu wird ein gitterförmiges Target mit einem Hochintensitätslaser beschossen, um Plasma zu erzeugen. Kurz nach dem Einsetzen des Hochintensitätslasers kommt ein Röntgenlaser zum Einsatz, um mit Kleinwinkelstreuung die Struktur des entstandenen Plasmas zu untersuchen.

Bei der Kleinwinkelstreuung, werden die Photonen des Röntgenlasers an den Teilchen des Plasmas gestreut und wie bei einer klassischen Röntgenaufnahme detektiert. Dabei werden durch den Detektor die ankommenden Lichtintensitäten gemessen. Der Detektor ist zeitintegrierend, das heißt, dass ankommende Intensitäten aufaddiert werden. Somit gehen die Zeitunterschiede (Phase) der ankommenden Wellen verloren und somit auch die Positionsinformationen der Plasmateilchen. Funktional ist das Detektorbild äquivalent zum Fourier-Betragsquadrat des Targets.

Ziel ist es nun die Positionsinformationen der Plasmateilchen wiederherzustellen. Um diese Aufgabe zu vereinfachen, wurde das Target als Gitter strukturiert, welches vollständig durch drei Parameter (Abbildung 2) beschrieben werden kann. Dabei beschreibt Pitch die Wellenlänge des Gitters, Feature Size die Länge einer Erhöhung und Sigma den Aufweichungseffekt, welcher durch den Hochintensitätslaser erzeugt wird. Nun soll mit Hilfe von Machine Learning die Gitterstruktur (beschrieben durch seine drei Parameter) anhand des

Detektorbildes rekonstruiert werden. Dafür ist ein Deep Convolutional Neuronal Network vorgesehen. Aufgrund mangelnder Experimentdaten soll eine Simulation als Datenbasis eingesetzt werden. Die vorliegende Simulation simuliert den Idealfall des Experiments und blendet somit ungewollte Randeffekte aus. Wenn die Datenbasis erstellt ist, sollen die synthetisierten Daten dazu genutzt werden um das Deep Convolutional Neuronal Network zu trainieren. Schlussendlich soll die Qualität des Deep Convolutional Neuronal Networks durch ein aussagekräftiges Qualitätsmaß bestimmt werden.

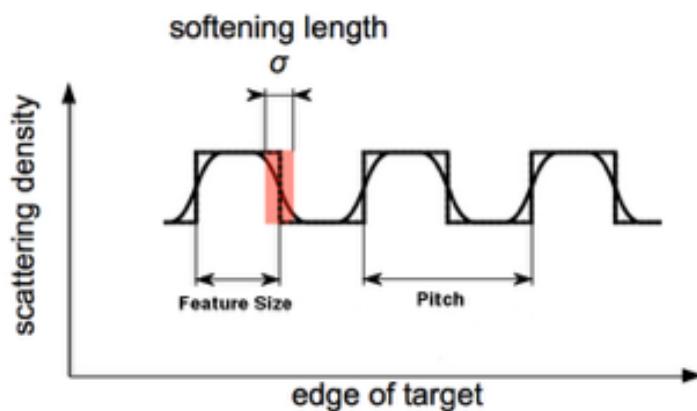


Abbildung 2: Beschreibung der Gitterstruktur durch die drei Parameter : Sigma, Feature Size und Pitch

In der Arbeit sollen schwerpunktmäßig folgende Teilspekte bearbeitet werden:

- Literaturrecherche, insbesondere Einarbeitung in die Thematiken: Fourier Transformation und Tiefe Neuronale Netze.
- Aufgabenanalyse, Entwicklung einer geeigneten Netzarchitektur.
- Trainieren des Netzes auf synthetischen Daten, Evaluierung der Ergebnisse.
- Optional: Untersuchung der Anwendbarkeit des trainierten Netzes für reale Daten.

Betreuer:

Dr. Dmytro Shlezinger, Dr. Heide Meißner (HZDR),
Dr. Michael Bussmann(HZDR)

Verantwortlicher Hochschullehrer:

Dr. Dmytro Shlezinger

Institut:

Software und Multimediatechnik

Beginn am:

22.10.2018

Einzureichen am:

11.03.2019

.....
Datum, Unterschrift der/des Studierenden

.....
Unterschrift des betreuenden Hochschullehrers

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tag dem Prüfungsausschuss der Fakultät Informatik eingereichte Arbeit zum Thema:

Parameterrekonstruktion für Röntgen-Kleinwinkelstreuung mit Deep Learning

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Dresden, den 8. März 2019

Patrick Stiller

Inhaltsverzeichnis

1 Einleitung	3
2 Grundlagen	5
2.1 Physikalischer Hintergrund	5
2.1.1 Kleinwinkelstreuung	5
2.1.2 Experimentbeschreibung	5
2.2 Problemidentifikation	7
2.3 Fouriertransformation	8
2.4 Filter	8
2.5 Neural Networks	8
2.5.1 Aufbau	9
2.5.2 Feed-Forward Neural Networks	9
2.5.3 Lernprozess	10
2.5.4 Convolutional Neural Networks	11
2.5.5 Residual Strukturen	12
3 Simulation	13
3.1 Motivation	13
3.2 Simulationsbeschreibung	13
3.2.1 Design der Gitterstruktur und Simulation des Hochenergielasereinflusses	13
3.2.2 Effekt der Startparameter auf die Gitterstruktur	14
3.2.3 Simulation der Streuung	15
3.2.4 Auswirkung der Startparameter auf das Simulationsergebnis	16
3.2.4.1 Mathematische Erklärung der Starparametereffekte	20
4 Lösungsansatz	23
4.1 Gesamtarchitektur	23
4.2 Convolutional Neural Network	24
4.2.1 Bayes'sche Optimierung	24
4.2.2 Architektur	24
4.3 Fully-Connected Neural Network	25
4.4 Training	25
5 Datenbasis	27
5.1 Generator	27
5.2 Wahl der Eingangsparameter	27
5.3 Performance	28
5.4 Trainings-, Validierungs- und Testdatensatz	28
6 Ergebnisse und Diskussion	31
6.1 Trainingsprozess	31
6.1.1 Convolutional Neural Network	31
6.1.2 Fully-Connected Neural Network	31
6.2 Evaluierung	33
6.2.1 Convolutional Neural Network	33

6.2.2	Fully-Connected Neural Network	35
6.3	Fazit	37
6.4	Ausblick	38
Literaturverzeichnis		39

1 Einleitung

Die Erforschung von laser-induzierten Plasmazuständen ist einer der Forschungsschwerpunkte des Helmholtz-Zentrums Dresden – Rossendorf. Im Jahr 2018 wurde von Thomas Kluge ein Paper mit dem Titel: „Observation of ultrafast solid-density plasma dynamics using femtosecond X-ray pulses from a free-electron laser“ [Klu18] veröffentlicht, welches die Charakterisierung von Plasma anhand des Streubildes der Kleinwinkelstreuung beschreibt. Das Experiment, welches in [Klu18] beschrieben wurde, nutzt ein Gitterförmiges Target, welches durch wenige Parameter beschreibbar ist. Um Plasma zu erzeugen, wird das Target mit einem Hochintensitätslaser beschossen. Für die Charakterisierung des entstandenen Plasmas wird zusätzlich ein Röntgenlaser verwendet. Während des Beschusses durch den Röntgenlaser kommt es zu Streueffekten, welche durch einen Detektor, der hinter dem Target platziert ist, aufgenommen werden. Dieser Detektor ist zeit integrierend, das heißt, dass ankommende Wellen summiert werden und somit die Phaseninformationen der Photonen verloren gehen.

Um die Struktur des Targets (Beschreibungsparameter) zu rekonstruieren, werden im Allgemeinen Phase-Retrieval-Algorithmen angewendet. Diese Algorithmen sind meistens iterativ und unterliegen Limitierungen. In diesem Beleg soll, anstatt iterativer Verfahren, Deep-Learning verwendet werden, um die Charakterisierung des Plasmas zu unterstützen. „Deep-Learning erlaubt mathematischen Modellen, welche aus mehreren Verarbeitungsschichten bestehen, die Repräsentation von Daten auf verschiedenen Abstraktionsniveaus zu lernen. Deep Learning hat den Stand der Technik in Spracherkennung, Objekterkennung und in vielen anderen Bereichen wie der Genetik stark verbessert. [...] Deep Convolutional Neuronal Networks gelang der Durchbruch in der Verarbeitung von Text, Bild und Video.“ ([LBH15], Seite 1, Übersetzung des Autors).

In diesem Beleg wurde eine umfangreiche Datenanalyse der Detektorbilder vorgenommen, um aus deren Erkenntnissen einen Prototyp eines Deep Learning Models zu entwickeln. Um das Deep-Learning Model zu trainieren, wurde mit Hilfe einer Simulation eine Datenbasis aus synthetisch erzeugten Detektorbildern erstellt. Das trainierte Deep-Learning-Model wurde anschließend auf seine Qualität getestet und auf seine Tauglichkeit als Ersatz für die derzeit genutzte Vorwärtssimulation geprüft.

2 Grundlagen

2.1 Physikalischer Hintergrund

2.1.1 Kleinwinkelstreuung

Die Kleinwinkelstreuung, in Englisch Small Angle X-Ray Scattering (SAXS), ist eine universelle Technik zur Untersuchung von Feststoffen. Dabei kann SAXS Informationen über die kristalline Struktur, chemische Komposition und die physikalischen Eigenschaften eines untersuchten Feststoffes liefern [Cro09]. Die Untersuchung von Feststoffen unter Einfluss von Hochintensitätslasern ist ein Schwerpunkt der heutigen Physik. Wissen über das Verhalten von Feststoffen unter Einfluss von Hochintensitätslasern kann offene Fragen in der Krebsforschung und in der Astrophysik beantworten. Dieser große Beleg bezieht sich auf die Publikation von Thomas Kluge, welche 2018 mit dem Titel: “Observation of ultrafast solid-density plasma dynamics using femtosecond X-ray pulses from a free-electron laser” veröffentlicht wurde. In der angesprochenen Publikation wurde SAXS für die Untersuchung von Plasma eingesetzt [Klu18]. Bisher konnte die Plasmadynamik, welche bei der Interaktion eines Feststoffes und eines Hochintensitätslasers entsteht, nur durch Simulationen modelliert werden. Mit Hilfe von SAXS gibt es neue Möglichkeiten, das entstehende Plasma zu charakterisieren. SAXS erreicht außerdem eine räumliche Auflösung im Femtosekunden-Bereich und eine zeitliche Auflösung im Nanosekunden-Bereich. Mit SAXS ist es möglich anhand der Einstrahlung auf den Detektor mit Hilfe von SAXS-Vorwärtssimulationen Aussagen über die Elektronendynamik im Plasma zu treffen [Klu18].

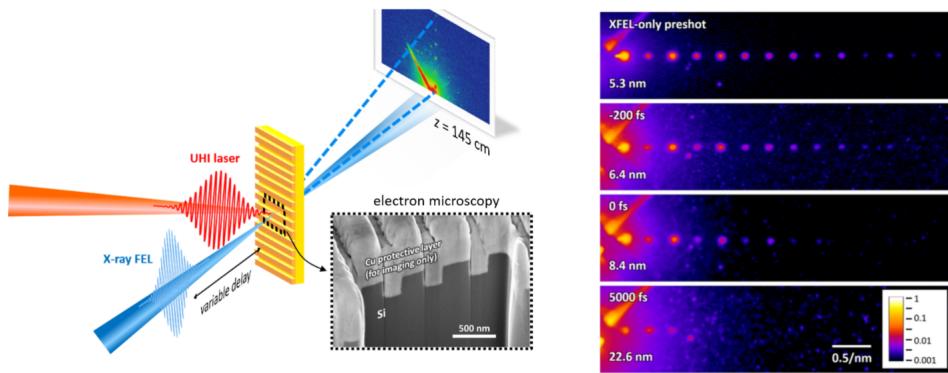


Abbildung 2.1: Links: Schematischer Aufbau des Experimentaufbaus. Rechts: der Abbildung sind Detektorbilder in Abhängigkeit der Bestrahlungsdauer des Hochintensitätslasers dargestellt. Bildquelle: [Klu18]

2.1.2 Experimentbeschreibung

Das in [Klu18] beschriebene Experiment besteht aus vier Hauptkomponenten (siehe Abbildung 2.1 links): dem Target (Mitte), dem Hochintensitätslasern (UHI) (Links), dem Röntgen-Freie-Elektronen-Laser (XFEL) (links unten) und dem Detektor (rechts oben). Der Hauptbestandteil des Targets ist Kupfer. Um eine analytische Beschreibung des Experimentausgangs zu ermöglichen, wurde eine Gitterstruktur in das Kupfertarget eingraviert. Zusätzlich wurde das Taget mit einer $2 \mu\text{m}$ breiten Siliziumschicht

überzogen. Durch die eingravierte Gitterstruktur besitzt das Target einen eindimensionalen Informationsgehalt und kann somit durch drei Parameter vollständig beschrieben werden. Die drei Parameter sind Pitch, Feature-Size und die Aufweichungsbreite Sigma(σ). Dabei beschreiben Pitch und Feature-Size die Struktur des Targets und Sigma(σ) den Aufweichungseffekt des Hochintensitätslasers (siehe Abbildung 2.2). Die Gesamtheit des Targets wird auch als Grating bezeichnet. Erhebungen des Gratings werden als Features bezeichnet. Im weiteren Verlauf dieser Arbeit wird der Begriff Streudichte η verwendet, welcher oft im Zusammenhang des SAXS-Ansatzes fällt. Der Begriff Streudichte η wird in dieser Arbeit als Synonym für das Gitter verwendet.

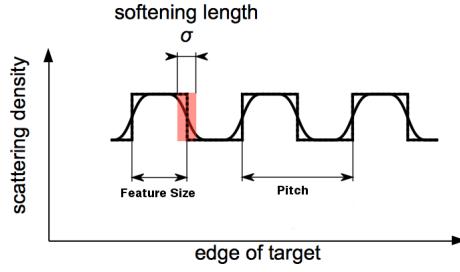


Abbildung 2.2: Analytische Beschreibung des Querschnittes des Targets. Dabei ist das Grating zu erkennen und dass es durch drei Parameter: Pitch, Feature-Size und Sigma beschrieben werden kann. Bildquelle: [Zac17]

Während des Experiments wird das Target durch einen Hochintensitätslaser in einem Winkel von 90° beschossen. Aufgrund des elektrischen Feldes des Lasers entsteht Plasma. Bei Plasma handelt es sich um ein Gemisch aus freien Elektronen, positiven Ionen und neutralen Teilchen, welche unter ständiger Wechselwirkung miteinander und mit Photonen stehen. Dadurch kann es zu unterschiedlichen Energie- bzw. Anregungszuständen kommen. Der Plasmazustand eines Stoffes wird auch als vierter Aggregatzustand bezeichnet [Wis18]. Während des Beschusses durch den Hochintensitätslaser ist außerdem ein Schmelzprozess und somit eine Aufweichung der Gitterstruktur des Targets wahrzunehmen (siehe Abbildung 2.3). Das durch den Hochintensitätslaser erzeugte Plasma soll auf seine Struktur und Elektrodynamik mit Hilfe von SAXS untersucht werden. Dafür wird leicht zeitversetzt ein zweiter Laser benutzt. Dabei handelt es sich um einen Röntgen-Freie-Elektronen-Laser, welcher in einem Winkel von 45° mit einer Pulsdauer von 40 Femtosekunden auf das Target schießt. Bei diesem Vorgang kommt es zu einer Streuung des Lichts des Röntgenlasers an den Elektronen des Targets. Die Lichtintensitäten des gestreuten Röntgenlasers werden durch einen Detektor, welches hinter dem Target platziert ist, gemessen (Abbildung 2.1 links).

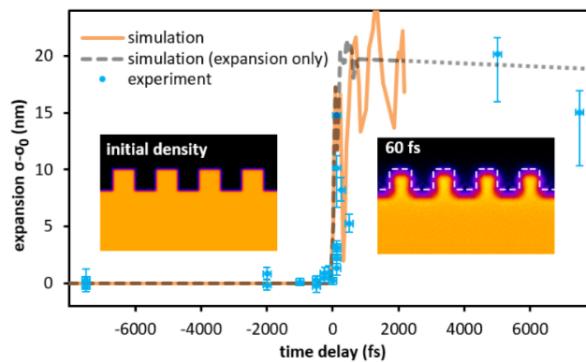


Abbildung 2.3: Veränderung des Gratings nach einer Bestrahlungsdauer von 60 Femtosekunden Bildquelle: [Klu18]

Beim Detektor handelt es sich um ein Raster von Lichtdetektoren, welche die ankommenden Lichtintensitäten messen. Dabei integriert der Detektor über die Zeit, das heißt, dass ankommende Lichtintensitäten über die Zeit summiert werden. Dieser Effekt ist in der Abbildung 2.4 zu erkennen, welche einen beispielhaften Streuprozess an zwei Elektronen zeigt.

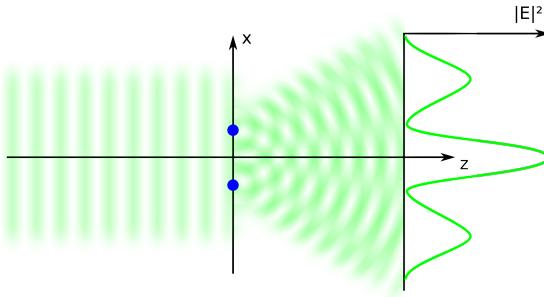


Abbildung 2.4: Beispielhafte Darstellung einer Streuung an zwei Elektronen. Die Röntgenlaserpulse treffen geradlinig auf die Elektronen und werden dann bei den Elektronen gestreut. Der Detektor dahinter misst die ankommenden Intensitäten der kreisförmigen Wellen und summiert diese über die Zeit. Bildquelle: [Zac17]

Durch die Zeitintegration des Detektors gehen die zeitlichen Abstände der Wellen verloren (Phase). Das entstandene Detektorsignal skaliert mit dem Betragsquadrat der Fouriertransformation der Gitterstruktur η (siehe Gleichung (2.1)) [Klu18].

$$\Phi \propto \left| \int \eta(\vec{r}) \cdot e^{i\vec{q}\vec{r}} d\vec{r} \right|^2 \quad (2.1)$$

2.2 Problemidentifikation

Eine der Herausforderungen des beschriebenen Experimentes ist die Zeitintegration des Detektors. Diese Detektoreigenschaft ruft den Verlust der Phase hervor, sodass eine Rekonstruktion der Elektronendichtheverteilung η mit Hilfe einer inversen Fouriertransformation nicht möglich ist. Um dieses Problem zu lösen, werden iterative Algorithmen, sogenannte Phaseretrieval-Algorithmen verwendet. Beispiele für solche Algorithmen sind [Fie82]: Error-Reduction Algorithm, Gradient Search Methods und der Input-Output Algorithmus. Probleme bei diesen Algorithmen sind, dass erstens bei allen Algorithmen keine Konvergenz zum richtigem Ergebnis nicht garantiert ist und zweitens die Anzahl der Iterationen, die notwendig sind, um eine Optimierung der errechneten Phase zu erzeugen nicht bekannt ist. Diese Eigenschaften führen dazu, dass eine hochfrequente Verarbeitung von Experimentergebnissen verwehrt bleibt. Deswegen soll mit Hilfe von Deep Learning die Parameter zur Beschreibung der Gitterstruktur rekonstruiert werden. Neural Networks haben den Vorteil, dass sie ihre Schätzung nicht iterativ bestimmen (Einschrittverfahren)

2.3 Fouriertransformation

Die Fouriertransformation (benannt nach Jean Baptiste Joseph Fourier) ist eine Transformation, welche zeitbezogene Wellen im Ortsraum in ihre frequenzmäßigen Spektralanteile zerlegt. Die Fouriertransformation wird auch als Transformation vom Orts- in den Frequenzraum bezeichnet. Die Fouriertransformation $F(u)$ einer Welle $f(x)$ ist folgendermaßen definiert:

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i x u} dx \quad (2.2)$$

Die Spektralanteile $F(u)$ werden wie bei einem Basiswechsel durch ein Integral bestimmt. Dabei ist das Ergebnis des Integrals die Länge der Projektion in Richtung der durch u kodierten Frequenz. Im endlich-diskreten Fall wird das Integral durch eine Summe bis zur Anzahl der zu verwendeten Wellen ersetzt. Jede Stelle u der Fouriertransformation kodiert eine Welle $e^{-2\pi i x u}$, welche über die Euler-Identität $e^{ikx} = \cos(kx) + i \cdot \sin(kx)$ auch durch ihre Kosinus- und Sinusanteile beschrieben werden kann. Der Funktionswert der Fouriertransformation $F(u)$ kodiert mit welchem Anteil die durch u kodierte Frequenz verwendet wird. Das Ergebnis der Fouriertransformation kann auch durch das Amplituden- und durch das Phasenspektrum beschrieben werden. Dabei bestimmt die Amplitude das Maximum und das Minimum der jeweiligen Welle und die Phase die Verschiebung der jeweiligen Welle. Um die Amplitude $|F(u)|$ und die Phaseninformation $\phi(u)$ einer komplexen Zahl u zu errechnen, werden folgende Gleichungen verwendet [Sch]:

$$|F(u)| = \sqrt{R^2(u) + I^2(u)} \quad (2.3)$$

$$\phi(u) = \tan^{-1} \frac{I(u)}{R(u)} \quad (2.4)$$

2.4 Filter

Ein Filter ist eine Operation, welche auf einem Signal verwendet wird, um Signale zu glätten, Signalstörungen zu vermeiden, oder um Rauschen zu verhindern. In den meisten Fällen wird die Filteroperation mit Hilfe von Faltung realisiert. Die Faltung zweier Funktionen ($f * g$) wird durch den funktionalen Zusammenhang in Formel (2.5) beschrieben. Dabei ist f die Funktion, welche das Signal beschreibt und g die Funktion, welche den Filter beschreibt. Im diskreten Fall wird das Integral durch eine Summe bis zur entsprechenden Filtergröße ersetzt [Kla13].

$$(f * g)(x) := \int_{\mathbb{R}^n} f(\tau)g(x - \tau)d\tau \quad (2.5)$$

2.5 Neural Networks

Neural Networks sind eine mathematische Adaption des realen menschlichen Gehirns. Wie im realen menschlichen Gehirn sind Neuronen miteinander verbunden, um einen Informationsfluss zu gewährleisten. Die ersten Ansätze für Neural Networks wurden bereits 1943 von Warren McCulloch und Walter Pitts entwickelt [Kri07]. Heute sind Neural Networks der Schwerpunkt des Machine Learnings und werden auf großen Datenmengen angewendet, um ein gewünschtes Verhalten zu trainieren.

2.5.1 Aufbau

Ein Neural Network besteht aus vielen kleinen Komponenten, Neuronen, welche durch gerichtete und gewichtete Verbindungen verbunden sind. Sämtliche Beschreibungen und Notationen beziehen sich auf [Kri07]. Mathematisch definiert ist ein Neural Network als ein Tripel (N, V, w) mit den Beiden Mengen N und V und der Funktion w . N ist die Menge aller Neuronen und $V = \{(i, j) | i, j \in \mathbb{N}\}$ die Menge der Verbindungen zwischen Neuron i und Neuron j. Die Funktion $w : V \rightarrow \mathbb{R}$ beschreibt die Gewichte des Neural Networks, wobei $w(i, j)$ das Gewicht zwischen dem Neuron i und Neuron j beschreibt. Im Allgemeinen wird anstatt der Funktionsnotation die Notation $w_{i,j}$ für die Gewichte zwischen zwei Neuronen verwendet. Die nächste wichtige Komponente ist die Propagierungsfunktion net_j eines Neurons, welche einen wichtigen Teil des Informationsflusses in einem Neural Networks definiert. Dabei wird der Input des Neurons j durch dessen Propagierungsfunktion net_j bestimmt. Die Propagierungsfunktion net_j nimmt den Output aller Neuronen, welche eine ausgehende Verbindung zum Neuron j besitzen als Input. So wird die Propagierungsfunktion net_j durch folgenden funktionalen Zusammenhang beschrieben :

$$net_j = \sum_{i \in I_j} (o_i \cdot w_{i,j}) \text{ mit } I_j = \{ i \in N \mid (i, j) \in V \} \quad (2.6)$$

Der Output o_j eines Neurons j wird mit Hilfe der Aktivierungsfunktion a_j und der Propagierungsfunktion net_j berechnet. Dazu wird noch der Schwellwert θ_j zur Hemmung des Aktivierungszustandes des Neurons zur Hilfe genommen. Somit ergibt sich für den Output des Neurons folgender funktionaler Zusammenhang:

$$o_j = a_j(net_j - \theta_j) = f(x) \quad (2.7)$$

In den meisten Fällen werden differenzierbare Aktivierungsfunktionen benutzt, da sie den Lernprozess des Neural Networks erleichtern. Für diesen Beleg ist die Rectified Linear Unit- Aktivierungsfunktion (Formel (2.8)) relevant.

$$f(x) = \max(0, x) \quad (2.8)$$

Die ReLu-Aktivierungsfunktion ist im Vergleich zu anderen Aktivierungsfunktionen schneller zu berechnen und bietet größere Gradienten. Jedoch können Neuronen, welche einen negativen Input bekommen nur noch 0 Ausgeben. Es wird in diesem Zusammenhang von einem gestorbenen Neuron gesprochen. Ein weiterer Nachteil der ReLu-Aktivierungsfunktion ist der Wertebereich, denn dieser ist nach oben nicht eingeschränkt. Somit können in einem Neuronalen Netz sehr große Werte entstehen, welche den Wertebereich von Zahlenstandards wie zum Beispiel IEEE 754 (Single Precision Float 32-Bit) [Kah97] überschreiten, womit kein valider Datenfluss im Neural Network gegeben ist.

2.5.2 Feed-Forward Neural Networks

Für ein Neural Network können verschiedene Netzwerktopologien verwendet werden. Ein Beispiel dafür sind Feed-Forward Neural Networks (Abbildung 2.5). Bei einem Feed-Forward Neural Network werden die Neuronen als Schichten (Layer) angeordnet. Diese Layer sind miteinander verbunden. Die erste Layer, welche die Input Daten bekommt, wird als Input-Layer bezeichnet. Die letzte Layer, welche die Netzberechnung ausgibt, wird als Output-Layer bezeichnet. Schichten, welche sich zwischen Input- und Output-Layer befinden und somit keinen Kontakt nach Außen haben, werden als Hidden Layer bezeichnet. Ein wichtiges Merkmal von Feed-Forward Neural Networks ist, dass der Datenfluss geradlinig vom Input-Layer über die Hidden-Layer zum Output-Layer ohne Rückkopplung verläuft. Eine wichtige

Komponente von Feed-Forward Neural Networks sind die Fully-Connected Layer. Bei einem Fully-Connected Layer ist jedes Neuron des Fully-Connected Layer mit jedem Neuron des vorherigen Layers verbunden. In Abbildung 2.5 sind die Verbindungen eines Fully-Connected Layer dargestellt.

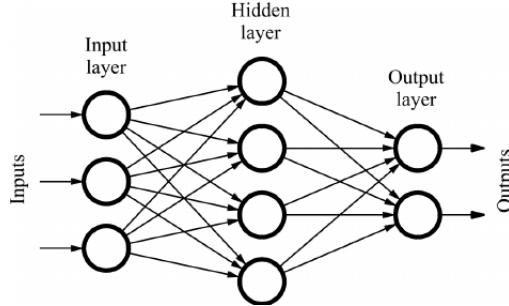


Abbildung 2.5: Beispielhafte Darstellung eines Feed-Forward Neural Networks. Bildquelle: [QR11]

2.5.3 Lernprozess

Das Anlernen von Neural Networks wird in den meisten Fällen durch überwachtes Lernen (supervised Learning) realisiert. Im Speziellen wird der Lernprozess durch den Backpropagation-Algorithmus und Gradientenabstiegsverfahren durchgeführt. Bei einem überwachten Lernansatz besteht der Datensatz zum Trainieren des Neural Networks aus zwei Teilen. Zu jedem Netzinput x_i gibt es ein zugehöriges Label y_i , welches den gewünschten Netzoutput definiert. Der Lernprozess eines Neural Networks kann in drei Phasen eingeteilt werden: dem Forward-Pass, die Loss-Calculation und dem Backward Pass [Bec]. Zu Beginn des Trainingsprozesses werden die Gewichte des Neuronalen Netzes mit Zufallszahlen initialisiert. In vielen Implementationen werden die Gewichte zufällig und normal verteilt initialisiert. Beim Forward Pass werden die Input-Daten zur Kalkulation des derzeitigen Netz-Outputs zum Input-Layer des Neural Network gegeben. Das Neural Network bestimmt dann durch die Rechenvorschriften des Neural Network den Netz-Output \hat{y}_i . Im zweiten Schritt wird die Qualität des Net-Outputs \hat{y}_i bestimmt. Dazu wird ein Fehlermaß benutzt, das den Grad des Unterschiedes zwischen \hat{y}_i und y_i bestimmt. Eine beispielhafte Fehlerfunktion ist Mean-Squared Error (2.9).

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (2.9)$$

Im dritten Schritt, dem Backward-Pass, kommt es zur Optimierung des Neuronalen Netzes. Mithilfe des Fehlers, welcher bis zur Eingabeschicht zurück propagiert wird, werden die Gewichte entsprechend ihres Einflusses auf den Net-Output (2.11) mittels Gradientenabstiegsverfahrens angepasst. Für die Anwendung des Gradientenabstiegsverfahrens werden die partiellen Ableitungen des Fehlerterms benötigt. Die Gewichtsveränderung Δw_{ij} des Gewichts zwischen Neuron i und Neuron j ergibt sich durch folgenden funktionalen Zusammenhang :

$$\Delta w_{ij} = -\lambda \frac{\partial E}{\partial w_{ij}} = -\lambda \delta_j o_i \quad (2.10)$$

Dabei ist E die Errorfunktion, δ_j dem Gradient des Neurons j , o_i die Ausgabe des Neurons i . Der Parameter λ , welcher die Learning-Rate des Gradientenabstiegsverfahrens bestimmt. Schlussendlich fehlt die Definition des Gradienten. Dieser ist davon abhängig, wie stark ein Neuron den Output des Neuronalen Netzes beeinträchtigt. Deswegen wird eine Unterscheidung getroffen, ob ein Neuron sich im Output-Layer oder dem Hidden- bzw. Input-Layer befindet. In der folgenden Gleichung (2.11) ist die Definition des Gradienten und die zugehörige Propagierung des Fehlers definiert.

$$\delta_j = \begin{cases} a_j(\text{net}_j)(o_j - t_j) & \text{falls } j \text{ sich in der Output-Layer befindet} \\ a_j(\text{net}_j) \sum_k \delta_k w_{jk} & \text{falls } j \text{ verdecktes Neuron oder ein Input-Neuron ist} \end{cases} \quad (2.11)$$

Dabei ist die Gleichung (2.11) in Abhängigkeit von den Variablen o_j , die Soll-Ausgabe t_j des Neurons j und der Aktivierungsfunktion a_j des Neurons j angegeben. Somit ergibt sich das neue Gewicht durch eine Addition des alten Gewichts mit der errechneten Gewichtsveränderung.

$$w_{ij}^{\text{neu}} = w_{ij}^{\text{alt}} + \Delta w_{ij} \quad (2.12)$$

Der Backpropagation-Algorithmus wird iterativ solange angewandt, bis eine bestimmte Anzahl von Iterationen erreicht ist oder andere Kriterien erfüllt sind [Wik18].

2.5.4 Convolutional Neural Networks

Klassische Neural Networks besitzen die Einschränkung, dass deren Inputs als Vektoren geliefert werden müssen. Soll ein klassisches Neural Network zum Beispiel ein Bild der Größe $N \times M \times C$ (C steht für Kanäle) verarbeiten, muss das Bild in einen Vektor der Größe $N * M * C$ transformiert werden (Flatten). Convolutional Neural Networks (CNNs) besitzen im Gegensatz zu klassischen Neural Networks Convolutional Layer, und können somit Inputs höherer Dimension verarbeiten. Dazu sind die Neuronen als Filter k der Größe $H \times W \times D$ (Höhe Breite, Tiefe) angeordnet. In diesem Kapitel wird von einem Filter mit quadratischer Grundfläche ausgegangen ($H = W$). Zur Berechnung des jeweiligen Layer-Outputs führt ein Filter k an einer Position (x, y) des Inputs I eine zweidimensionale diskrete Faltung (2.13) [Tho18] durch und berechnet somit einen Datenpunkt der Feature-Map I^* (Ergebnis eines Filters des Convolutional Layers). Der Parameter a in der Faltungsformel (2.13) steht für die Koordinate des Mittelpunktes des Filters. Ist der Filter die Grundfläche des Filters zum Beispiel 5×5 , so ist der Wert von a drei [ON15].

$$I^*(x, y) = \sum_{i=1}^H \sum_{j=1}^W I(x - i + a, y - j + a) k(i, j) \quad (2.13)$$

Zur Berechnung des nächsten Datenpunktes der Feature Map I^* wird der Filter um eine Schrittweite (*stride*) auf dem Input verschoben. Sollten für die Berechnung von $I^*(x, y)$ Datenpunkte benötigt werden, welche über den Rand des Inputs hinaus liegen, müssen diese Punkte auf eine andere Weise bestimmt werden. Eine mögliche Strategie ist das vernachlässigen dieser Punkte, was je nach Größe des Filters eine Verkleinerung der Feature-Map I^* zur Folge hätte. Die zweite mögliche Strategie, welche die fehlenden Datenpunkte zur Verfügung stellt, ist das Zero-Padding, welches die fehlenden Datenpunkte mit dem Wert Null ersetzt. Außerdem besteht die dritte Möglichkeit, dass die fehlenden Datenpunkte mit dem dazugehörigen Randwert ersetzt werden [Des].

Nach Ausführung der Faltung wird die errechnete Feature-Map durch eine ausgewählte Aktivierungsfunktion verarbeitet. Ein Convolutional-Layer besteht nicht nur aus einem Filter, sondern kann mehrere Filter enthalten. Für jeden Filter wird die vorher beschriebene Faltung separat ausgeführt und die entstandenen Feature-Maps werden konkateniert. Somit ergibt sich für den Output eines Convolutional-Layer mit einer Stride von 1, Zero-Padding und 32 Filtern eine Dimension von $(N, M, 32)$. In einem Convolutional Neural Network können dann beliebige viele Layer hintereinander platziert werden. Zur Verarbeitung des letzten Convolutional-Layers wird in den meisten Fällen die letzte Layer in einen Vektor transformiert, um durch ein Fully-Connected Layer weiterverarbeitet werden zu können (siehe Abbildung 2.6).

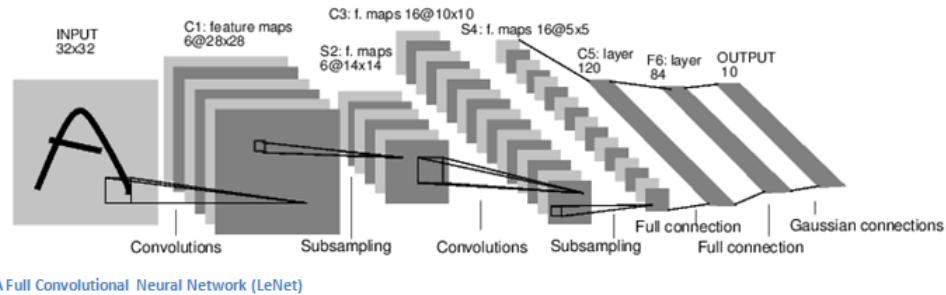


Abbildung 2.6: Beispielhafte Darstellung eines Convolutional Neural Networks. Bildquelle: [Des]

Für diesen Beleg sind aufgrund der Eindimensionalität der Input-Daten 1D-Convolutional-Layer relevant. Bei einem 1D-Convolutional Layer sind die Filter eindimensional, das heißt, dass die Filter eine Höhe von 1 besitzen und der Filter nur entlang einer Dimension verschoben wird..

2.5.5 Residual Strukturen

Wie in Kapitel 2.5.4 beschrieben, können beliebig viele Convolutional-Layers hintereinander platziert werden. Je mehr Layer verwendet werden, desto tiefer wird das Netz. Mit steigender Tiefe des Netzes erhöht sich das Problem kleiner werdender Gradienten (Vanishing-Gradient Problem). Das Vanishing Gradient Problem ist auf die Eigenschaft der Backpropagation zurückzuführen, dass die Gradienten Neuronen-Gewichte in Abhängigkeit zu deren Einfluss auf den Net-Output stehen. Bei tieferen Netzstrukturen verringert sich dieser Einfluss. Um den Einfluss der Neuronen auf den Net-Output zu erhöhen, werden Skip-Connections, auch Residual Strukturen genannt, benutzt. Dazu wird der Output eines Layers auf den Input z.B. der übernächsten Schicht addiert. Es sind auch Skip-Connections mit einer höheren Schrittweite möglich [HZRS15].

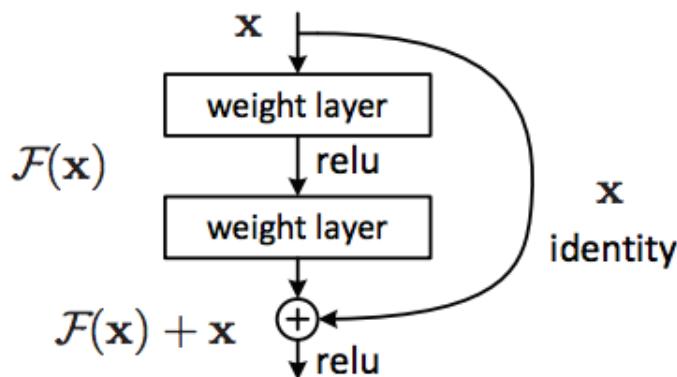


Abbildung 2.7: Realisierung einer Skip-Connection in einem Neural Network. Bildquelle: [HZRS15]

3 Simulation

3.1 Motivation

Um das Ergebnis des SAXS-Experiments bestmöglich zu verstehen und aufgrund einer geringen Anzahl von Experimentdaten, wurde im Rahmen der Master-Arbeit von Malte Zacharias mit dem Titel “Model-Driven Parameter Reconstruction from Small Angle X-Ray Scattering Images“ eine Modellierung entwickelt. Die Modellierung beinhaltet das Design der Gitterstruktur (Elektronendichteverteilung), die Modellierung des Hochenergielasereinflusses, die Modellierung des Streuvorgangs und die Modellierung des Detektorbildes. Die durch Malte Zacharias erstellte Simulation wurde für diese Arbeit zur Generierung der Datenbasis genutzt.

3.2 Simulationsbeschreibung

3.2.1 Design der Gitterstruktur und Simulation des Hochenergielasereinflusses

Wie in Kapitel 2 beschrieben sind die Targets des SAXS-Experiments als Gitter strukturiert, um eine analytische Beschreibung des Experiments zu ermöglichen. Der erste Simulationsschritt ist die Bestimmung der Gitterstruktur. Die Breite des Gratings ist auf N Pixel beschränkt. Die Struktur des Gratings ist über die drei Startparameter Pitch, Feature-Size und Sigma definiert. Der Parameter Feature-Size legt die Breite eines Features fest und der Parameter Pitch bestimmt die Periodizität eines Features, womit beide Parameter für die Struktur des Gratings ohne Einfluss des Hochintensitätslasers verantwortlich sind. Der Parameter Sigma σ bestimmt die Aufweichungsbreite des Gratings und modelliert den Einfluss des Hohenenintensitätslasers.

Zur Modellierung eines Features mit Hochintensitätslasereinfluss. Dazu wird eine Rechteckfunktion ($1_{[0, \text{Feature-Size}]}(x)$), welche die Breite eines Features festlegt mit einer Gaussverteilung ($\exp(-x^2/2\sigma^2)$) gefaltet [Klu18]. Das Ergebnis der Faltung wird mit Hilfe der Errorfunktion (Gleichung (3.1)) dargestellt.

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\xi^2} d\xi \quad (3.1)$$

$$\tilde{\eta} = \frac{\sqrt{\pi}\sigma}{2} (\text{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) - \text{erf}\left(\frac{x - \text{fsize}}{\sqrt{2}\sigma}\right)) \quad (3.2)$$

Das modellierte Feature wird durch eine weitere Faltung mit mehreren um Pitch verschobenen Impulsen periodisch fortgesetzt wird, um die Grating-Struktur zu komplettieren.

3.2.2 Effekt der Startparameter auf die Gitterstruktur

Die drei Startparameter Sigma (σ), Pitch und Feature-Size, welche die Struktur des Gratings beschreiben, haben unterschiedliche Effekte auf die Gratingstruktur. Sigma ist der Parameter, welcher, wie bereits beschrieben, den Einfluss des Hochenergielasers auf ein Feature modelliert. Je höher Sigma gewählt wird, desto mehr wird die Kantenstruktur des Targets aufgeweicht. In Abbildung 3.1 ist dieser Effekt dargestellt.

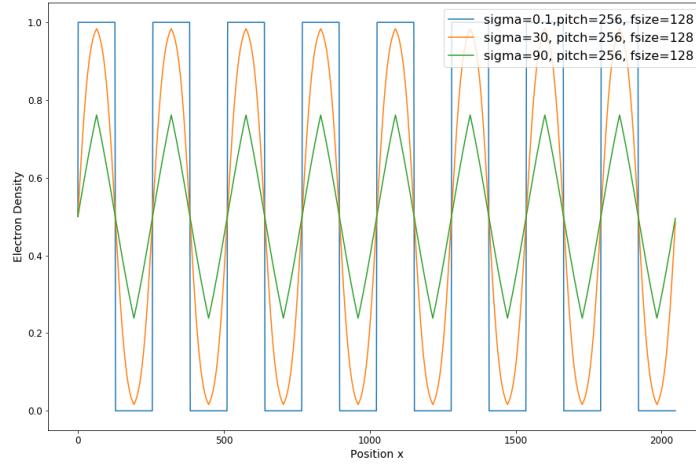


Abbildung 3.1: Vergleich der Kantenstruktur unter variierenden Sigma-Werten bei gleichbleibenden Werten für Pitch und Feature-Size

Der Parameter Pitch bestimmt die Periodizität der Kantenstruktur. Daraus folgt, dass mit steigendem Pitch-Wert die Anzahl der Features sinkt. Dieser Effekt ist in der Abbildung 3.2 in der rechten Spalte dargestellt. Der dritte Startparameter Feature-Size bestimmt die Größe der Features. Dabei kann die Feature-Size nicht größer als Pitch gewählt werden, da sonst keine Kantenstruktur mehr erkennbar wäre. Der Effekt der Feature-Size wird in Abbildung 3.2 in der linken Spalte dargestellt.

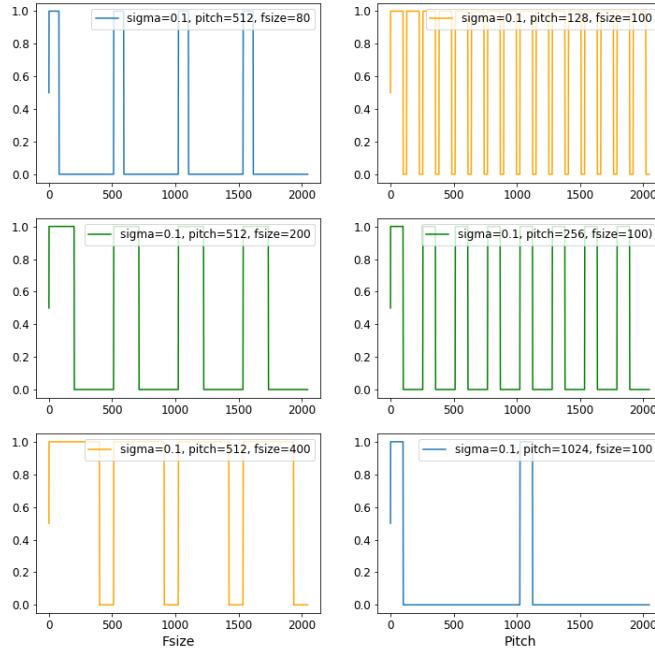


Abbildung 3.2: Vergleich der Kantenstruktur unter variierenden Pitch-Werten (rechte Spalte) und variierenden Feature-Size-Werten (linke Spalte)

3.2.3 Simulation der Streuung

Wie im Kapitel 2.1.2 beschrieben, skaliert das aufgenommene Detektorbild zum Betragsquadrat der Fouriertransformation der Elektronendichte η . Im Fall der Simulation ist die Elektronendichte eine diskrete eindimensionale Elektronendichtheverteilung. Deswegen wird zur Bildung des Detektorbildes eine eindimensionale diskrete Fouriertransformation (3.3) verwendet.

$$\hat{a}_k = \sum_{j=0}^{N-1} e^{-2\pi i \cdot \frac{jk}{N}} \eta_j \text{ für } k = 0, \dots, N-1 \quad (3.3)$$

Das entstandene Produkt der Fouriertransformation ist im Raum der Komplexen Zahlen und weist den gleichen Informationsgehalt wie die Elektronendichtheverteilung auf. Das heißt die Amplituden- und Phaseninformationen sind nach wie vor vorhanden. Wie in Kapitel 1 beschrieben, ist der Detektor Zeit integrierend und die Phase geht verloren, weswegen die Phaseninformation im weiteren Simulationsverlauf nicht weiter verwendet wird. Um schlussendlich die aufgenommenen Intensitäten des Detektors (siehe Abbildung 3.3) zu erhalten, wird das Betragsquadrat des Amplitudenspektrums gebildet ($\Phi = |\hat{a}_k|^2$)

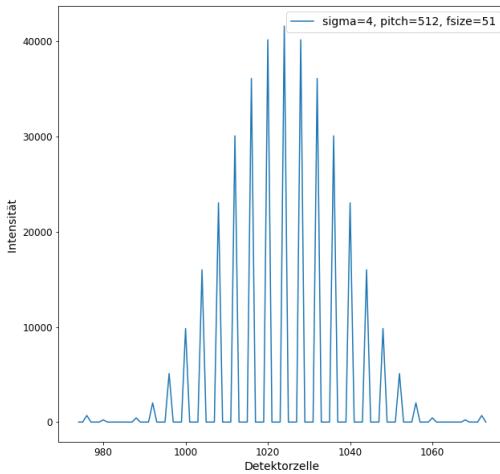


Abbildung 3.3: Resultierendes Detektorbild als Endprodukt der Simulation

3.2.4 Auswirkung der Startparameter auf das Simulationsergebnis

In diesem Kapitel werden die Auswirkungen der Startparameter auf das Detektorbild untersucht. In der Abbildung 3.6 ist der Einfluss der Parameter Pitch und Feature-Size auf das Detektorbild dargestellt. Es ist zu erkennen, dass die Anzahl der Peaks (Intensitäten größer 0) maßgeblich durch das Verhältnis von Pitch und Feature-Size bestimmt werden. Auch die maximale Intensität wird durch das Verhältnis von Pitch und Feature-Size beeinflusst. Des Weiteren ist zu beobachten, dass mit steigendem Pitch-Wert sich die Abstände der Peaks verkleinern. Wird der Feature-Size-Wert erhöht, ist zu erkennen, dass die Anzahl der Peaks sich verringert und die maximale Intensität steigt. Der Effekt des Feature-Size-Wertes auf die maximale Intensität kann ebenfalls im Plot in Abbildung 3.7 beobachtet werden. Der letzte Startparameter Sigma hat mit steigenden Wert ähnliche Auswirkungen auf die Detektorbilder wie Feature-Size. Bei steigendem Sigma-Wert steigt das Intensitätsmaximum und die Anzahl der Peaks verringert sich. Und es kommt zu einer Glättung der Randpeaks (Peaks werden kleiner oder verschwinden). Feature-Size zeigt ähnliche Effekte, jedoch skaliert der Parameter Sigma anders als Feature-Size. Kleinere Unterschiede im Sigma-Wert können große Auswirkungen auf die Anzahl der Peaks haben. Auch der Anstieg des Intensitätsmaximums in Abhängigkeit zu Sigma ist geringer im Vergleich zur Abhängigkeit zu Feature-Size. Zusätzlich ist zu bemerken, dass es einen Schwellwert gibt, bis der Effekt von Sigma Auswirkungen auf die maximale Intensität hat. Der Effekt von Sigma auf die maximale Intensität im Detektorbild ist in Abbildung 3.5 zu sehen. Auswirkungen auf das Detektorbild sind in Abbildung 3.4 dargestellt.

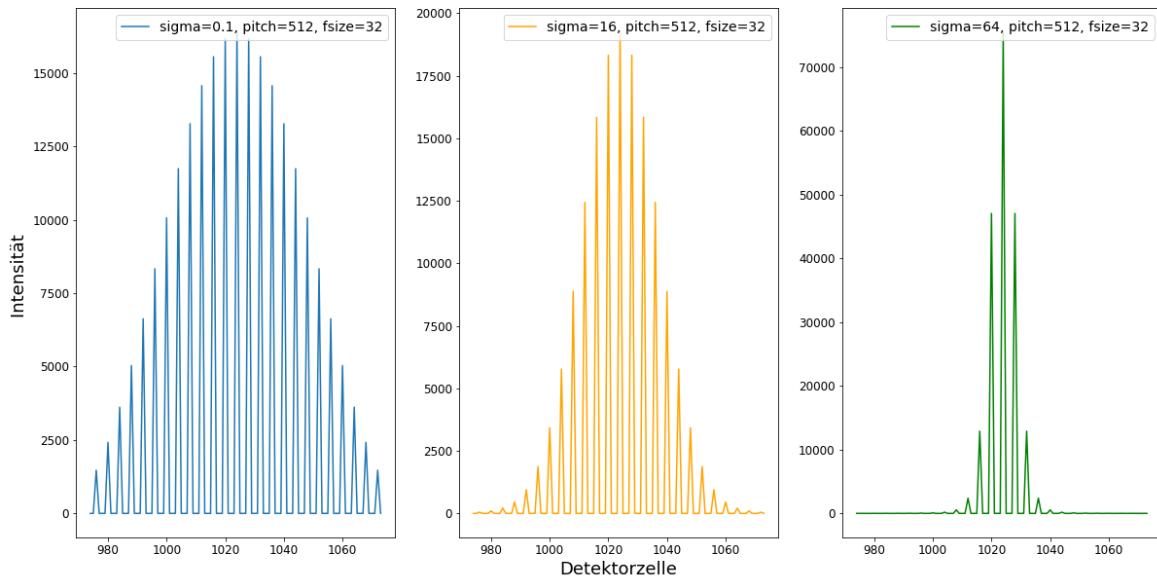


Abbildung 3.4: Auswirkung des Parameter Sigma auf das Detektorbild

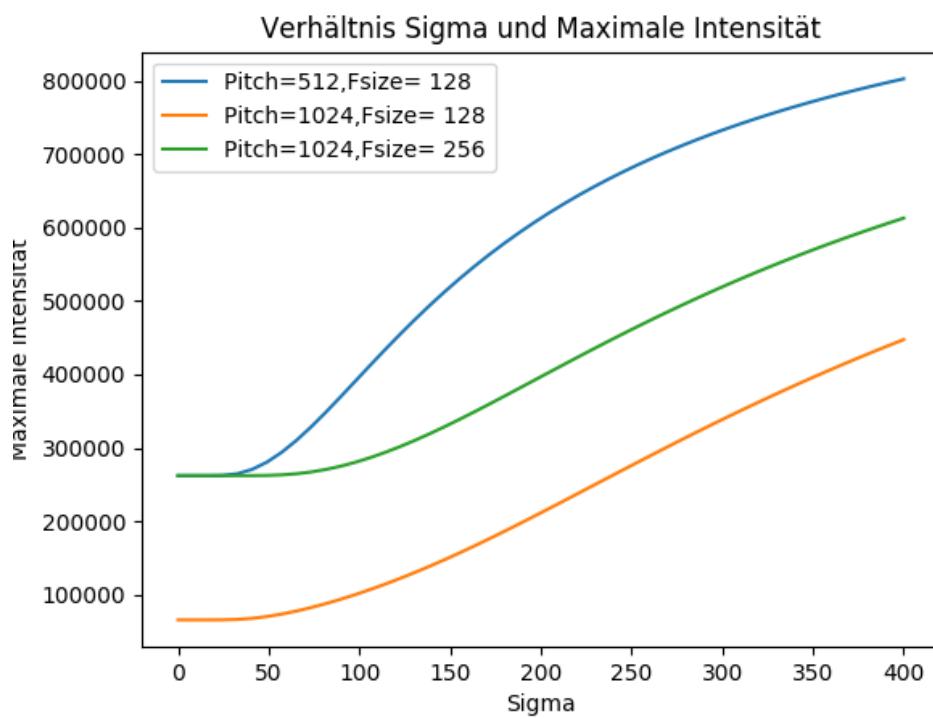


Abbildung 3.5: Auswirkung des Parameters Sigma auf die maximale Intensität im Detektorbild

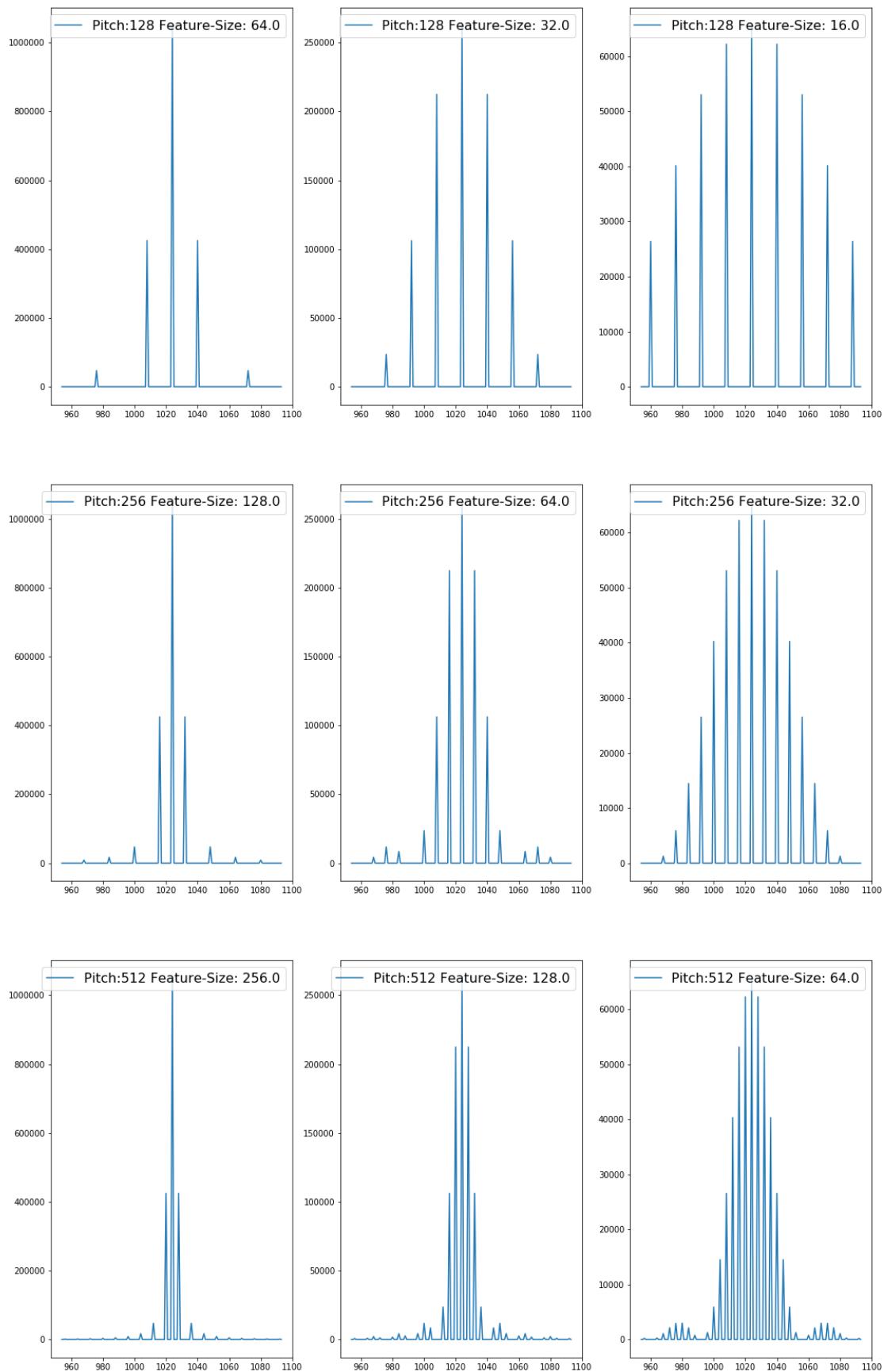


Abbildung 3.6: Auswirkung der Parameter Pitch und Feature-Size auf das Detektorbild. In der Tabelle wurde der Pitch-Wert zeilenweise erhöht. In jeder Spalte wurde der Feature-Size-Wert in Relation zum Pitch-Wert geändert. 1.Spalte: $\frac{1}{2}$ Pitch, 2.Spalte: $\frac{1}{4}$ Pitch, 3.Spalte: $\frac{1}{8}$ Pitch

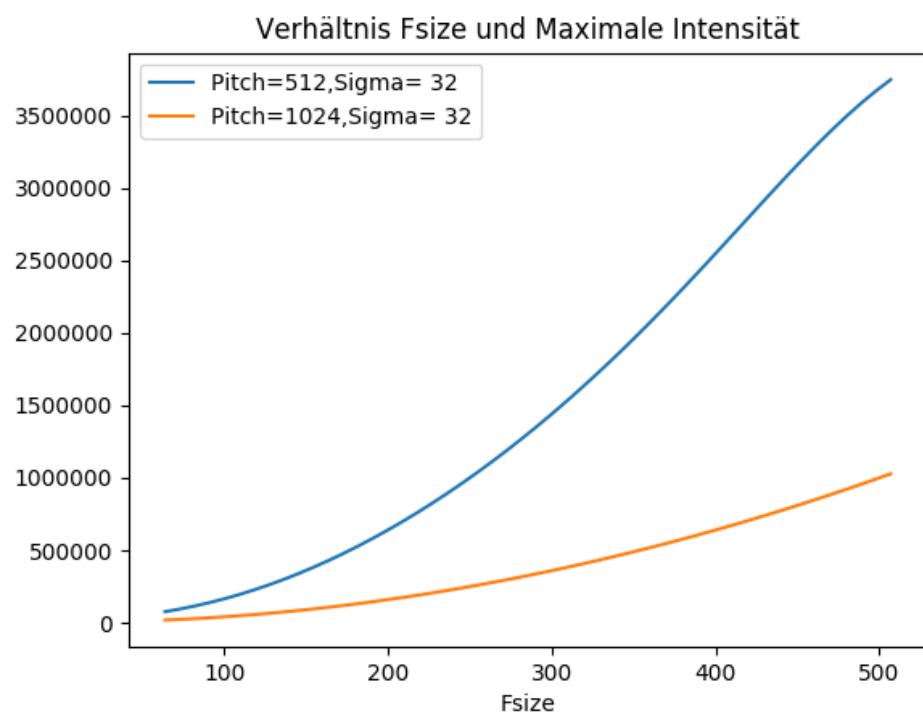


Abbildung 3.7: Auswirkung des Feature-Size-Wertes auf die maximale Intensität im Detektorbild.

3.2.4.1 Mathematische Erklärung der Starparametereffekte

Wie in Kapitel 3 beschrieben, ergibt sich die Gitterstruktur des Gratings aus einer Faltung aus Gauss-Verteilung (g), Impulsverteilung (h) und Rechteckimpuls(f). Diese Faltung wird dann fouriertransformiert um das Detektorbild zu erzeugen. Eine wichtige Eigenschaft der Fouriertransformation F ist, dass Fourier-Faltungs-Theorem, welches besagt, dass die Fouriertransformation einer Faltung äquivalent zum Produkt der Fouriertransformationen der Faltungsfunktionen ist (Gleichung (3.4))[Rot06]. Für die Faltung der drei Faltungskomponenten f , g , und h ergibt sich :

$$F(f * g * h) = F(f) \cdot F(g) \cdot F(h) \quad (3.4)$$

Die Fouriertransformation einer Rechteckimpulses der Breite $2T$ wird laut [Hem00] folgend beschrieben: „Als Fourier-Transformierte ergibt sich eine oszillierende Funktion [...] Je schmäler der Rechteckimpuls - desto breiter die Frequenzverteilung [...] und umgekehrt.“ Aus dieser Eigenenschaft kann geschlussfolgert werden, dass bei steigendem Feature-Size-Wert die Frequenzverteilung $F(f)$ schmäler wird. Augrund der Muliplikation in (3.4) kann geschlussfolgert werden, dass durch eine Erhöhung des Feature-Size-Wertes die maximale Intensität und die Intensitäten der Peaks im Zentrum erhöht werden. Dieser Effekt ist auch in Abbildung 3.6 festzustellen.

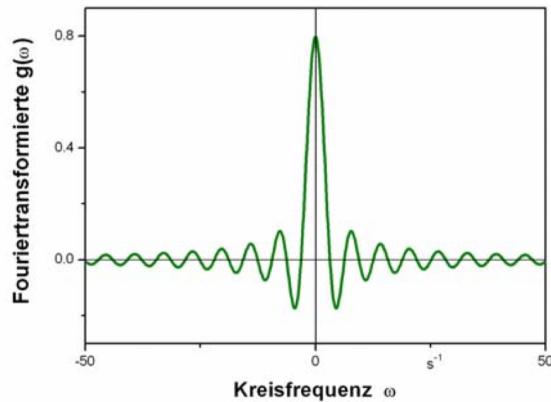


Abbildung 3.8: Fourier-Transformation des Rechteckimpulses. Bildquelle: [Hem00]

Die zweite Komponente der Faltung ist die Impulsverteilung h oder Dirac-Kamm. „Unter einem Dirac-Kamm versteht man [...] eine Folge von äquidistanten Deltafunktionen“ ([Leh06], Seite 18).

$$\text{comb}(t) = \sum_{n=-\infty}^{+\infty} \delta(t - n\Delta t) \text{ mit } \delta(t) = 1 \quad (3.5)$$

Das Ergebnis der Fouriertransformation \mathcal{F} des Dirac-Kamms ist ebenfalls ein Dirac-Kamm, dessen Periodizität sich reziprok zum ursprünglichen Dirac-Kamm verhält [Leh06]:

$$\mathcal{F}\{\text{comb}(t)\} = \text{COMB}(f) = \frac{1}{\Delta t} \sum_{n=-\infty}^{+\infty} \delta(f - n/\Delta t) \quad (3.6)$$

Diese Eigenschaft spiegelt sich auch bei der Erhöhung des Pitch-Wertes wieder (siehe Abbildung 3.6). Durch die Erhöhung des Pitch-Wertes kommt es zu einer Verringerung der maximalen Intensität und die Abstände zwischen den Peaks verringern sich. Der letzte Bestandteil der Faltung ist die Faltung mit der Gauss-Verteilung. Eine Erhöhung des Sigma-Wertes führt zu einer Steigerung der maximalen

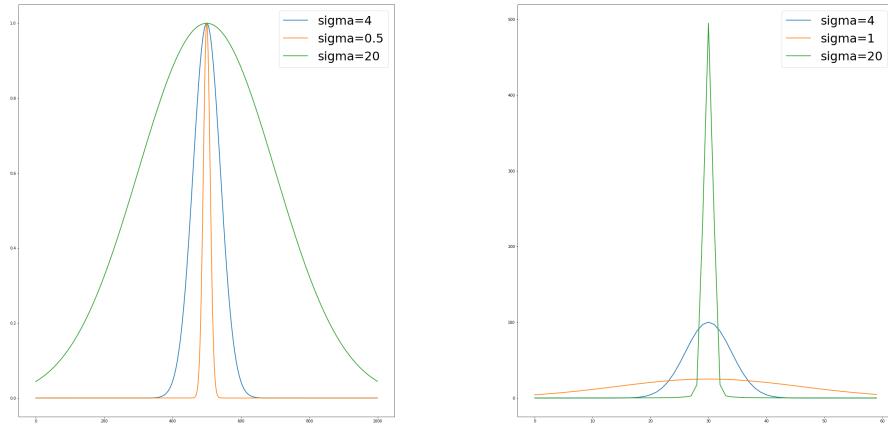


Abbildung 3.9: Vergleich der Gauss-Verteilung (links) und deren Fourier-Transformation (rechts)

Intensität und einer Glättung der Randpeaks. Betrachtet man die Fouriertransformation einer Gauss-Verteilung (3.7), dann ist zu erkennen, dass die Abweichung σ im Zähler des Exponenten steht. Daraus kann ein reziprokes Verhalten der Abweichung in der Fouriertransformation geschlussfolgert werden (siehe Abbildung 3.9. Für das Detektorbild bedeutet dies mit Hinblick auf das Fourier-Faltungs-Theorem, dass mit steigendem Sigma-Wert die Intensitäten im Zentrum erhöht werden und Peaks am Rand geglättet werden (siehe Abbildung 3.4)).

$$F(g) = G(\omega) = e^{-\frac{\omega^2 \sigma^2}{2}} \quad (3.7)$$

4 Lösungsansatz

In diesem Kapitel wird der Lösungsansatz für die Parameterrekonstruktion beschrieben. Dabei wird auf die Gesamtarchitektur, der konzeptionelle Aufbau der einzelnen Komponenten und auf den Aufbau der Neural Networks und deren Training eingegangen.

4.1 Gesamtarchitektur

Wie in Kapitel 3 beschrieben, ergibt sich die Gitterstruktur des Targets aus einer Faltung von Impulsfunktion, Gauss-Verteilung und Rechteckfunktion. Laut Faltungstheorem, welches besagt, dass die Faltung zweier Funktionen durch die Fouriertransformation \mathcal{F} in ein Produkt überführt wird, sind die Fouriertransformationen der einzelnen Komponenten der Faltung im Fourierbild vorhanden. Aufgrund des Faltungstheorems wurde die Parameterrekonstruktion als Zweischrittverfahren konstruiert. Die Architektur besteht aus zwei Neural Networks. Das erste Neural Network ist ein Convolutional Neural Network, welches versucht anhand des Detektorbildes den Pitch-Wert zu schätzen. Der geschätzte Pitch-Wert wird anschließend dafür genutzt die Impulsverteilung zu rekonstruieren. Dafür wird folgender funktionaler Zusammenhang verwendet:

$$\delta_{pitch}(x) = \begin{cases} 1 & \text{für } x \bmod pitch = 0 \\ 0 & \text{sonst} \end{cases} \quad (4.1)$$

Die generierte Impulsverteilung dient als Support für das zweite Neural Network (Fully-Connected Neural Network), um eine genauere Schätzung des Feature-Size- und Sigma-Wertes zu ermöglichen. Dabei wird der durch das CNN erstellte Support mit dem Detektorbild konkateniert. Die Konkatenation zwischen Detektorbild und Support wird als Input in das Fully-Connected Neural Network gegeben. In Abbildung 4.1 ist ein schematischer Aufbau des gesamten Architektur zu sehen. Der Output ist dabei nicht als Komponente zu betrachten, sondern nur als Verbindungsstück der Schätzungen der beiden Neural Networks. Aufgrund verschiedener Peak-Pattern bei verschiedenen Pitch-Werten wurde für die Schätzung des Pitch-Wertes ein CNN gewählt. Unterschiedlichen Feature-Size- und Sigma-Werten haben globale Auswirkung auf das Detektorbild. Deshalb wurde für die Schätzung des Feature-Size und Sigma-Wertes ein Fully-Connected Neural Network gewählt.

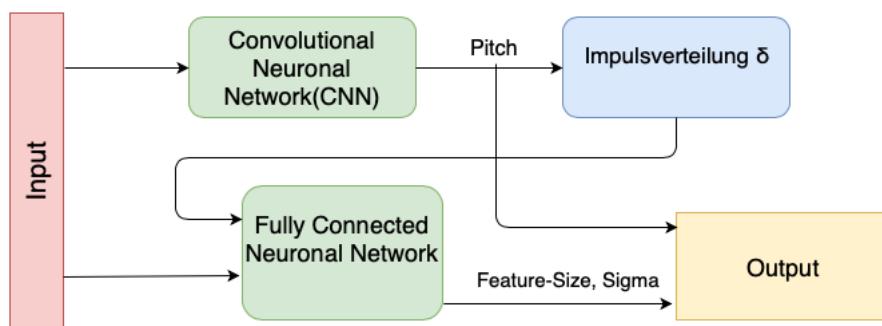


Abbildung 4.1: Schematische Beschreibung der Gesamtarchitektur

Variable	Minimum	Maximum
Learning-Rate	1e-15	1e-5
Anzahl Convolutional Layer	3	12
Filtergröße	4	20
Anzahl der Filter pro Layer	16	64

Tabelle 4.1: Definition des Suchraums für die Bayes'sche Optimierung

4.2 Convolutional Neural Network

4.2.1 Bayes'sche Optimierung

Das Finden der geeigneten Hyperparameter für ein Neural Network ist eine große Herausforderung. Der Suchraum wird durch zahlreiche Parameter wie Tiefe des Netzes, Anzahl der Filter, Größe der Filter usw. sehr groß. Ansätze wie Random- oder Gridsearch sind bekannte Ansätze für das Finden geeigneter Hyperparameter. Jedoch beziehen diese Ansätze die bisher evaluierten Hyperparameter nicht für die Wahl des als nächstes zu evaluierten Hyperparameters ein. In diesem Bezug ist eine Evaluierung das Training eines Neuronalen Netzes, welches unter festgelegten Hyperparametern erstellt wurde und dieses nach einem festgelegten Qualitätsmaß bewertet wurde. Im Gegensatz zu den eben erwähnten Optimierungsverfahren verfolgt die Bayes'sche Optimierung den Ansatz bereits evaluierte Hyperparametersätze in die Optimierung mit einzubeziehen. Die Grundidee der Bayes'schen Optimierung ist das Erstellen einer Wahrscheinlichkeitverteilung $p(L|\theta)$ mit L der Qualität eines Neural Networks, da s in Abhängigkeit eines Hyperparamtersatzes θ erstellt wurde. Zu Beginn der Bayes'schen Optimierung werden bereits bekannte Evaluierungspunkte verwendet oder ein festgelegter Hyperparametersatz evaluiert, um ein erstes Wahrscheinlichkeitsmodell mit Hilfe der Maximum Likelihood-Methode [Myu02] zu erstellen. Das erstellte Wahrscheinlichkeitsmodell wird dann benutzt, um den nächsten Evaluierungspunkt aus dem definierten Suchraum zu bestimmen. Nach dessen Evaluierung wird erneut das Wahrscheinlichkeitsmodell angepasst und ein neuer Evaluierungspunkt gefunden. Dieser Prozess wird solange durchgeführt, bis eine bestimmt Anzahl von Iterationen erreicht wurde. Die genauere Beschreibung des Algorithmus kann [Fra18] entnommen. Um eine Bayes'sche Optimierung für das CNNs vorzunehmen, wurde die Implementierung von Scikit-learn verwendet. Dabei wurden die Hyperparameter: Anzahl der Convolutional Layer, Anzahl der Filter pro Convolutional Layer, Filtergröße und die Learning-Rate des Optimizers über die Bayes'scher Optimierung gefunden. Die Bayes'sche Optimierung wurde durch den Suchraum aus Tabelle 4.1 eingeschränkt. Die Einschränkungen wurden aus Erfahrungswerten vorheriger Experimente getroffen. Ein Evaluierungsschritt ist die Erstellung des CNN anhand des festgelegten Hyperparametersatzes, das Trainieren des CNNs mit 1024 Bildern über 3000 Epochen und die Bestimmung der Qualität. Als Qualitätsmaß wurde der beste erreichte Mean-Squared-Error auf dem Validierungsdatensatz verwendet. Insgesamt wurden 50 Evaluierungsschritte ausgeführt. Zusätzlich wurden Residual Strukturen mit einer Sprungweite von 2 dem erstellten CNN hinzugefügt.

4.2.2 Architektur

Das Ergebnis der Bayes'schen Optimierung ist ein CNN mit einem Block von Convolutional Layers. Die Größe des Blockes, die Anzahl der Filter und die Größe der Filter pro Convolutional Layer wurden durch die Bayes'sche Optimierung bestimmt. Nach dem Block von Convolutional Layers wurde zusätzlich ein Convolutional-Layer mit einem Filter und der Filtergröße 1 hinzugefügt. Dieser Layer hat eine Dimensionreduktion zu folge, um beim anschließenden Dense-Layer(Fully-Connected Layer) nicht zu viele Verbindungen zu erzeugen. Zwischen dem Regressions-Layer und dem ersten Dense-Layer befindet sich ein weiterer Hidden-Layer, der eine weitere Dimensionreduktion erzeugt. Als letzte Layer wurde eine weitere Dense-Layer mit linearer Aktivierungsfunktion hinzugefügt. Die Entscheidung für eine li-

neare Aktivierungsfunktion wurde aus den Erfahrungen vorheriger Experimente getroffen. Wurde die Ausgabe-Layer mit einer ReLu-Aktivierungsfunktion versehen, starb das Output-Neuron während des Trainings frühzeitig und es wurden ausschließlich Pitch-Werte mit dem Wert 0 geschätzt. In Tabelle 4.2 ist die komplette Architektur des CNNs dargestellt.

Layer	Input	Anzahl Filter	Filtergröße	Anzahl der Neuronen	Aktivierungssf.
1DConv	Detektorbild	32	16	-	ReLU
1DConv	Layer 1	32	16	-	ReLU
1DConv	Layer 2 + Layer1	32	16	-	ReLU
1DConv	Layer3	32	16	-	ReLU
1DConv	Layer4 + Layer3	32	16	-	ReLU
1DConv	Layer5	32	16	-	ReLU
1DConv	Layer6 + Layer5	32	16	-	ReLU
1DConv	Layer7	32	16	-	ReLU
1DConv	Layer8 + Layer7	32	16	-	ReLU
1DConv	Layer9	32	16	-	ReLU
1DConv	Layer10	1	1	-	ReLU
Dense	Layer11	-	-	64	ReLU
Regression-Layer	Layer12	-	-	1	Linear

Tabelle 4.2: Architektur des Convolutional Neural Networks

4.3 Fully-Connected Neural Network

Das Fully-Connected Neural Network besteht ausschließlich aus Dense-Layern. Die Aufgabe des Fully-Connected Neural Networks ist die Schätzung des Sigma- und Feature-Size-Wertes. Der erste Layer bekommt die Konkatenation des Detektorbildes und der Impulsverteilung, welche aus dem vom CNN geschätzten Pitch-Wert erstellt wurde, als Input. Die Architektur des Fully-Connected Neural Networks wurde aus Zeitgründen nicht mit der Bayes'schen Optimierung optimiert, sondern aus Erkenntnissen mehrerer Experimenten gebildet. In Tabelle 4.3 ist die Architektur des Fully-Connected Neural Networks dargestellt.

Layer	Input	Anzahl Neuronen	Aktivierungsfunktion
Dense-Layer	(Detektorbild, Impulsverteilung)	4096	ReLU
Dense-Layer	Layer1	2048	ReLU
Dense-Layer	Layer2	2048	ReLU
Dense-Layer	Layer3	64	ReLU
Regression-Layer	Layer4	2	Linear

Tabelle 4.3: Architektur des Fully-Connected Neural Networks

4.4 Training

Das Training der beiden beschriebenen Neural Networks erfolgt getrennt. Als erstes wird der Schätzer des Pitch-Wertes trainiert(CNN), da dieser die Grundlage für das Training des Fully-Connected Neural Networks bildet. Anschließend wird der Schätzer für den Sigma- und Feature-Size-Wert trainiert. Für die Generierung der Impulsverteilung, welche für das Training des Fully-Connected Neural Networks benötigt wird, wird ein festes CNN verwendet. Das heißt, dass sich die Architektur bzw. die Gewichte des CNN's während des Trainings des Fully-Connected Neural Networks nicht verändern.

Für das Training beider Neural Networks wurde der Adam-Optimizer verwendet. Der Adam-Optimizer ist eine Erweiterung des klassischen Gradientenabstiegsverfahren. Der Adam-Optimizer verknüpft die Vorteile zweier weiterer bekannter Optimierungsverfahren: Adaptive Gradient Algorithm (AdaGrad) und Root Mean Square Propagation (RMSProp) [Bro17]. Der Adaptive Gradient Algorithmus hat den Vorteil, dass er für jeden Parameter eine unterschiedliche Learning-Rate festlegt, dadurch ist es möglich kleinen Gradienten entgegenzuwirken [Rud16][Bro17]. Die Root Mean Square Propagation stellt ebenfalls für jede Variable unterschiedliche Learning-Rates bereit, jedoch werden diese Learning-Rates hinsichtlich des Trainingsverlaufs angepasst. Zum Beispiel können zu schnell propagierende Variablen durch eine geringere Learning-Rate ausgebremst werden [Rud16][Bro17].

Für jedes Neural Network steht wie in Kapitel 5.4 eine eigene Datenbasis zu Verfügung, welche mit Hilfe von Batch-Processing trainiert wurde. Beim Batch-Processing werden dem Neural Network mehrere Inputdaten gleichzeitig zur Verfügung gestellt und durch das Netz propagiert [Kri07].

Um die Qualität der Gesamtarchitektur zu überprüfen, wurde der Pitch- und Feature-Size-Schätzer auf zwei verschiedene Weisen trainiert. Der erste Trainingsansatz ist das Training des Netzwerks mit Hilfe der Impulsverteilungen, welche durch die Pitch-Werte des CNN's generiert wurden. Im zweiten Trainingsansatz werden die Impulsverteilungen durch das Pitch-Label generiert. Für beide Neural Networks wurde MSE (2.9) als Errorfunktion verwendet.

Der gesamte Trainingsprozess, wurde mit Tensorflow realisiert und mit Hilfe des Tracking-Tools Tensorboard überwacht und alle 50 Epochen mit Hilfe des Validierung-Datensatzes evaluiert.

5 Datenbasis

5.1 Generator

Um einen Machine-Learning-Ansatz auszuführen, muss eine ausreichend große Datenbasis existieren. Dazu wurde im Rahmen dieser Arbeit ein Generator entwickelt, welcher mit Hilfe der vorher beschriebenen Simulation ausreichend viele Trainings-, Validierungs- und Testsdaten(Evaluierung) produzieren kann. Der erste Schritt der Generierung der Datenbasis ist die Festlegung der Menge P von Startparametern, wobei ein Startparameter ein Tripel (Feature-Size, Pitch, Sigma) ist. Um die Berechnung der Simulationsdaten, welche aus P resultieren, zu beschleunigen wurde die Simulation aus Kapitel 3 mit Hilfe des Message Parsing Interfaces (MPI) parallelisiert. MPI dupliziert auszuführenden Code auf N beliebige Prozesse, welche mit Hilfe von vordefinierten Routinen miteinander kommunizieren können [GL94]. Wurde die Menge von Startparametern festgelegt, wird diese Menge über MPI-Routinen auf N Prozesse aufgeteilt. Jeder Prozess führt für seine zugewiesene Menge von Startparametern die in Kapitel 3 beschriebene Simulation aus und schreibt deren Ausgabe als Dateien in den Speicher des Rechenclusters. Zusätzlich zum Simulationsergebnis wird die Menge der Startparameter, welche als Labels für das Training des Neural Networks vorgesehen sind, gespeichert.

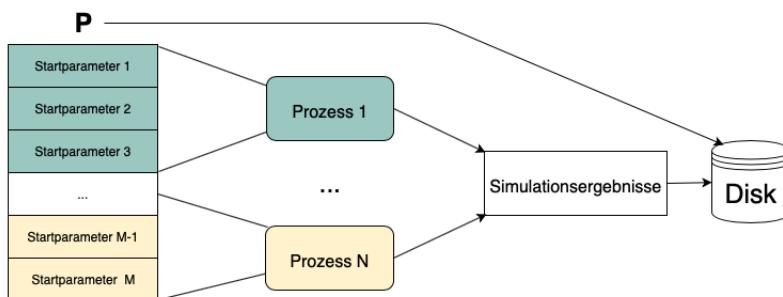


Abbildung 5.1: Schematischer Aufbau des Generators

5.2 Wahl der Eingangsparameter

Um einen validen Datensatz zu erstellen, müssen diese Abhängigkeiten berücksichtigt werden. Der erste Startparameter Pitch ist der Parameter, welcher den Definitionsbereich der anderen beiden Parametern Feature-Size und Sigma signifikant bestimmt. Wie in Kapitel 3 bemerkt, bestimmt Pitch die Periodizität der Gitterstruktur. Um aussagekräftige Detektorausgaben zu produzieren, wurde festgelegt, dass jedes Target mindestens vier und maximal 32 Features besitzt. Das bedeutet im Umkehrschluss, dass bei einer festen Targetbreite von 2048 Pixeln sich ein minimaler Pitch-Wert bei 64 und der maximale Pitch-Wert bei 512 Pixeln ergibt. Je nach Wahl des Pitch-Wertes, ergeben sich die Definitionsbereiche für Sigma und Feature-Size. Um zu kleine Features zu vermeiden, wurde der minimale Feature-Size-Wert auf 25 % des Pitch-Wertes festgelegt. Für ausreichend große Abstände zwischen den Features, wurde der maximale Feature-Size-Wert auf 75 % des Pitch-Wertes beschränkt. Nach der Bestimmung von Pitch und Feature Size kann der Definitionsbereich von Sigma festgesetzt werden. Der minimale Sigma-Wert ist auf 10^{-9} festgelegt, um auch Simulationsergebnisse ohne Hochintensitätslasereinfluss zu generieren. Zur Bestim-

mung des maximalen Sigma-Wertes ist der Abstand zwischen zwei Features notwendig. Der Abstand zwischen zwei Features ergibt sich aus Pitch - Feature-Size. Da der Aufweichungseffekt gleichmäßig auf das Feature wirkt, muss für den maximalen Sigma-Wert der Feature-Abstand halbiert werden. Um keine zu hohen Aufweichungseffekte zu erhalten, wurde die obere Grenze des Sigma-Wertes auf $(\text{Pitch} - \text{Fsize}) / 4$ festgesetzt. Für die Umsetzung der voneinander abhängigen Wertebereichen, werden Folgen konstruiert, dessen Elemente zu Startparameter Tripeln konkateniert werden. Eine Beispielhafte Implementierung ist in der Abbildung 5.2 zu sehen, welche jeweils eine Folge mit 64 Elementen (Anzahl der Prozessoren) erstellt.

```
num_pitches = 64
num_fsizes = 64
num_sigmas = 64
for pitch in np.linspace(64, 512, num=num_pitches):
    for feature_size in np.linspace(0.25*pitch, 0.75*pitch, num=num_fsizes):
        for sigma in np.linspace(1e-9, (pitch-feature_size)/4, num=num_sigmas):
            add_to_P(sigma, pitch, feature_size, number)
```

Abbildung 5.2: Umsetzung der Definitionsbereiche mit Hilfe drei verschachtelter Schleifen, welche von-einander Abhängig sind

5.3 Performance

Die Erstellung der Datenbasis wurde auf dem Rechencluster Hypnos des Helmholtz Zentrums Dresden Rossendorf auf 64 AMD 16-Kern Opteron Prozessoren ausgeführt. Die Parallelisierung der Simulation hat einen großen Speedup zur Folge, so dass 100.000 Bilder in knapp zwei Stunden anstatt in ca. 50 h Stunden(* lineare Regression aus Rechenzeiten kleinerer Datensätze) generiert werden können. Diese kurze Rechendauer ermöglicht ein schnelleres Validieren und Korrigieren des erstellten Datensatzes. Die Rechenzeiten für kleinere Datensätze sind der Tabelle 5.1 zu sehen. Bei kleineren Datensätzen ist der Speedup nicht signifikant, da durch die Parallelisierung ein Kommunikationsoverhead entsteht.

	1 Bild	100 Bilder	1000 Bilder	100.000 Bilder
sequentiell	1,889 s	187,613 s	1870,095 s	186940,286 s *
Generator	1,917 s	4,531 s	42,831 s	6900,243 s

Tabelle 5.1: Vergleich der Berechnungsdauer für verschiedene große Datensätze

5.4 Trainings-, Validierungs- und Testdatensatz

Für das Training von Pitch, Feature-Size und Sigma stehen zwei verschiedene Datensätze zur Verfügung. Der erste Datensatz wurde für das Training des Neural Networks erstellt, welches Pitch schätzen soll. Dieser Datensatz hat die Besonderheit, dass die Folge vom minimalen Pitch-Wert bis zum maximalen Pitch-Wert deutlich höher abgetastet ist, um dem Neural Network eine möglichst aussagekräftige Verteilung von Pitch-Werten und deren Auswirkung auf das Detektorbild zu geben. Die Feature-Size- und Sigma-Folgen wurden deutlich geringer abgetastet, jedoch steht aufgrund der verschachtelten Schleifen zur Generierung der Startparameter (Abbildung 5.2) eine ausreichend große Verteilung von Sigma und Feature-Size-Werten für das Training von Pitch zur Verfügung (Tabelle 5.2).

Der Datensatz für das Training des Neural Networks weist eine andere Verteilung wie der Pitch-Datensatz auf. In diesem Datensatz wurden die Folgen für Pitch, Sigma und Feature-Size gleich abgetastet, jedoch

Dataset	Anzahl Pitch	Anzahl Feature-Size	Anzahl Sigma
CNN	368	23	23
Fully-Connected	46	46	46

Tabelle 5.2: Abtastwerte für die jeweiligen Startparameter. Diese Werte werden wie in Abbildung 5.2 verarbeitet

ist die Verteilung der Sigma- und Feature-Size-Werte aufgrund der Implementierung der Parametergenerierung deutlich ausgeprägter. In der Tabelle 5.2 sind die Abtastwerte der jeweiligen Folgen zu sehen. Für das Training, die Validierung des Trainingsprozesses und die Evaluierung des Neural Networks wurde der jeweilige Datensatz zufällig umsortiert und in Trainings-, Validierung- und Test-Datensatz aufgespalten. Um die Länge nicht zu groß zu gestalten, wurde der Trainingsdatensatz bei beiden Datensätzen auf 16384 Detektorbilder beschränkt. Für die Validierung stehen jedem Neural Network 64 Bilder zur Verfügung. Die restlichen Bilder wurde komplett für eine aussagekräftige Evaluierung verwendet. In Tabelle 5.3 ist die Verteilung der Samples auf Trainings-, Validierungs- und Testdatensatz zu sehen. Aufgrund der höheren Abtastung von Pitch imd Pitch-Datensatz (siehe Tabelle 5.2) besitzt dieser mehr Detektorbilder und somit mehr Samples zur Evaluierung.

Datensatz	Training	Validierung	Test
Pitch	16384	64	178224
Feature-Size und Sigma	16384	64	80888

Tabelle 5.3: Verteilung der Samples auf Trainings-, Validierungs- und Testdatensatz

6 Ergebnisse und Diskussion

6.1 Trainingsprozess

6.1.1 Convolutional Neural Network

Der Trainingsverlauf des CNNs wurde mit Hilfe der Tensorboard-Bibliothek überwacht. Parallel zum Trainingsprozess wurde, in einem Abstand von 50 Epochen, der Trainingsprozess mit Hilfe des Validierungsdatensatzes evaluiert. Um die Qualität des CNNs zu bestimmen, wurde Mean Squared Error als Qualitätsmaß eingesetzt (Gleichung (2.9)). Das CNN wurde über 3000 Epochen trainiert. Der Verlauf des Trainings- und Validierungsfehlers während des Trainingsprozesses ist in Abbildung 6.1 dargestellt.

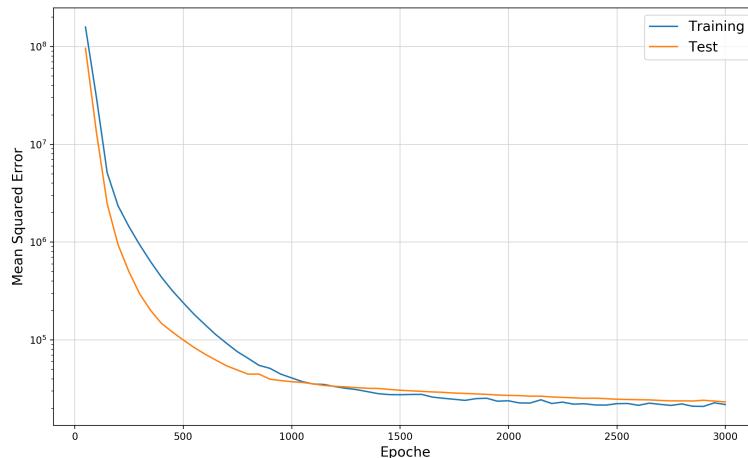


Abbildung 6.1: Verlauf des Trainings- und Validierungsfehlers des CNNs in Anhängigkeit der Epoche

In Abbildung 6.1 ist bis zur tausendsten Epoche ein starker Abfall des Fehlers zu erkennen. Ab der tausendsten Epoche ist ein schwächeres Konvergenzverhalten zu beobachten. Der Verlauf des Trainings- und des Validierungsfehlers ist sehr ähnlich. Somit können nicht trainierte Samples mit nahezu gleicher Qualität wie trainierte Samples geschätzt werden. Dies spiegelt sich auch im ähnlichen minimalen Trainings- und Validierungsfehler wieder. Während des Trainings konnte ein minimaler Trainingsfehler von 20910.45 und ein minimaler Validierungsfehler 23247.19 von erreicht werden.

6.1.2 Fully-Connected Neural Network

In Kapitel 4 wurde beschrieben, dass das Fully-Connected Neural Network auf zwei verschiedene Weisen trainiert wurde. Im ersten Experiment wurde das Fully-Connected Neural Network mit der Impulsverteilung ,welche aus dem Pitch-Wert des CNNs generiert wurde, trainiert. Im zweiten Experiment wurde die Impulsverteilung aus dem Pitch-Label generiert. In beiden Experimenten wurde das Network 500 Epochen trainiert. Die kleinere Epochenzahl resultiert aus dem schnelleren Konvergenzverhalten des Fully-Connected Neural Networks. Bei einer Epochenzahl von 1000 konnten keine besseren Ergebnisse

erzielt werden. Für das Fully-Connected Neural Network wurde ebenfalls Mean Squared Error als Qualitätsmaß verwendet. In Abbildung 6.2 ist die Lernkurve des ersten Experimentes dargestellt. Während des Trainings konnte ein minimaler Trainingfehler von 75.80 und ein minimaler Validierungssfehler von 163.74 erreicht werden. Der Verlauf des Training- und des Validierungsfehlers zeigt ähnliches Konvergenzverhalten, das für Generalisierung spricht. Dennoch gibt es im minimal erreichten Trainings und Validierungsfehlers einen Unterschied von einem Faktor zwei. Somit ist die Qualität des Fully-Connected Neural Networks auf den Trainingsdaten deutlich höher als die der Validierungsdaten.

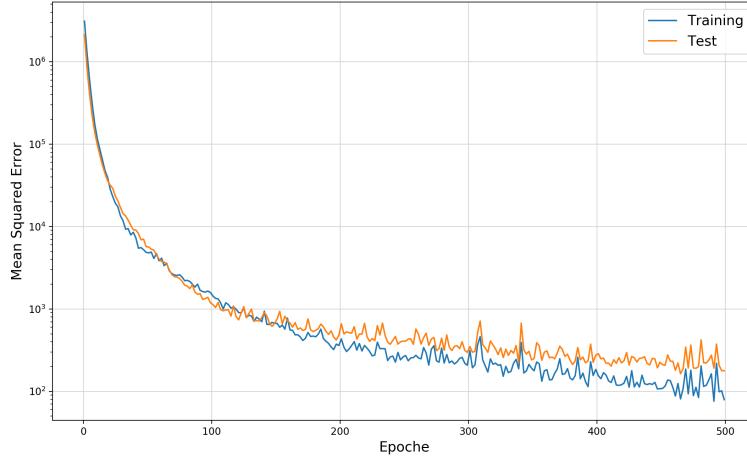


Abbildung 6.2: Verlauf des Trainings- und Validierungsfehlers in Abhängigkeit der Epoche im ersten Experiment

Das zweite Experiment (Abbildung 6.3) zeigt ähnliches Konvergenzverhalten wie beim ersten Experiment. Jedoch kommt es bei einer höheren Epochenzahl zu weniger Oszillation des Fehlers. Der minimale Trainingsfehler liegt bei 100.98 und der minimale Validierungsfehler liegt bei 190.72. Somit konnte eine ähnliche Qualität auf den Trainings- und Validierungsdaten erreicht werden wie beim ersten Experiment.

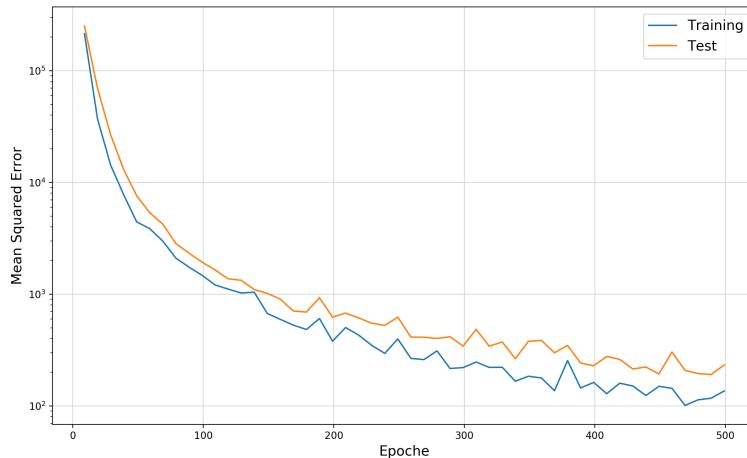


Abbildung 6.3: Verlauf des Trainings- und Validierungsfehlers in Abhängigkeit der Epoche im zweiten Experiment

6.2 Evaluierung

6.2.1 Convolutional Neural Network

Um die Qualität des CNNs zu evaluieren, wurden die nicht trainierten Samples des Testdatensatzes zur Qualitätsbestimmung genutzt. Es konnte eine durchschnittliche absolute Abweichung von 109.73 und eine durchschnittliche relative Abweichung von 37% erreicht werden. Aus der durchschnittlichen relativen und absoluten Abweichung kann eine schlechte Qualität des CNNs geschlussfolgert werden.

Eine wichtige Beobachtung der Analyse des CNNs ist, dass mit steigendem Pitch-Label die Qualität der Pitch-Schätzung sich verringert (siehe Abbildung 6.4). Im Kapitel 3 wurde bereits analysiert, dass mit steigendem Pitch-Wert die Abstände der Peaks kleiner werden und somit ein dichteres Detektorbild entsteht. Daraus kann geschlussfolgert werden, dass bei höherer Peakdichte Pitch schlechter geschätzt werden kann. Zusätzlich ist in Abbildung 6.4 zu erkennen, dass bei höheren Feature-Size-Werten Pitch besser geschätzt werden kann.

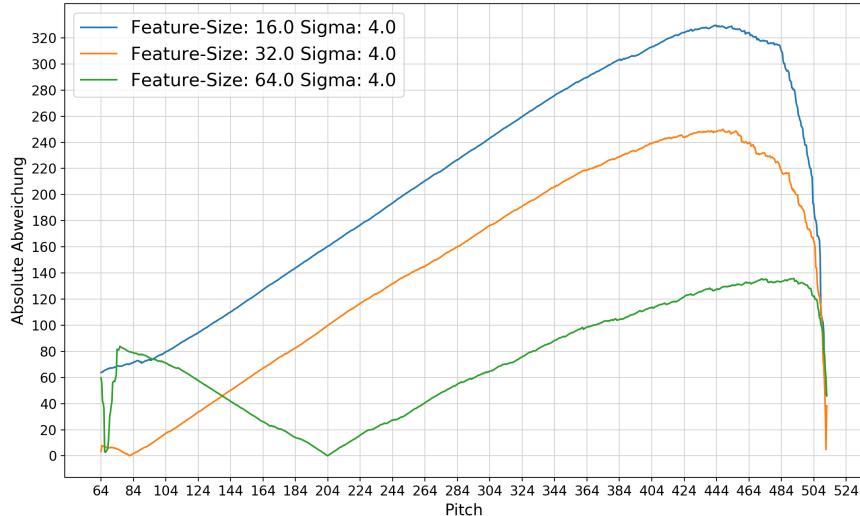


Abbildung 6.4: Qualität der Pitch-Schätzung in Abhängigkeit des Startparameters Feature-Size

Eine wichtige Eigenschaft von CNNs bzw der Convolutional-Layer ist, dass sie Lokalität der Input-Daten nicht berücksichtigen, somit können globale Änderungen, wie zum Beispiel eine Verdichtung des Detektorbildes durch ein CNN schwerer erfasst werden. Dies spiegelt sich auch in den Detektorbildern in Abbildung 6.5 wieder, welche alle im Zentrum des Bildes sehr dicht besetzt sind. Bilder mit niedrigerem Pitch-Wert können besser geschätzt werden, wie die Detektorbilder in Abbildung 6.6. Sie weisen eine geringere Peak-Dichte auf, womit das CNN scheinbar besser umgehen kann.

Ein höheres Sampling der Fouriertransformation könnte die Qualität der Pitchschätzung steigern. In der Simulation wurde das 2048 Pixel große Grating mit einer Fouriertransformation mit 2048 Stützstellen (Sampling) implementiert, dies hat zur Folge, dass Peaks im Detektorbild nur durch einen Wert dargestellt werden und Zwischenwerte komplett verloren gehen. In Abbildung 6.7 ist ein Vergleich der Simulation mit höherer Sampling-Rate(links) und einem Sampling von 1:1. Die Abbildung zeigt außerdem, dass zwischen den höheren Peaks im Zentrum noch kleinere Peaks sind, welche durch die niedrigere Sampling-Rate nicht gemessen werden können. Diese Peaks können laut Fraunhofer-Diffraction dazu genutzt werden die Anzahl der Features und im Umkehrschluss Pitch zu bestimmen [Nav00]. Denn die Anzahl der Minima zwischen dem Intensitätsmaximum und dem nächsten höheren Peak entspricht der

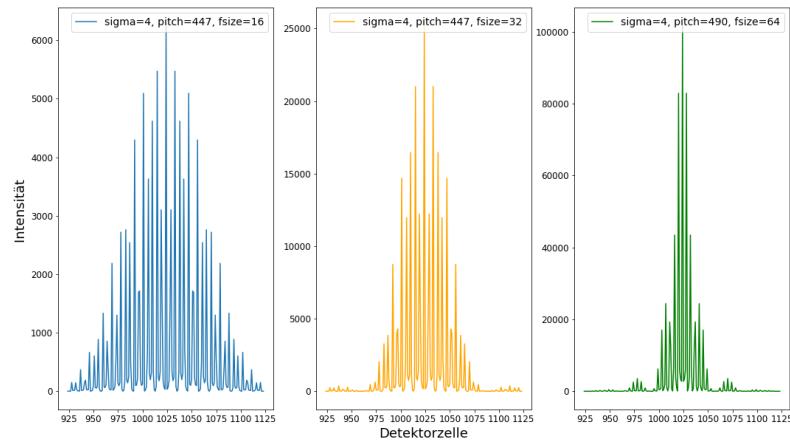


Abbildung 6.5: Detektorbilder zu schlechten Pitch-Schätzungen. Alle drei Detektorbilder weisen mit einer Absoluten Abweichung über 100 eine überdurchschnittliche schlechte Qualität auf.

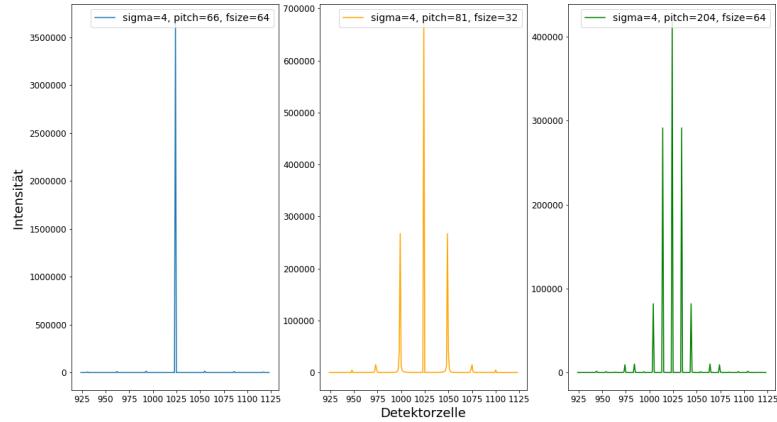


Abbildung 6.6: Detektorbilder zu guten Pitch-Schätzungen

Anzahl der Features plus eins [Nav00]. Somit können nur Parameter des Detektorbildes wie Intensitäten, Peakabstände und Anzahl der Peaks benutzt werden, um Pitch zu schätzen. .

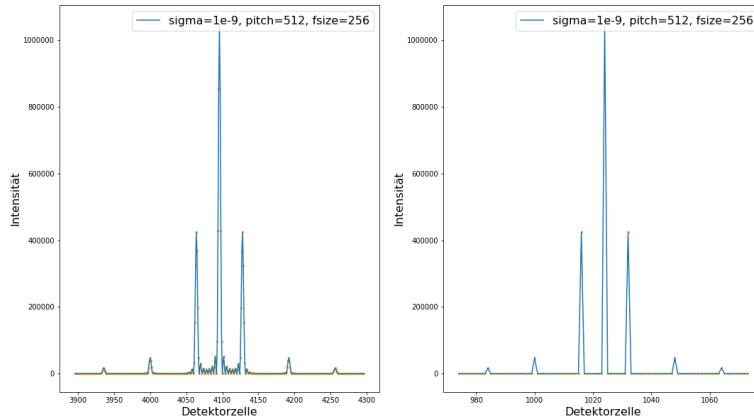


Abbildung 6.7: Unterschied des Detektorbildes bei einem höheren Sampling(links) und einem Sampling von 1:1 (rechts). Die gelben Punkte signalisieren die jeweiligen Sampling-Punkte

6.2.2 Fully-Connected Neural Network

Für das Fully-Connected Neural Network wurde ebenfalls der Testdatensatz zur Evaluierung genutzt. Wie bereits im Trainings-Kapitel (Kapitel 4.4) beschrieben, wurde das Fully-Connected Neural Network auf zwei verschiedene Weisen trainiert. Wie in Kapitel 4 beschrieben, wurde mit Hilfe des geschätzten Pitch-Wertes des CNNs eine Intensitätsverteilung bestimmt, welche die Schätzung des Feature-Size- und Pitch-Wertes verbessern soll. Um diesen Ansatz zu evaluieren, wurde das gleiche Fully-Connected Neural Network mit Impulsverteilungen, welche vom Pitch-Label generiert wurden, trainiert. Der Parameter Feature-Size konnte mit beiden Netzen mit einer durchschnittlichen absoluten Abweichung von 13.043 (CNN) und 13.743(Label) geschätzt werden. Die prozentualen Abweichung liegen 12,46 Prozent (CNN) und 13,713 (Label).

Aus diesen Zahlen kann geschlussfolgert werden, dass Feature-Size mit einer relativ guten Qualität geschätzt werden kann. Des Weiteren kann geschlussfolgert werden, dass der Parameter Pitch und die generierte Impulsverteilung keinen Einfluss auf die Schätzung des Feature-Size-Wertes hat. Dies spiegelt sich auch in der Abbildung 6.8 wieder. In dieser Abbildung ist zu erkennen, dass die Qualität der Feature-Size-Schätzung bei unterschiedlichen Pitch-Werten sehr ähnlich ist. Des Weiteren ist zu erkennen, dass mit steigendem Feature-Size-Wert bis zu einem bestimmten Schwellwert die Qualität gleichbleibend ist. Ab diesem Schwellwert kommt es zu einem starken Qualitätsabfall. Dies kann damit begründet werden, dass der Feature-Size-Wert so hoch ist, dass Randpeaks verschwunden sind und somit nur noch ein Peak für die Schätzung des Feature-Size-Wertes zur Verfügung steht.

	Feature-Size	Sigma
Fully-Connected (CNN)	13,043	13,174
Fully-Connected (Label)	13,713	12,641

Tabelle 6.1: Absolute Abweichung bei der Schätzung von Sigma und Feature-Size auf den Testdaten

	Feature-Size	Sigma >1
Fully-Connected (CNN)	12,46 %	184,43 %
Fully-Connected (Label)	13,50 %	176,75 %

Tabelle 6.2: Prozentuale Abweichung von Sigma und Feature-Size auf den Testdaten

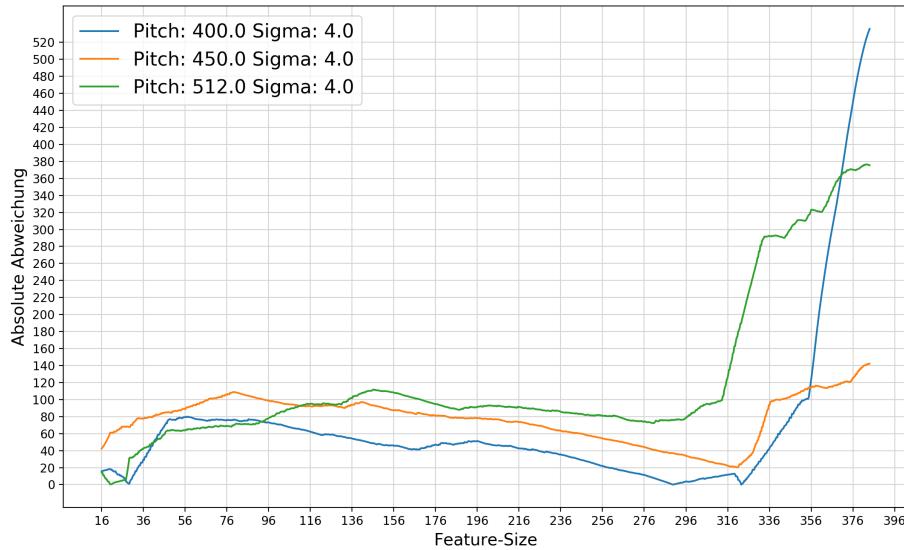


Abbildung 6.8: Absolute Abweichung der Feature-Size-Schätzung des Fully-Connected Neural Network (CNN) in Abhängigkeit von Pitch

Der Startparameter Sigma kann vom Fully-Connected Neural Network deutlich schlechter geschätzt werden als Feature-Size. Es konnte eine durchschnittliche absolute Abweichung 13,174 (CNN) und 12,641 erreicht werden. Da der Wertebereich von Sigma deutlich kleinere ist, zeigt sich der Qualitätsunterschied zur Feature-Size-Schätzung in der prozentualen Abweichung. Es konnte eine prozentuale Abweichung von 184,43 Prozent (CNN) und 176,75 Prozent für Sigma größer 1 erreicht werden. Wobei der Median der Median der prozentualen Abweichung bei 55,09 Prozent (CNN) und 54,165(Label) liegt. Aus den Abweichungszahlen kann geschlussfolgert werden, dass eine bessere Pitch-Schätzung zu einer leicht besseren Sigma-Schätzung beiträgt. In Abbildung 6.9 ist die absolute Abweichung in Abhängigkeit von Sigma und Feature-Size dargestellt. In dieser Abbildung ist zu erkennen, dass die Abweichung nahezu linear mit höheren Sigma-Wert steigt. Außerdem ist zu erkennen, dass der Sigma-Wert besser geschätzt werden kann, wenn der Feature-Size-Wert höher ist. Dieser Zusammenhang kann damit begründet werden, dass bei größeren Features Sigma größer sein muss, um einen Informationsverlust zu erzeugen. Zusätzlich besitzt der erstellte Datensatz sehr wenig Bilder mit hohen Sigma-Werten. Nur 2,3 Prozent der Bilder im Datensatz wurden mit einem Sigma-Wert größer 60 erstellt. Ein weiterer erschwerender Punkt ist, dass Sigma und Feature-Size gleiche Effekte auf das Detektorbild haben. Beide erhöhen das Intensitätsmaximum und verringern die Anzahl der Peaks im Detektorbild. Jedoch erkennt das Network eher den Effekt der Feature-Size auf das Detektorbild und erreicht eine bessere Qualität als für Sigma.

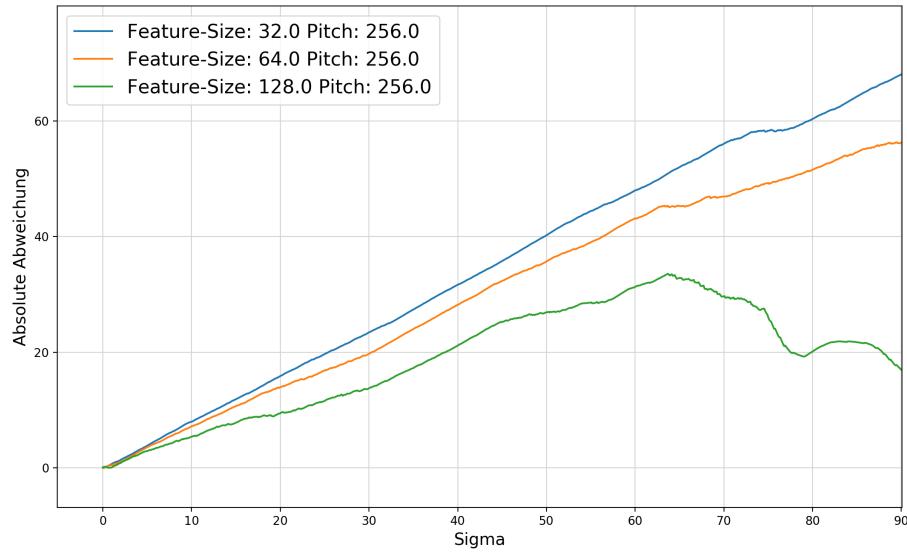


Abbildung 6.9: Absolute Abweichung der Sigma-Schätzung des Fully-Connected Neural Network (CNN) in Abhängigkeit von Feature-Size

6.3 Fazit

In dieser Arbeit wurde ein Prototyp entwickelt, der die Modellparameter der Gitterstruktur anhand der zugehörigen Detektorbild schätzen soll. Es wurde eine umfangreiche Datenanalyse vorgenommen, um die Effekte der Startparameter auf das Detektorbild und auf die Network-Schätzungen zu verstehen. Aus den Ergebnissen kann bisher die Schlussfolgerung gezogen werden, dass die in diesem Beleg benutzte Technik keine gute Qualität liefert, aber die Resultate für die weitere Arbeit an diesem Problem wertvoll sind. Nur der Startparameter Feature-Size konnte mit einer besseren Qualität geschätzt werden. Die Parameter Sigma und Pitch können nur mit einer unzureichenden Qualität geschätzt werden. Eine große Schwierigkeit sind die Effekte der Parameter auf das Detektorbild. Parameter wie Feature-Size und Sigma zeigen die gleichen Effekte auf das Detektorbild, nur unterschiedlich skaliert. Und auch der Parameter Pitch legt im Zusammenspiel mit dem Parameter Feature-Size die Grundstruktur des Detektorbildes fest. In dieser Arbeit konnten mehrere Schwachstellen der gewählten Architektur festgemacht werden. Der Pitch-Schätzer(CNN) kann nur mit lokalen Pattern umgehen und zeigt schwächen, wenn es globale Änderungen im Detektorbild gibt. Des Weiteren konnte festgestellt werden, dass der Parameter Feature-Size unabhängig des Parameters Pitch geschätzt werden kann. Auch die Generierung der Datenquelle hat Schwächen, welche während der Evaluierung deutlich geworden sind. Zum Beispiel existieren im Datensatz nur zwei Prozent der Detektorbildern einen höheren Sigma-Wert. Die Erkenntnisse dieser Arbeit können genutzt werden die Architektur zu verbessern und die Qualität der Schätzungen zu erhöhen.

6.4 Ausblick

Der in dieser entwickelte Prototyp hat in vielen Punkten Verbesserungspotential. Aus den Erkenntnissen dieser Arbeit sind bereits neue Ideen entstanden, wie man die Qualität des Prototyps noch steigern kann. Dabei ist ein möglicher Ansatz, dass die beiden Netze der Deep-Learning- Architektur gemeinsam trainiert werden und die Qualität für die Feature-Size und Sigma-Schätzung in das Training des Pitch-Schätzers einfließt. Bisher wurden beide Netzwerke getrennt trainiert und der Feature-Size und Sigma-Schätzer hat Impulsverteilungen eines festen Pitch-Schätzers bekommen. Ein weiterer Verbesserungsvorschlag ist die Modifizierung des Pitch-Schätzers. Die Auswirkung des Parameters Pitch auf das Detektorbild hat nicht nur lokale Auswirkungen, sondern auch globale Auswirkungen. Für den Pitch-Schätzer sollte deswegen eine Architektur gefunden werden, welche die globalen Änderungen des Detektorbildes besser erfassen kann bzw. mit deren Lokalität umgehen kann. Bisher wurde das Deep-Learning-Model mit einer eher geringen Datenmenge trainiert. Es sollte definitiv evaluiert werden, ob eine größere Datenmenge Einfluss auf die Qualität der Schätzer hat. Auch die Verteilung der Labels sollte überdacht werden. Ein letzter Verbesserungsvorschlag ist die Qualität der Simulation. Die gegebene Simulation nutzt eine niedrig aufgelöste Fourier-Transformation, welche wichtige Informationen verliert. Es sollte überprüft werden, ob ein Training des Deep-Learning-Models mit Detekorbildern mit höherer Auflösung eine höhere Qualität bei der Schätzung der Startparameter zur Folge hat. Zum Beispiel könnte ein weiteres Neural Network dazu genutzt werden die Auflösung der Fouriertransformation zu erhöhen. Im Verlauf dieser Arbeit konnten viele Erkenntnisse gesammelt werden und neue Ideen entwickelt werden. Diese Ideen müssen evaluiert auf ihre Machbarkeit geprüft werden.

Literaturverzeichnis

- [Bec] BECK, Fabian: *Backpropagation.* <http://www.neuronalesnetz.de/backpropagation1.html>. – Letzter Aufruf: 07.02.2019
- [Bro17] BROWNLEE, Jason: *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.* <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>, 2017. – Letzter Aufruf: 05.02.2019
- [Cro09] CROCE, Gianluca: *The Small Angle X-ray Scattering Technique: An Overview.* <http://people.unipmn.it/gcroce/download/theory.pdf>, 2009. – Letzter Aufruf: 04.02.2019
- [Des] DESHPANDE, Adit: *A Beginner's Guide To Understanding Convolutional Neural Networks.* <http://https://adeshpande3.github.io/A-Beginners-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2.html> – Letzter Aufruf: 05.02.2019
- [Fie82] FIENUP, J. R.: *Phase retrieval algorithms: a comparison.* 1982
- [Fra18] FRAZIER, Peter I.: A Tutorial on Bayesian Optimization. (2018)
- [GL94] GROPP, William ; LUSK, Ewing: An Introduction to MPI - Parallel Programming with the Message Passing Interface. (1994)
- [Hem00] HEMPEL, Thomas: *Mathematisch Grundlagen - Fourier-Integral / Fourier-Transformation.* http://www.uni-magdeburg.de/exph/mathe_gl/fourierintegral_beispiele.pdf, 2000
- [HZRS15] HE, Kaiming ; ZHANG, Xiangyu ; REN, Shaoqing ; SUN, Jian: Deep Residual Learning for Image Recognition. (2015)
- [Kah97] KAHAN, William: IEEE Standard 754 for Binary Floating-Point Arithmetic. (1997)
- [Kla13] KLAKOW, Dietrich: Grundlagen der Signalverarbeitung. (2013)
- [Klu18] KLUGE, Thomas: *Observation of ultrafast solid-density plasma dynamics using femtosecond X-ray pulses from a free-electron laser.* 2018
- [Kri07] KRIESEL, David: *Ein kleiner Überblick über Neuronale Netze.* 2007
- [LBH15] LECUN, Yann ; BENGIO, Y ; HINTON, Geoffrey: Deep Learning. (2015)
- [Leh06] LEHMANN, Peter: *Methoden der Messtechnik - Signal und Bildverarbeitung.* 2006
- [Myu02] MYUNG, In J.: Tutorial on maximum likelihood estimation. (2002)
- [Nav00] NAVÉ, R.: *Fraunhofer Diffraction.* <http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/mulslid.html>, 2000
- [ON15] O'SHEA, Keiron ; NASH, Ryan: *An Introduction to Convolutional Neural Networks.* 2015
- [QR11] QUIZA R., Davim J.: Computational Methods and Optimization. Machining of Hard Materials. (2011)
- [Rot06] ROTH, Ina: *Proseminar: Grundlagen Bildverarbeitung / Bild verstehen Orthogonale Funktionen.* 2006

- onstransformationen.* 2006
- [Rud16] RUDER, Sebastian: *An overview of gradient descent optimization algorithms.* 2016
- [Sch] SCHLESINGER, Dmitrij: *Bildverarbeitung: Fourier-Transformation.* http://www1.inf.tu-dresden.de/~ds24/lehre/bvme_ss_2012/bv_fourier.pdf, . – Letzter Aufruf: 07.02.2019
- [Tho18] THORMÜHLEN, Thorsten: Multimediale Signalverarbeitung, Bildverarbeitung: Filter. (2018)
- [Wik18] WIKIPEDIA: *Backpropagation — Wikipedia, The Free Encyclopedia.* <http://de.wikipedia.org/w/index.php?title=Backpropagation&oldid=181456626>, 2018. – Letzter Aufruf: 07.02.2019
- [Wis18] WISSENSCHAFT, Spektrum der: *Plasma.* <https://www.spektrum.de/lexikon/physik/plasma/11335>, 2018. – Letzter Aufruf: 07.02.2019
- [Zac17] ZACHARIAS, Malte: *Model-driven parameter reconstructions from Small Angle X-ray Scattering images.* 2017

Danksagung

Das Erstellen einer so umfangreichen Studienarbeit ist keineswegs nur die Leistung einer Person. Deswegen möchte ich mich bei all denjenigen bedanken, die mich bei der Erstellung dieses großen Belegs unterstützt haben.

Als Erstes möchte ich mich bei meinen beiden Betreuern Heide Meißner und Dmytro Schlesinger bedanken. Vielen Dank für die Geduld, konstruktive Kritik und vielen Ratschläge.

Des Weiteren möchte ich mich bei meinem Kollegen Max Glaser bedanken, der immer für eine fachliche Diskussion zur Verfügung stand und die Arbeit Korrektur gelesen hat.

Außerdem möchte ich mich bedanken bei:

- Michael Bussmann (Betreuung)
- Malte Zacharias (Hilfe bei der Simulation)
- Matthias Werner (Motivation)
- Thomas Kluge (Hilfestellung SAXS)
- Tobias Sebastian Hahn (Korrektur)

