

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. Dmitrij Schlesinger(TU Dresden), Dr. Heide Meißner(HZDR),
Dr. Michael Bussmann(HZDR)

Dresden, 22. Februar 2019

Aufgabenstellung

aufgabenstellung

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 22. Februar 2019

Patrick Stiller

Inhaltsverzeichnis

1	Einleitung	3
2	Grundlagen	4
2.1	Physikalischer Hintergrund	4
2.1.1	Kleinwinkelstreuung	4
2.1.2	Experimentbeschreibung	4
2.2	Problemidentifikation	6
2.3	Fouriertransformation	7
2.4	Filter	8
2.5	Neuronal Networks	9
2.5.1	Aufbau	9
2.5.2	Feed-Forward Neuronal Networks	10
2.5.3	Lernprozess	11
2.5.4	Convolutional Neuronal Networks	12
2.5.5	Residual Strukturen	13
3	Simulation	15
3.1	Motivation	15
3.2	Simulationsbeschreibung	15
3.2.1	Design der Gitterstruktur und Simulation des Hochenergielasereinflusses	15
3.2.2	Effekt der Startparameter auf die Gitterstruktur	16
3.2.3	Simulation der Streuung	17
3.2.4	Auswirkung der Startparameter auf das Simulationsergebnis	18
3.2.4.1	Fraunhofer-Beugung	20
3.2.4.2	Effekt von Pitch	21
3.2.4.3	Effekt von Feature-Size	21
3.2.4.4	Effekt von Sigma	22

4	Datenbasis	23
4.1	Generator	23
4.2	Wahl der Eingangsparameter	24
4.3	Performance	25
4.4	Trainings-, Validierungs- und Testdatensatz	25
5	Lösungsansatz	27
5.1	Gesamtarchitektur	27
5.2	Convolutional Neuronal Network	28
5.2.1	Bayes'sche Optimierung	28
5.2.2	Architektur	29
5.3	Fully-Connected Neuronal Network	30
5.4	Training	31
6	Ergebnisse und Diskussion	32
6.1	Trainingsprozess	32
6.1.1	Convolutional Neuronal Network	32
6.1.2	Fully-Connected Neuronal Network	33
6.1.2.1	CNN-Input	33
6.1.2.2	Label-Input	34
6.2	Evaluierung	34
6.2.1	Convolutional Neuronal Network	34
6.2.2	Fully Connected Neuronal Network	35
6.3	Fazit	35
6.4	Ausblick	35
	Literaturverzeichnis	36

1 Einleitung

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 physikalische Eigenschaften eines untersuchten Feststoffes liefern [Cro]. 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 [KRM⁺18]. Bisher konnte die Plasmadynamik, welche bei der Interaktion eines Feststoffes und eines Hochintensitätslasers entsteht, nur durch Simulationen erfasst werden mit Hilfe von SAXS gibt es neue Möglichkeiten, die Laser-Plasma-Interaktion auf einem Feststoff 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 numerischen Simulationen Aussagen über die Elektronendynamik im Plasma zu treffen [KRM⁺18].

2.1.2 Experimentbeschreibung

Das in [KRM⁺18] beschriebene Experiment besteht aus vier Hauptkomponenten (siehe Abbildung 2.1 links): Dem Target (Mitte), dem Hochintensitätslaser (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 Target mit einer 2 μm breiten Siliziumschicht überzogen. Durch die eingravierte Gitterstruktur besitzt das Target einen eindimensionalen

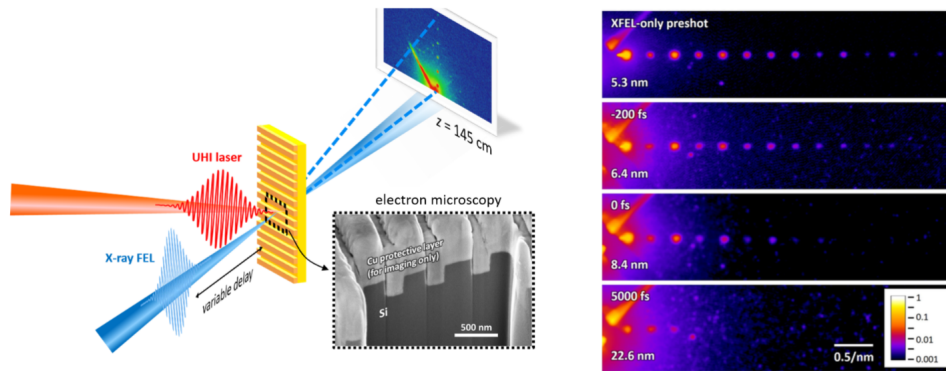


Abbildung 2.1: Links: Schematischer Aufbau des Experimentaufbaus. Rechts: der Abbildung sind Detektorbilder in Abhängigkeit der Bestrahlungsdauer des Hochintensitätslasers dargestellt.
Bildquelle: [KRM⁺18]

Informationsgehalt und kann somit durch drei Parameter vollständig beschrieben werden. Die drei Parameter sind Pitch, Feature-Size und die Aufweichungsbreite σ . Dabei beschreiben Pitch und Feature-Size die Struktur des Targets und 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 Elektronendichteverteilung η verwendet, welcher oft im Zusammenhang des SAXS-Ansatzes fällt. Der Begriff Elektronendichteverteilung η wird in dieser Arbeit als Synonym für das Grating verwendet.

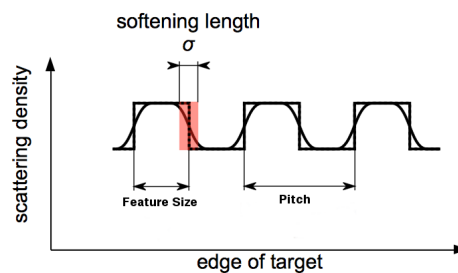


Abbildung 2.2: Analytische Beschreibung des Querschnittes des Targets. Dabei ist das Grating zu erkennen und dass es durch drei Parameter: Pitch, Feature-Size und Sigma beschrieben werden kann. Bildquelle: [Zac17]

Während des Experiments wird das Target durch einen Hochintensitätslasers 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 [Wis]. Während des Beschusses durch den Hochintensitätslasers ist außerdem ein Schmelzprozess und somit eine Aufweichung der Gitterstruktur des Targets wahrzunehmen (siehe Abbildung 2.3). Das durch den Hochintensitätslasers 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).

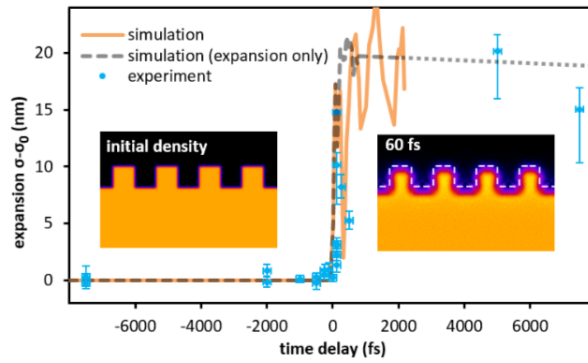


Abbildung 2.3: Veränderung des Gratings nach einer Bestrahlungsdauer von 60 Femtosekunden Bildquelle: [KRM⁺18]

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.

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)) [KRM⁺18].

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

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 Elektronendichte-

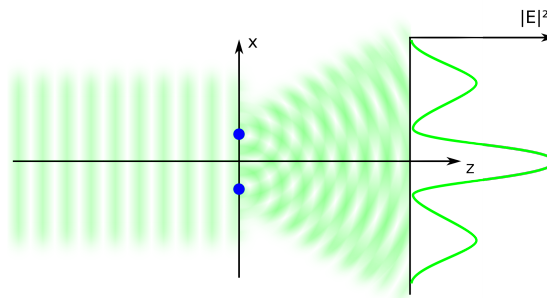


Abbildung 2.4: Beispielhafte Darstellung einer Streuung an zwei Elektronen. Die Röntgenlaserpulse treffen geradlinig auf die Elektronen und werden dann bei den Elektronen gestreut. Der Detektor dahinter misst die ankommenden Intensitäten der kreisförmigen Wellen und summiert diese über die Zeit. Bildquelle: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

verteilung η mit Hilfe einer inversen Fouriertransformation (IFT) 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 garantiert ist und zweitens sehr viele Iterationen notwendig sind um eine Optimierung der errechneten Phase zu erzeugen. Diese Eigenschaften führen dazu, dass eine hochfrequente Verarbeitung von Experimentergebnissen verwehrt bleibt. Deswegen soll mit Hilfe von Deep Learning die Charakterisierung der Elektronendichteverteilung η (Parameter zur Beschreibung der Gitterstruktur) rekonstruiert werden. Neuronale Netze haben den Vorteil, dass sie sich im Gegensatz zu iterativen Verfahren parallelisieren und durch GPUs beschleunigen lassen und die Parameter der Elektronendichtenverteilung nicht-iterativ bestimmt werden (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 (Abbildung ??). 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 in ihren Kosinus- und Sinusanteilen 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 Amplituden- $|F(u)|$ 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 $(f * g)$ 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 Neuronal Networks

Neuronal 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 neuronale Netze wurden bereits 1943 von Warren McCulloch und Walter Pitts entwickelt [Kri07]. Heute sind Neuronale Netze der Schwerpunkt des Machine Learnings und werden auf großen Datenmengen angewendet, um ein gewünschtes Verhalten zu trainieren.

2.5.1 Aufbau

Ein Neuronal 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 Neuronal Network als ein Tripel (N, V, w) mit den 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 Neuronalen Netzes. 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 Neuronalen Netz 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 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 neuronalen Netzes erleichtern. Für diesen Beleg ist die Rectified Linear Units- Aktivierungsfunktion (Formel (2.8)) relevant.

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

Die ReLu-Aktivierungsfunktion ist im Vergleich 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 Neuronalen Netz gegeben ist.

2.5.2 Feed-Forward Neuronal Networks

Für ein Neuronal Network können verschiedene Netzwerktopologien werden. Ein Beispiel dafür sind Feed Forward Neuronal Networks (Abbildung 2.5). Bei einem Feed Forward Neuronal 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 Schicht, 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 Neuronal 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 Neuronal Networks sind die Fully-Connected Layer. Bei einem Fully-Connected Layer ist jedes Neuron des Fullyconnected Layer mit jedem Neuron des vorherigen Layers verbunden. In Abbildung 2.5 sind die Verbindungen eines Fully-Connected Layer dargestellt.

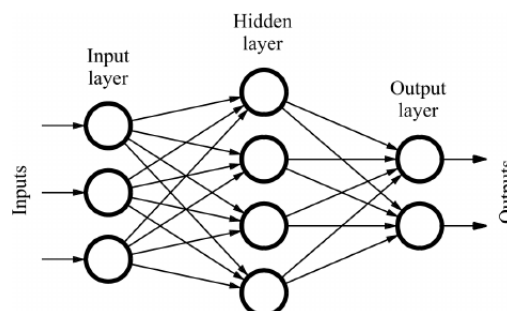


Abbildung 2.5: Beispielhafte Darstellung eines Feed-Forward Neuronal Networks. Bildquelle: [QR11]

2.5.3 Lernprozess

Das Anlernen von Neuronalen Netzen 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 Neuronalen Netzen aus zwei Teilen. Zu jedem Netzinput x_i gibt es ein zugehöriges Label y_i , welches den gewünschten Netzoutput definiert. Der Lernprozess eines neuronalen Netzes 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 Neuronalen Netzes gegeben. Das Neuronale Netz bestimmt dann durch die Rechenvorschriften des Neuronalen Netzes den Netz-Output \hat{y}_i . Im zweiten Schritt wird die Qualität des Netz-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 Errorfunktion 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 :

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

Dabei ist E die Errorfunktion, δ_j des Gradient Neurons j , o_i die Ausgabe des Neurons i . Der Parameter η , welcher die Learning-Rate des Gradientenabstiegsverfahren 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.

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

Dabei ist die Gleichung (2.11) in Abhängigkeit von den Variablen o_j , dem Output des Neurons j , t_j , die Soll-Ausgabe 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 Neuronal Networks

Klassische Neuronal Networks besitzen die Einschränkung, dass deren Inputs als Vektoren geliefert werden müssen. Soll ein klassisches Neuronal 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 Neuronal Networks (CNNs) besitzen im Gegensatz zu klassischen Neuronal 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 Layer-Output 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. Sollte 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 als dritte Möglichkeit, dass die fehlenden Datenpunkte mit dem dazugehörigen Randwert ersetzt wird [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 Neuronal 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 2.7).

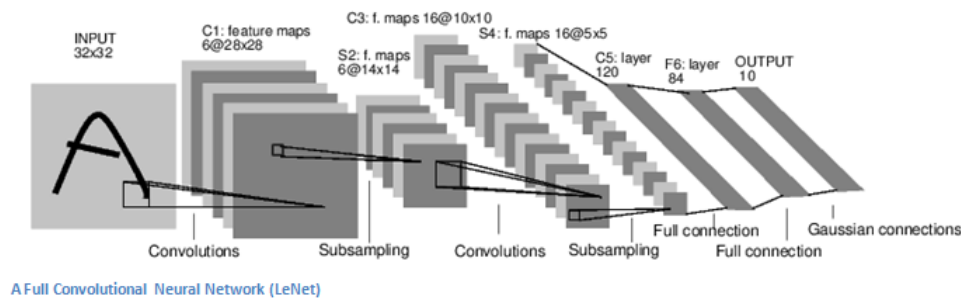


Abbildung 2.6: Beispielhafte Darstellung eines Convolutional Neuronal Networks. Bildquelle: [Des]

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 (H) von 1 besitzen und der Filter nur entlang einer Dimension verschoben werden.

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 dessen 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. 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].

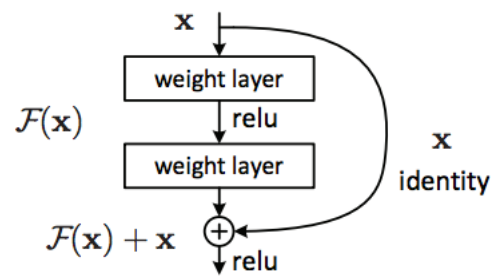


Abbildung 2.7: Realisierung einer Skip-Connection in einem Neuronal Network. Bildquelle: [HZRS15]

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. Das ganze Kapitel bezieht sich auf die Master-Arbeit von Malte Zacharias [Zac17].

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 Targets 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 Hochenergieintensitätslasers.

Der erste Schritt der Simulation ist die Modellierung eines Features mit Hochintensitätslasereinfluss. Dazu wird eine Rechteckfunktion ($1_{[0, \text{Feature-Size}]}$), welche die Breite eines Features festlegt mit einer Gaussverteilung ($\exp(-x^2/2\sigma^2)$) gefaltet [KRM⁺18]. Das Ergebnis der Faltung wird mit Hilfe der Errorfunktion (Gleichung (3.1)) dargestellt.

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

Das Ergebnis der Faltung des Rechteckimpulses und der Gauss-Verteilung ist die Modellierung eines Features:

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

Das modellierte Feature wird durch eine weitere Faltung mit mehreren um Pitch verschobenen Impulsen[Beu11] 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.

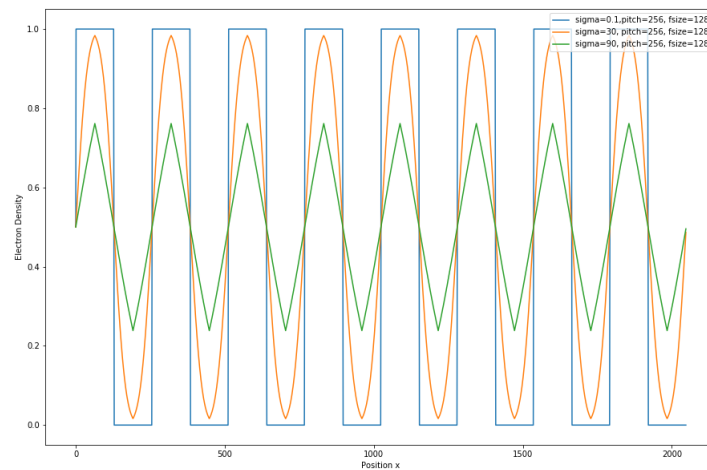


Abbildung 3.1: Vergleich der Kantenstruktur unter variierenden Sigma-Werten bei gleichbleibenden Pitch- und Feature-Size- Werten

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.

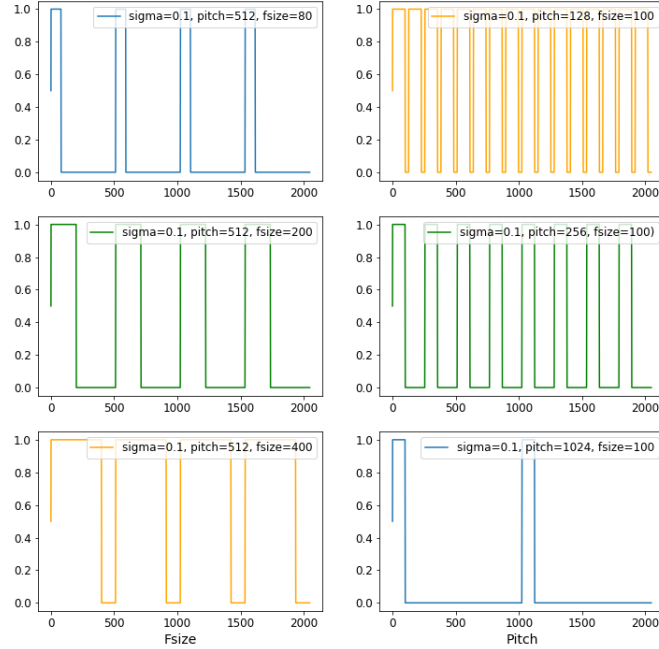


Abbildung 3.2: Vergleich der Kantenstruktur unter variierenden Pitch-Werten (rechte Spalte) und variierenden Feature-Size-Werten (linke Spalte)

3.2.3 Simulation der Streuung

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

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

Das entstandene Produkt der Fouriertransformation ist im Raum der Komplexen Zahlen und weist den gleichen Informationsgehalt wie die Elektronendichteverteilung 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 wird die Phaseninformation im weiteren Simulations-

verlauf nicht weiter verwendet. Um schlussendlich die aufgenommenen Intensitäten des Detektors (siehe Abbildung 3.3) zu erhalten, wird das Betragsquadrat des Amplitudenspektrums gebildet ($\Phi = |\hat{a}_k|^2$)

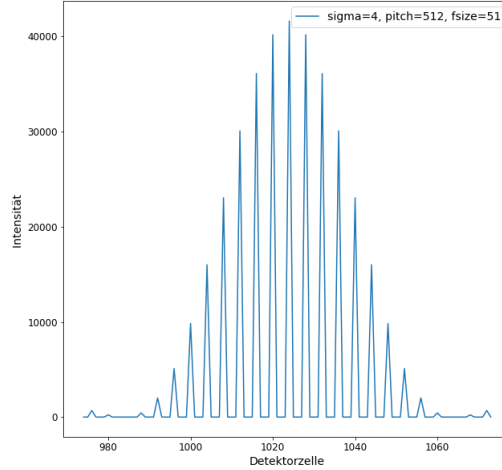


Abbildung 3.3: Resultierendes Detektorbild als Endprodukt der Simulation

3.2.4 Auswirkung der Startparameter auf das Simulationsergebnis

Die Startparameter der Simulation haben ebenfalls Auswirkungen auf das Endprodukt der Simulation. Betrachtet man zunächst die Auswirkungen des Startparameters Pitch (Abbildung 3.4), ist zu erkennen, dass mit steigendem Pitch sich das Intensitätsmaximum verringert. Zusätzlich kommen noch mehr Peaks hinzu. Das hängt damit zusammen, dass sich mit steigendem Pitch die Anzahl der Features verringert und somit mehr Frequenzen benötigt werden, um die Gitterstruktur zu modellieren.

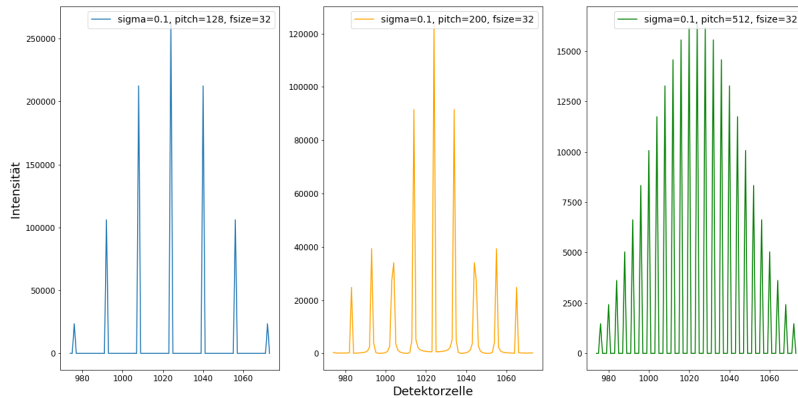


Abbildung 3.4: Verhalten der Detektoraufnahme in Abhängigkeit des Parameters Pitch

Der Startparameter Feature-Size verhält sich invers zum Parameter Pitch. Mit steigendem Feature-Size Wert kommt es zu einer Erhöhung des Intensitätsmaximums und es werden weniger Frequenzen benötigt, um die Gitterstruktur zu modellieren, somit sind auch weniger Peaks im Detektorsignal vorhanden (Abbildung 3.5).

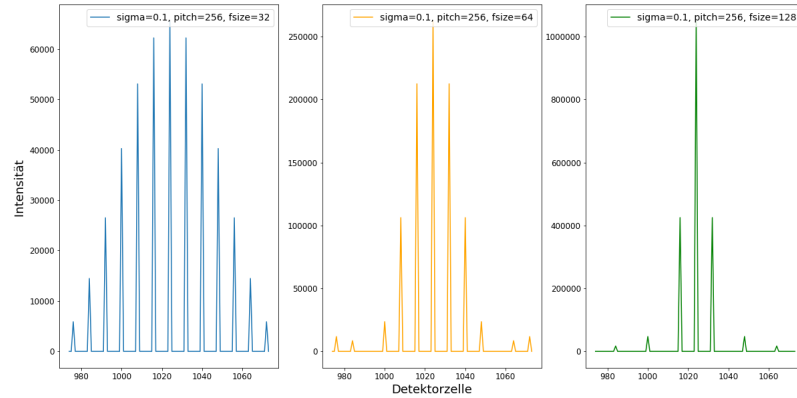


Abbildung 3.5: Verhalten der Detektoraufnahme in Abhängigkeit des Parameters Feature-Size

Der letzte Startparameter Sigma hat mit steigenden Wert ähnliche Auswirkungen auf die Detektoraufnahmen wie Feature-Size. Bei steigendem Sigma-Wert steigt das Intensitätsmaximum und die Anzahl der Peaks verringert sich (Abbildung 3.6). Jedoch skaliert der Parameter Sigma anders als Feature-Size. Kleinere Unterschiede im Sigma-Wert können größere 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.

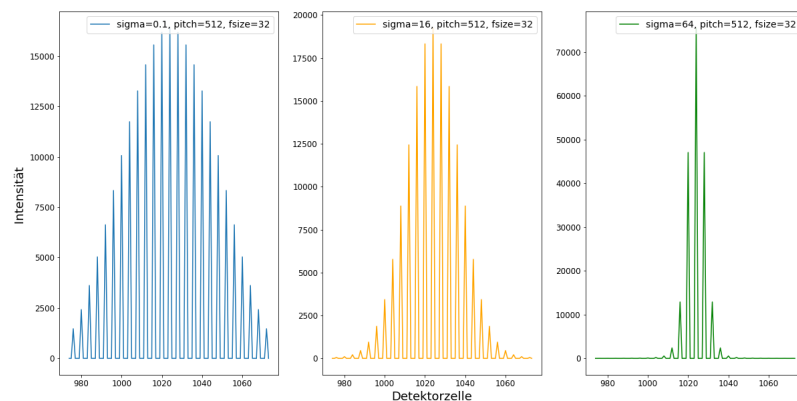


Abbildung 3.6: Verhalten der Detektoraufnahme in Abhängigkeit des Parameters Sigma

3.2.4.1 Fraunhofer-Beugung

Grundlage für die Simulation und des Experimentes ist die Fraunhofer-Beugung. Die Fraunhofer-Beugung ermöglicht die Berechnung des Intensitätswertes an jedem Punkt des Detektors. Dabei geht die Fraunhofer-Beugung von einer idealen Lichtquelle und von einem kontinuierlichen Detektor aus (unendlich viele Abtastpunkte des Intensitätssignals), jedoch sind die Anzahl der Detektorzellen im Experiment sowie die Anzahl der Pixel in der Simulation limitiert, deshalb können die Simulation und das Experiment als diskrete Abtastung der Fraunhofer-Beugung betrachtet werden. In diesem Kapitel wird die Fraunhofer-Beugung dazu genutzt die Effekte der Startparameter, welche im vorherigen Kapitel dargestellt wurden, physikalisch zu begründen. Bei der Fraunhofer-Beugung werden ein oder mehrere Öffnungen in einem undurchsichtigen Schirm mit einer ebenen Lichtwelle beleuchtet. Hinter dem Spalt liegt eine Bildebene, um die Beugungseffekte zu beobachten [OUO18]. In Abbildung 3.7 ist eine vereinfachte Darstellung des Experimentaufbaus der Fraunhofer-Beugung zu sehen.

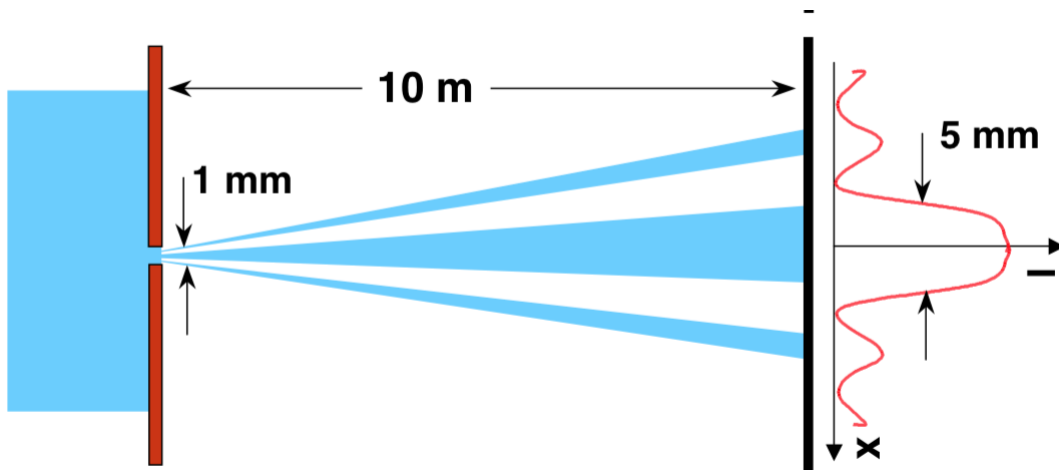


Abbildung 3.7: Vereinfachte Darstellung der Fraunhofer-Beugung. Bildquelle: [Gro03]

Die Assoziation zwischen dem Experiment und der Fraunhofer-Beugung liegt darin, dass die Beugung am Spalt nahezu die selben Effekte aufweist, wie die Streuung des Röntgenlasers an den Features des Gratings. Wird die Anzahl der Spalte N erhöht, ist dies mit einer Senkung des Pitch-Wertes gleichzusetzen. Wird die Spaltbreite a erhöht, ist dies mit einer Erhöhung des Feature-Size-Wertes gleichzusetzen. Der funktionale Zusammenhang für die N-Spalten Fraunhofer-Beugung ist in Formel 3.4 dargestellt. Dabei kann für jeden Detektorpunkt θ die ankommende Intensität berechnet werden. Weitere Parameter der Formel 3.4 sind λ die Wellenlänge des Lichtes und I_0 ein Skalierungsfaktor für die maximal zu messende Intensität. Beide Parameter sind für die Erklärung der Effekte der Startparameter irrelevant.

$$I(\theta) = I_0 \left(\frac{\sin \beta}{\beta} \right)^2 \left(\frac{\sin N\alpha}{\sin \alpha} \right)^2 \quad \text{mit } \alpha \equiv (\pi a / \lambda) \sin \theta \text{ und } \beta = (\pi b / \lambda) \sin \theta \quad (3.4)$$

3.2.4.2 Effekt von Pitch

In Abbildung 3.8 ist die Bildebene der Fraunhofer-Beugung bei unterschiedlicher Spaltanzahl zu sehen. Erhöht man die Spaltanzahl, ist zu erkennen, dass sich das Bild, welches bei einem Spalt zu sehen ist, bei mehreren Spalten auf mehrere Peaks aufteilt. Dieser Effekt ist auch bei der Senkung des Pitch-Wertes zu sehen (Abbildung 3.4). Dennoch gibt einen großen Unterschied zur Simulation. Ein wichtiger Unterschied ist, dass Aufgrund der Diskretisierung des Detektorsignals, kleinere Peaks zwischen den größeren Peaks verloren gehen. Diese kleineren Peaks erzeugen Minima. Anhand der Anzahl der Minima M kann die Anzahl der Spalten im Target abgelesen werden. Der Zusammenhang ist dass die Anzahl der Spalten N durch $M + 1$ errechnen lässt. Befinden sich zwischen den größeren Peaks 4 Minima, hat das Target 5 Spalten [Nav00].

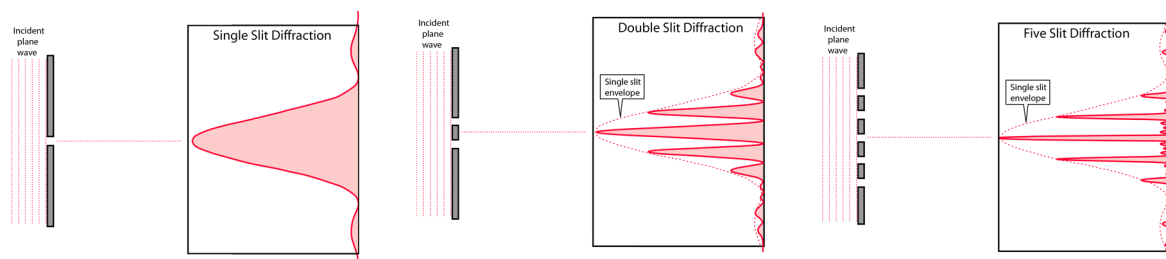


Abbildung 3.8: Vergleich der Bildebene bei unterschiedlich Anzahl von Spalten. Bildquelle: [Nav00]

3.2.4.3 Effekt von Feature-Size

Erhöht man die Größe des Spalts, kommt es zu geringerer Beugung am Spalt bzw. das Licht kann den Spalt geradliniger durchdringen. Das hat zur Folge, dass weniger Licht die Ränder der Bildebene erreicht und somit weniger Intensität am Rand gemessen werden kann (Siehe Abbildung 3.9). Dieser Effekt ist auch bei der Erhöhung des Feature-Size-Wertes in der Simulation erkennen (Siehe Abbildung 3.5). Die Peaks im Zentrum des Detektors werden verstärkt und die Detektorzellen am Rand bekommen weniger Lichtintensität bzw. gar keine Lichtintensität ab [Heb11].

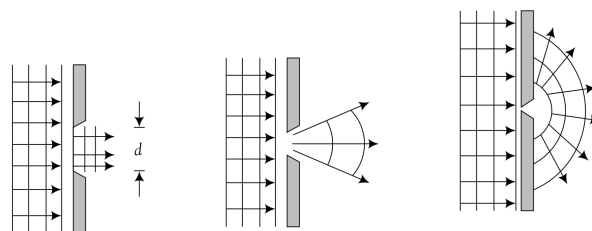


Abbildung 3.9: Stärke der Beugung in Abhängigkeit der Spaltgröße. Bildquelle: [Heb11]

3.2.4.4 Effekt von Sigma

Wie in Kapitel 3 beschrieben, ergibt sich die Gitterstruktur des Gratings aus einer Faltung aus Gauss-Verteilung, Impulsverteilung und Rechteckimpuls. Diese Faltung wird dann fouriertransformiert um das Detektorbild. 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 (3.5))[Rot06] .

$$F(f * g) = F(f) \cdot F(g) \quad (3.5)$$

Das Fourier-Faltungs-Theorem kann dazu genutzt werden, den Effekt von Sigma auf das Detektorbild zu erklären. In Abbildung 3.10 sind verschiedene Gauss-Verteilungen (linke Seite) und deren zugehörige Fouriertransformationen (rechte Seite) dargestellt. Hinweis: Alle Werte im FFT-Bild sind über 0. Aufgrund des Fourier-Faltungstheorems kommt es zu einer Multiplikation mit den Gaussverteilungen. Die Multiplikation stärkt die Intensitäten im zentralen Bereich und glättet die Intensitäten am Rand. Dieser Effekt ist ebenfalls in der Simulation zu beobachten (siehe Abbildung 3.6).

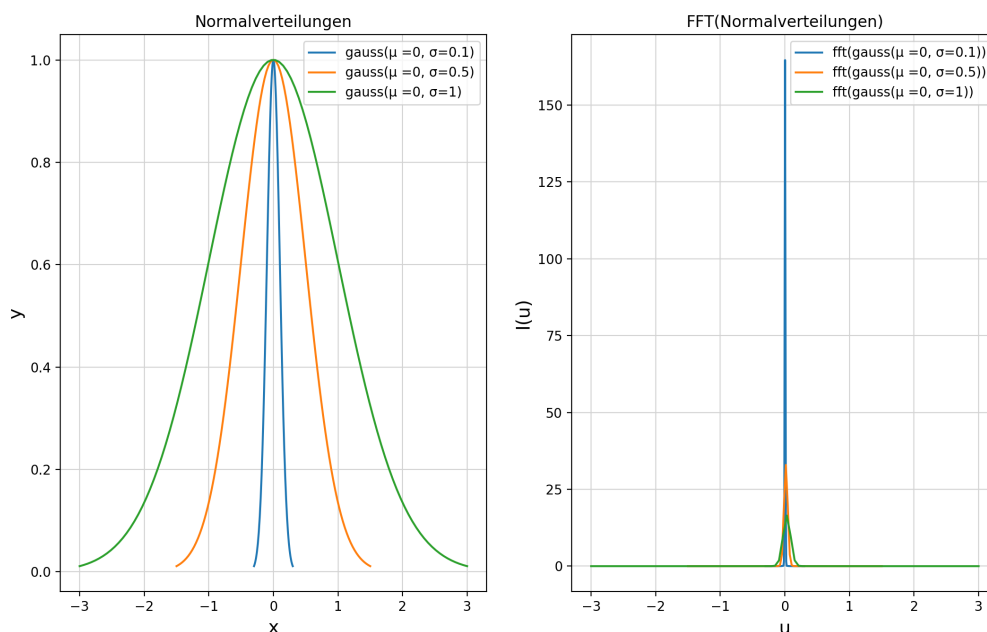


Abbildung 3.10: Linke Seite: Gauss-Verteilungen mit unterschiedlichen Sigma-Werten Rechte Seite: Fouriertransformationen der Gauss-Verteilungen

4 Datenbasis

4.1 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 Testdaten produzieren kann. Der erste Schritt der Genierierung der Datenbasis ist die Festlegung der Menge P von Startparametern, wobei ein Startparamter 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) parallisiert. 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 Neuronal Networks vorgesehen sind, gespeichert.

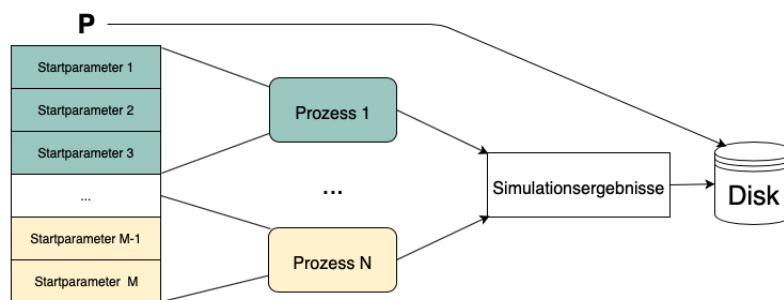


Abbildung 4.1: Schematischer Aufbau des Generators

4.2 Wahl der Eingangsparameter

Anhand der Auswertung der Simulation in Kapitel 3 sind Abhängigkeiten zwischen den Startparametern Pitch, Feature-Size und Sigma festzustellen. Um einen validen Datensatz zu erstellen, müssen diese Abhängigkeiten berücksichtigt werden. Der erste Startparameter Pitch ist der Parameter, welcher den Wertebereich 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 maximaler 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 Simulationsergebnisse ohne Hochintensitätslasereinfluss zu generieren. Zur Bestimmung des maximalen Sigma-Wertes ist der Abstand zwischen zwei Features notwendig. Der Abstand zwischen zwei Features ergibt sich aus Pitch - FeatureSize. 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 4.2 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 4.2: Umsetzung der Definitionsbereiche mit Hilfe drei verschachtelter Schleifen, welche voneinander Abhängig sind

4.3 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 4.1 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 sek	187,613 sek	1870,095 sek	186940,286 sek *
Generator	1,917 sek	4,531 sek	42,831 sek	6900,243 sek

Tabelle 4.1: Vergleich der Berechnungsdauer für verschieden große Datensätze

4.4 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 Neuronal 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 Neuronal 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 4.2) eine ausreichend große Verteilung von Sigma und Feature-Size-Werten für das Training von Pitch zur Verfügung (Tabelle 4.2).

Dataset	Anzahl Pitch	Anzahl Feature-Size	Anzahl Sigma
Pitch	368	23	23
Feature-Size und Sigma	46	46	46

Tabelle 4.2: Abtastwerte für die jeweiligen Startparameter. Diese Werte werden wie in Abbildung 4.2 verarbeitet

Der Datensatz für das Training des Neuronal Networks weißt eine andere Verteilung wie der Pitch-Datensatz auf. In diesem Datensatz wurden die Folgen für Pitch, Sigma und Feature-Size gleich ab-

getastet, jedoch ist die Verteilung der Sigma- und Feature-Size-Werte aufgrund der Implementierung der Parametergenerierung deutlich ausgeprägter. In der Tabelle 4.2 sind die Abtastwerte der jeweiligen Folgen zu sehen. Für das Training, die Validierung des Trainingsprozesses und die Evaluierung des Neuronal Networks wurde der jeweilige Datensatz geschuffled und in Trainings, Validation und Test- Datensatz aufgespaltet. Um die Länge nicht zu groß zu gestalten, wurde der Trainingsdatensatz bei beiden Datensätzen auf 16384 Detektorbilder beschränkt. Für die Validierung stehen jedem Neuronal Network 64 Bilder zur Verfügung. Die restlichen Bilder wurde komplett für eine aussagekräftige Evaluierung verwendet. In Tabelle 4.3 ist die Verteilung der Samples auf Trainings-, Validierungs- und Testdatensatz zu sehen. Aufgrund der höheren Abtastung von Pitch im Pitch-Datensatz (siehe Tabelle 4.2) besitzt dieser mehr Detektorbilder und somit mehr Samples zum evaluieren.

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

Tabelle 4.3: Verteilung der Samples auf Trainings, Validierungs- und Testdatensatz

5 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 Neuronal Networks und deren Training eingegangen.

5.1 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 Parameterkonstruktion als Zweischrittverfahren konstruiert. Die Architektur besteht aus zwei Neuronal Networks. Das erste Neuronal Network ist ein Convolutional Neuronal 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:

$$\delta_{pitch}(x) = \begin{cases} 1 & \text{für } x \bmod pitch = 0 \\ 0 & \text{sonst} \end{cases} \quad (5.1)$$

Die generierte Impulsverteilung dient als Support für das zweite Neuronal Network (Fully Connected Neuronal 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 für das Fully Connected Neuronal Network gegeben. In Abbildung 5.1 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 Neuronal Networks. Aufgrund lokaler Änderung im Detektorbild bei verschiedenen Pitch-Werten wurde für die Schätzung des Pitch-Wertes ein CNN gewählt. Das Gegenteil ist bei unterschiedlichen Feature-Size- und Sigma-Werten der Fall, diese haben globale Auswirkung auf das Detektorbild. Deshalb wurde für

die Schätzung des Feature-Size und Sigma-Wertes ein Fully Connected Neuronal Network gewählt.

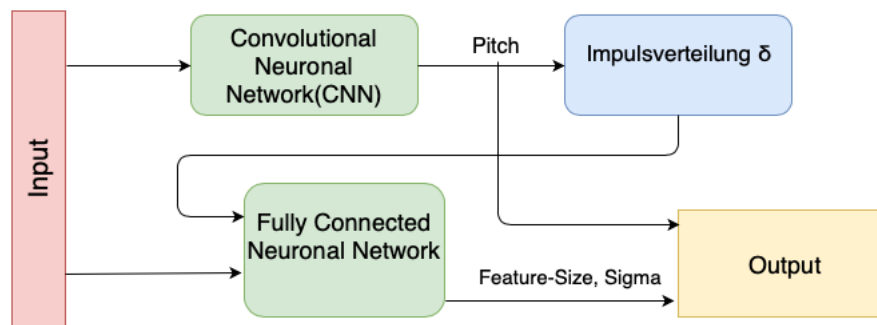


Abbildung 5.1: Schematische Beschreibung der Gesamtarchitektur

5.2 Convolutional Neuronal Network

5.2.1 Bayes'sche Optimierung

Das Finden der geeigneten Hyperparameter für ein Neuronales Netz 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 Wahrscheinlichkeitsverteilung $p(L|\theta)$ mit L der Qualität eines Neuronal Networks, dass in Abhängigkeit eines Hyperparametersatzes 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 bestimmte Anzahl von Iterationen erreicht wurde. Die genauere Beschreibung des Algorithmus kann [Fra18] entnommen. Um eine Bayes'sche Optimierung für das CNN vorzunehmen, wurde die Implementierung von Scikit-learn verwendet. Dabei wurden die Hyperparameter: Anzahl der Convolutional Layer,

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 5.1: Definition des Suchraums für die Bayes'sche Optimierung

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 5.1 eingeschränkt. 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 Validationdatensatz verwendet. Insgesamt wurden 50 Evaluierungsschritte ausgeführt. Zusätzlich wurden Residual Strukturen mit einer Sprungweite von 2 dem erstellten CNN hinzugefügt.

5.2.2 Architektur

Das Ergebnis der Bayes'schen Optimierung ist ein CNN mit einem Block von Convolutional Layer. 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 lineare 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 5.2 ist die komplette Architektur des CNN's 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 5.2: Architektur des Convolutionl Neuronal Networks

5.3 Fully-Connected Neuronal Network

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

5.4 Training

Das Training der beiden beschriebenen Neuronal Networks erfolgt getrennt. Als erstes wird der Schätzer des Pitch-Wertes trainiert(CNN), da dieser die Grundlage für das Training des Fully-Connected Neuronal 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 Neuronal 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 Neuronal Networks nicht verändern.

Für das Training beider Neuronal Networks wurde der Adam-Optimizer verwendet. Der Adam-Optimizer ist eine Erweiterung des klassischen Gradientenabstiegsverfahren. Der Adam-Optimizer verknüpft die Vorteile zwei 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 Neuronal Network steht wie in Kapitel 4.4 eine eigene Datenbasis zu Verfügung, welche mit Hilfe von Batch-Processing trainiert wurde. Beim Batch-Processing werden dem Neuronal 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 Netzwerk 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 Neuronal Networks wurde MSE (2.9) als Errorfunktion verwendet.

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

6 Ergebnisse und Diskussion

6.1 Trainingsprozess

6.1.1 Convolutional Neuronal Network

Der Trainingsverlauf des Convolutional Neuronal Networks 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.

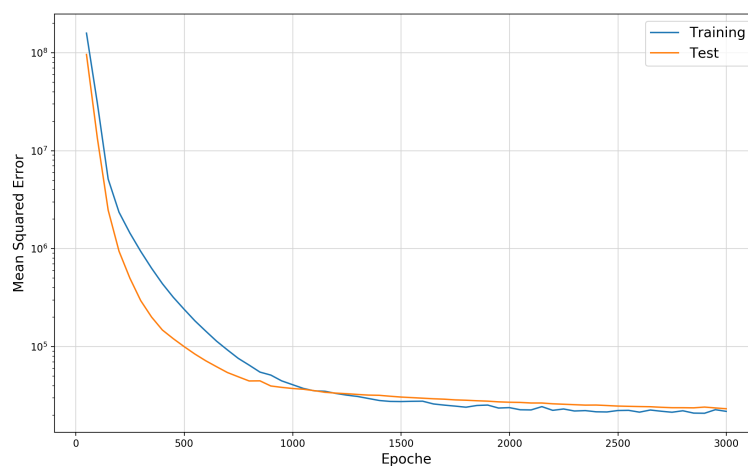


Abbildung 6.1: Verlauf des Trainings- und Validierungsfehlers des CNNs in Abhängigkeit der Epoche

In Abbildung 6.1 sieht man 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. Während des Trainings konnte ein minimaler Trainingsfehler von 20910.45 und ein minimaler Validierungsfehler 23247.19 von erreicht werden.

6.1.2 Fully-Connected Neuronal Network

In Kapitel 5 wurde beschrieben, dass das Fully-Connected Neuronal Network auf zwei verschiedene Weisen trainiert wurde. Im ersten Experiment wurde das Fully-Connected Neuronal 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 Epochenanzahl resultiert aus dem schnelleren Konvergenzverhalten des Fully-Connected Neuronal Networks. Für das Fully-Connected Neuronal Network wurde ebenfalls Mean Squared Error als Qualitätsmaß verwendet.

6.1.2.1 CNN-Input

In Abbildung 6.2 ist die Lernkurve des ersten Experimentes dargestellt. Während des Trainings konnte ein minimaler Trainingsfehler von 75.80 und ein minimaler Validierungsfehler 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 Neuronal Networks auf den Trainingsdaten deutlich höher als die der Validierungsdaten.

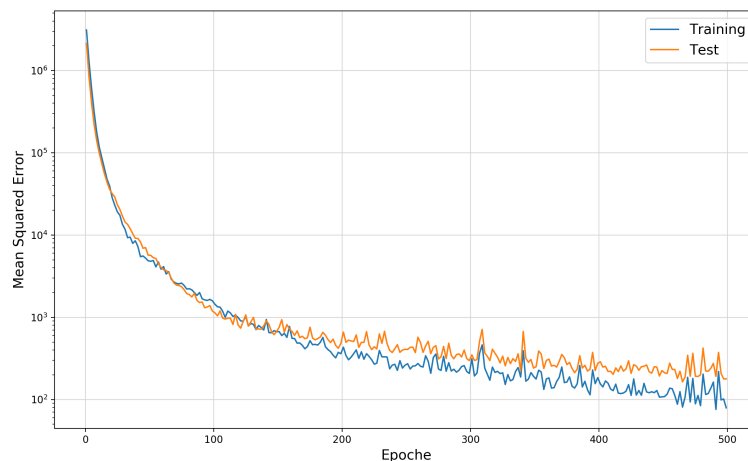


Abbildung 6.2: Verlauf des Trainings- und Validierungsfehlers in Abhängigkeit der Epoche im ersten Experiment

6.1.2.2 Label-Input

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.

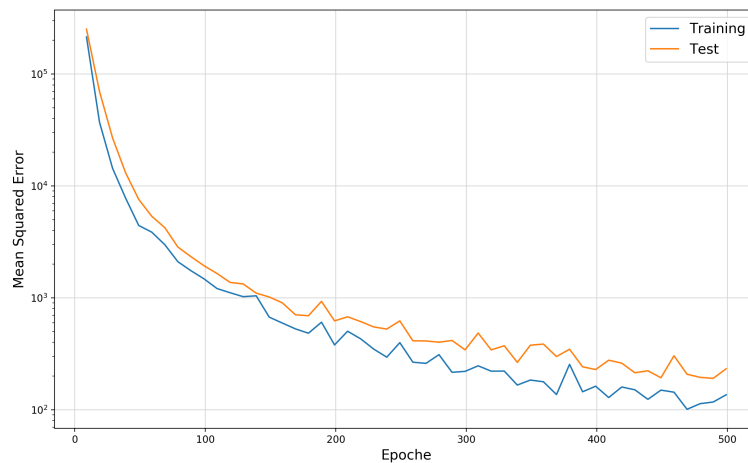


Abbildung 6.3: Verlauf des Trainings- und Validierungsfehlers in Abhängigkeit der Epoche im zweiten Experiment

6.2 Evaluierung

6.2.1 Convolutional Neuronal Network

Um die Qualität des CNNs zu evaluieren, wurden die nicht trainierten Samples des Testdatensatzes verwendet. Für die Qualitätsanalyse wurde die Absolute Abweichung zwischen dem Pitch-Label und der Pitch-Schätzung in Abhängigkeit der Parameter Sigma und Feature-Size dargestellt. Der erste Plot (Abbildung) wurde die Abhängigkeit der Absoluten Abweichung zwischen dem Pitch-Label und der Pitch-Schätzung in Abhängigkeit des Startparameters Feature-Size untersucht.

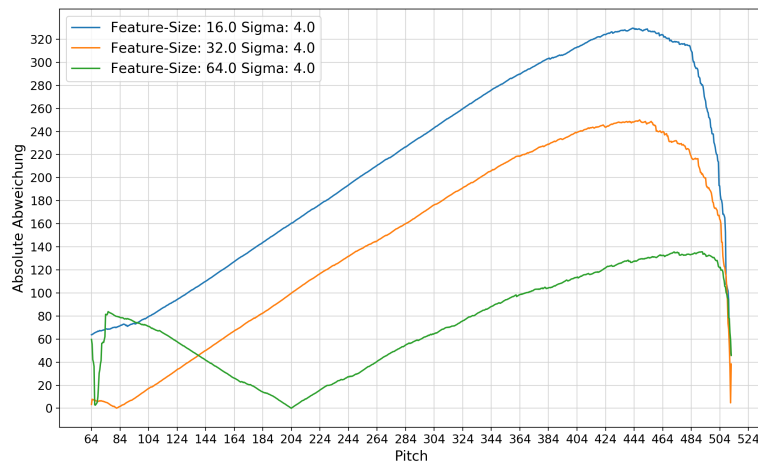


Abbildung 6.4: Qualität der Pitch-Schätzung in Abhängigkeit des Startparameters Feature-Size

6.2.2 Fully Connected Neuronal Network

6.3 Fazit

6.4 Ausblick

Literaturverzeichnis

- [Bec] BECK, Fabian: *Backpropagation*. <http://www.neuronalesnetz.de/backpropagation1.html>
- [Beu11] BEUCHER, Ottmar: *Übungsbuch Signale und Systeme Lösungsband zum Lehrbuch Signale und Systeme - Theorie, Simulation, Anwendung*. 2011
- [Bro17] BROWNLEE, Jason: *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>, 2017. – Accessed: 2019-02-11
- [Cro] CROCE, Gianluca: *The Small Angle X-ray Scattering Technique: An Overview*. – <http://people.unipmn.it/gcroce/download/theory.pdf>
- [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>
- [Fie82] FIENUP, J. R.: Phase retrieval algorithms: a comparison. In: *Appl. Opt.* 21 (1982), Aug, Nr. 15, 2758–2769. <http://ao.osa.org/abstract.cfm?URI=ao-21-15-2758>
- [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)
- [Gro03] GROSS, Rudolf: *Physik3-Skript*. 2003
- [Heb11] HEBBAR, K. R.: *Basics of X-Ray Diffraction and its Applications*. 2011
- [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)

- [Kri07] KRIESEL, David: *Ein kleiner Überblick über Neuronale Netze*. 2007 <http://www.dkriesel.com>
- [KRM⁺18] KLUGE, Thomas ; RÖDEL, Melanie ; METZKES, Josefine ; PELKA, Alexander ; GARCIA, Alejandro L. ; PRENCIPE, Irene ; REHWALD, Martin ; NAKATSUTSUMI, Motoaki ; MCBRIDE, Emma E. ; SCHÖNHERR, Tommy ; GARTEN, Marco ; HARTLEY, Nicholas J. ; ZACHARIAS, Malte ; ERBE, Arthur ; GEORGIEV, Yordan M. ; GALTIER, Eric ; NAM, Inhyuk ; LEE, Hae J. ; GLENZER, Siegfried ; BUSSMANN, Michael ; GUTT, Christian ; ZEIL, Karl ; RÖDEL, Christian ; HÜBNER, Uwe ; SCHRAMM, Ulrich ; COWAN, Thomas E.: *Observation of ultrafast solid-density plasma dynamics using femtosecond X-ray pulses from a free-electron laser*. 2018
- [Myu02] MYUNG, In J.: Tutorial on maximum likelihood estimation. (2002)
- [Nav00] NAVE, R.: *Fraunhofer Diffraction*. <http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/fraunhofcon.html>, 2000
- [ON15] O'SHEA, Keiron ; NASH, Ryan: An Introduction to Convolutional Neural Networks. (2015)
- [OUO18] OSSIETZKY UNIVERSITÄT OLDENBURG, Carl von: *FRAUNHOFER- und FRESNEL-Beugung, Interferenz*. https://uol.de/fileadmin/user_upload/physik/ag/physikpraktika/download/GPR/pdf/Beugung_Fraunhofer_Fresnel.pdf, 2018. – Accessed: 2019-02-19
- [QR11] QUIZA R., Davim J.: Computational Methods and Optimization. *Machining of Hard Materials*. 190 5 (2011), S. 1396–405
- [Rot06] ROTH, Ina: *Proseminar: Grundlagen Bildverarbeitung / Bildverstehen Orthogonale Funktionstransformationen*, 2006
- [Rud16] RUDER, Sebastian: *An overview of gradient descent optimization algorithms*. 2016
- [Sch] SCHLESINGER, Dmitrij: *Bildverarbeitung: Fourier-Transformation*
- [Tho18] THORMÄHLEN, 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. – [Online; accessed 04-October-2018]
- [Wis] WISSENSCHAFT, Spektrum der: *Plasma*
- [Zac17] ZACHARIAS, Malte: *MODEL-DRIVEN PARAMETER RECONSTRUCTIONS FROM*

SMALL ANGLE X-RAY SCATTERING IMAGES. <http://dx.doi.org/10.5281/zenodo.1208410>. Version: November 2017

Danksagung

Die Danksagung...

Erklärungen zum Urheberrecht

Hier soll jeder Autor die von ihm eingeholten Zustimmungen der Copyright-Besitzer angeben bzw. die in Web Press Rooms angegebenen generellen Konditionen seiner Text- und Bildübernahmen zitieren.