

Rheinisch-Westfälische Technische Hochschule Aachen
Institut für Eisenhüttenkunde

- Werkstofftechnik-

Masterarbeit

Weiterentwicklung eines Machine Learning tools
zur Darstellung der dreidimensionalen
Parameter verschiedener Mikrostrukturen

von

Niklas Fehlemann
Matr.-Nr. 345046

Durchgeführt am Lehr- und Forschungsgebiet für
Werkstoff- und Bauteilintegrität

Vom 01. September 2020 bis 17. Februar 2021

Prüfer: Univ. Prof. Dr.-Ing. S. Münstermann
 Dr.-Ing. Michael Dölz

Betreuer: M. Sc. F. Pütz

Eidesstattliche Versicherung

Fehlemann, Niklas

345046

Name, Vorname

Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

Weiterentwicklung eines Machine Learning tools zur Darstellung der dreidimensionalen Parameter verschiedener Mikrostrukturen

selbstständig und ohne unzulässige Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen,

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

- (1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
- (2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen,

Ort, Datum

Unterschrift

Inhaltsverzeichnis

1 Zusammenfassung	1
2 Einleitung	2
3 Stand der Wissenschaft und Technik	4
3.1 Werkstoffcharakterisierung	4
3.1.1 Gefüge und Mikrostruktur	5
3.1.2 Statistische Charakterisierung	6
3.1.3 EBSD	8
3.2 Machine Learning	10
3.2.1 Grundlagen Neuronaler Netze	12
3.2.2 Generative Adversarial Networks	16
3.2.3 Wasserstein Generative Adversarial Networks	19
3.3 Optimaler Transport	23
3.3.1 Wasserstein Distanz	23
3.3.2 Sinkhorn Iterationen - Entropische Regularisierung der Wasserstein Distanz	25
4 Entwicklung der Algorithmen	29
4.1 Zweidimensionale Mikrostruktur	29
4.1.1 Aufbau des Conditional Wasserstein Generative Adversarial Network - Gradient Penalty	30
4.1.2 Prinzipien der Auswertung	34
4.1.3 Materialien	35
4.2 Dreidimensionale Mikrostruktur	36
4.2.1 Verfahrensentwicklung	36
4.2.2 Validierung des Algorithmus	43
5 Ergebnisse und Diskussion	47
5.1 Generierung zweidimensionaler Mikrostruktur	47
5.1.1 Stabilisierung des Trainings	47
5.1.2 Vergleich verschiedene Aktivierungsfunktionen	53
5.1.3 Vergleich verschiedener Optimizer	59

5.2 Rekonstruktion dreidimensionaler Mikrostruktur	63
5.2.1 Anwendung des Algorithmus	63
5.2.2 Validierung des Algorithmus	69
5.2.3 Einfluss der Hyperparameter <i>threshold/range</i> auf die Rekonstruktion des Gefüges	78
5.2.4 Rekonstruktion der Drehungen	79
6 Schlussfolgerung und Ausblick	91
7 Literaturverzeichnis	93
8 Anhang	100

Kapitel 1 Zusammenfassung

Die statistisch repräsentative Modellierung der Mikrostruktur ist entscheidend für die korrekte Simulation der Schädigung und Verwendung von repräsentativen Volumenelementen (RVE's). Die verschiedene Eigenschaften der Mikrostruktur beeinflussen hierbei insbesondere die Initiierung und Akkumulation von Schädigung. Dazu sind vor allem zwei Dinge notwendig, zum einen müssen die verschiedenen Parameter der Mikrostruktur inklusive aller Interdependenzen multidimensional abgebildet werden. Zum anderen sollte die Mikrostruktur auch dreidimensional abgebildet werden können, das geläufige Vorgehen über EBSD-Aufnahmen berücksichtigt jedoch nur zwei Dimensionen, für die fehlende Richtungen müssen Annahmen getroffen werden.

In dieser Arbeit wurden Lösungsansätze für beide Problem entwickelt. Erstens wurde ein bestehendes WGAN zu einem CWGAN-GP weiterentwickelt, welches die zweidimensionalen Parameter statistisch repräsentativ klassifizieren und generieren soll. Es konnte gezeigt werden, dass dieser Algorithmus die Parameter mehrerer verschiedener Materialien gleichzeitig in höherer Geschwindigkeit und Genauigkeit im Vergleich zum bislang vorhandenen System abbilden kann. Dabei kann mittels sorgfältiger Auswahl der Aktivierungsfunktionen der beiden neuronalen Netzen die Qualität der erzeugten Daten noch weiter verbessert werden. Zudem wurde eine neue und verbesserte Auswertemethodik über die regularisierte Wasserstein-Distanz eingeführt.

Zweitens wurde ein Algorithmus entwickelt, welcher durch eine Kreuzkorrelation von Wertepaaren mittels drei EBSD-Aufnahmen die dreidimensionale Mikrostruktur des Materials rekonstruiert. Für diesen Algorithmus wurde ein eigenes Validierungsverfahren entworfen, mit welchem gezeigt werden konnte, dass der vorgeschlagene Algorithmus die dreidimensionale Mikrostruktur in guter Näherung rekonstruieren kann. Dies ist auch bei Berücksichtigung der Neigungen möglich. Weiterhin wurde gezeigt, dass die Rekonstruktion der Mikrostruktur ausgehend von zweidimensionalen Aufnahmen immer zu breit geschätzt wird, da nicht davon ausgegangen werden kann, bei einer zweidimensionalen Aufnahme jedes Korn in der Mitte zu treffen. Eine Kombination der rekonstruierten dreidimensionalen Mikrostruktur mit dem CWGAN-GP ermöglicht die Darstellung des vollständigen Gefüges.

Kapitel 2 Einleitung

Die Realisierung von Leichtbaupotenzialen gehört zu den wichtigsten Möglichkeiten, dem fortschreitenden Klimawandel beispielsweise durch die Gewichtsreduktion von Automobilen entgegenzuwirken. Hierbei bilden die exakte Beschreibung der Leistungsfähigkeit im Zusammenhang mit einer genauen Beschreibung der Schädigungsmechanismen von Stahl eine hervorragende Grundlagen, um Leichtbaupotenziale insbesondere in der Umformtechnik zu realisieren. Dabei gewinnt die Idee, Schädigung im Umformprozess gezielt auszunutzen, seit einigen Jahren an Bedeutung. Dazu ist eine genaue Kenntnis zum einen der *Schädigungsprozesse* und zum anderen der zugrundeliegenden *Mikrostruktur* nötig. Ist dies gegeben, kann die Mikrostruktur für die weitere Verwendung in der Simulationstechnik aufbereitet werden. Diese Grundlage zur Simulation dreidimensionaler Mikrostruktur bildet das *Statistische Representative Volumenelement (SRVE)*. Dabei handelt es sich um ein synthetisch generiertes Gefüge auf Basis einer statistischen Analyse der Materialeigenschaften. Diese müssen genug Informationen über die Mikrostruktur enthalten, allerdings trotzdem noch auf einer Skala deutlich kleiner als die des Realen Bauteils liegen [16]. Entscheidend für die Qualität eines RVE's ist es, dass möglichst alle Parameter der Mikrostruktur (dazu gehören die Anteile der einzelnen Phasen am Gefüge, die Formen und Größenverteilungen der Körner, die Textur etc.) abgebildet werden können, um möglichst genaue Simulationen der Leistungsfähigkeit durchführen zu können.

Daher müssen vor der Erstellung eines RVE's zunächst Informationen über die Mikrostruktur eines Stahls zur statistischen Beschreibung gewonnen werden. Hierfür existieren verschiedene Möglichkeiten: Die genauesten Ergebnisse lassen sich erzielen, indem die Mikrostruktur *unmittelbar* dreidimensional untersucht wird: Dazu wird eine Aufnahme des Materials mittels Scanning Elektron Microskopy (SEM) und Elektron Backscatter Diffraction (EBSD) angefertigt, Material in definierter Höhe abgetragen und eine neue Aufnahme angefertigt. Dies wird so lange wiederholt, bis genügend Aufnahmen vorliegen, welche zu einer dreidimensionalen Darstellung übereinander gelegt werden können. Dieses Verfahren ist allerdings sehr aufwendig, da zum einen immer eine exakt definierte Menge Material abgetragen werden muss und zum anderen müssen die Schnitte zusätzlich durch Härtemessungen aufeinander angepasst werden [37], [62]. Einfache Informationen über die Mikrostruktur lassen sich mittels

einer einzelnen EBSD Aufnahme gewinnen. Allerdings wird so die Mikrostruktur nur zweidimensional beleuchtet und es gehen wichtige Informationen verloren.

Am IEHK existiert bereits ein Algorithmus zur Erstellung von RVE's [29]. Ziel der vorliegenden Arbeit ist die **Verbesserung der Inputdaten** für diesen Generator. Aktuell werden diese aus einer einzelnen *zweidimensionalen* EBSD-Aufnahme gewonnen. Hierbei konnten in einer vorangehenden Studie bereits Fortschritte erzielt werden, indem aus den per EBSD-Aufnahme gewonnenen Gefügeeigenschaften mittels eines *generativen Machine learning*-Algorithmus synthetische Mikrostruktur erzeugt wird [49]. Dieses Verfahren garantiert, dass alle Interdependenzen zwischen den Gefügeparametern korrekt abgebildet werden, beispielsweise der Zusammenhang zwischen Kornlängung und Korngröße. Aus dem vorgestellten Stand ergeben sich zwei Aufgabenstellungen für diese Arbeit:

1. Aktuell ist das generative Verfahren (ein sogenanntes Wasserstein GAN) nur in der Lage eine begrenzte Anzahl Korneigenschaften zu erzeugen. Weiterhin kann nicht zwischen verschiedenen Materialien differenziert werden. Dahingehend soll im ersten Teil dieser Arbeit der Algorithmus um diese Eigenschaften erweitert werden und nach Möglichkeit die Qualität der synthetischen Mikrostruktur verbessert werden.
2. Wie beschrieben, liefert eine EBSD-Aufnahme nur *zweidimensionale* Informationen über das Gefüge. Daher werden für die fehlende dritte Raumrichtung Annahmen getroffen, z.b. wird die dritte Dimension eines Korns gleich der ersten angenommen. Mittels EBSD-Aufnahmen aus drei verschiedenen Raumrichtungen soll die dritte Dimension durch passende Korrelationen der Kornparameter extrapoliert und somit eine Rekonstruktion der dreidimensionalen Mikrostruktur erreicht werden.

Lösungen zu beiden Problemstellungen verbessern den RVE-Input dahingehend, dass zum einen mehr Parameter der Mikrostruktur (bsp. die Textur) berücksichtigt werden können und zum anderen die Qualität durch die Berücksichtigung der fehlenden dritten Dimension verbessert wird.

Kapitel 3 Stand der Wissenschaft und Technik

Übergeordnetes Ziel dieser Arbeit ist die Charakterisierung und statistische Beschreibung von Mikrostruktur, sowie weiterhin die Verbesserung der Darstellung statistisch repräsentativer Mikrostruktur in Repräsentativen Volumenelementen (RVE). Hierzu sind Kenntnisse über die Grundlagen von Mikrostruktur und Gefüge (Kapitel 3.1.1) und die Formen der statistischen Charakterisierung (Kapitel 3.1.2) notwendig. Der Fokus liegt hierbei besonders auf der Charakterisierung der Mikrostruktur mittels zweidimensionaler EBSD-Aufnahmen (Kapitel 3.1.3), da diese Arbeit im Speziellen das Ziel verfolgt, dreidimensionale Mikrostruktureigenschaften aus zweidimensionalen Aufnahmen herzuleiten.

Für die statistische Charakterisierung der Mikrostruktur und zur Mustererkennung in Materialdaten werden Verfahren aus dem Bereich des Maschinellen Lernens verwendet (Kapitel 3.2). Ein Unterbereich des Maschinellen Lernens ist das sogenannte *Deep Learning*, welches mit künstlichen Neuronalen Netzen, einer dem menschlichen Gehirn nachempfundenen Technik, arbeitet (Kapitel 3.2.1). Da für die Erstellung von RVE's *synthetische Mikrostruktur* benötigt wird, wird im Rahmen dieser Arbeit ein sogenanntes Wasserstein Generative Adversarial Network verwendet: Hierbei handelt es sich um einen Algorithmus auf Basis neuronaler Netze, welcher auf der Grundlage von realen Mikrostrukturdaten statistisch repräsentative Mikrostruktur generieren kann. Die theoretischen Hintergründe werden in den Kapiteln 3.2.2 und 3.2.3 erläutert. Vertiefte mathematische Details im Zusammenhang mit den Konzepten des maschinellen Lernens sind in Kapitel 3.3 zu finden.

3.1 Werkstoffcharakterisierung

In diesem Kapitel sollen wesentliche Grundlagen der Charakterisierung von Werkstoffen besprochen werden. Zunächst sollen wesentliche Merkmale wie Gefüge und Mikrostruktur, sowie die verschiedenen Phasen des Stahls besprochen werden. Teil zwei hat die wesentlichen Parameter einer Mikrostruktur und die statistische Beschreibung dieser zum Thema. Zum Abschluss wird noch kurz auf die Experimentelle Charakterisierung mittels der *Electron Backscatter Diffraction* (EBSD) eingegangen.

3.1.1 Gefüge und Mikrostruktur

Nach Schott/Worch lautet die Definition von Gefüge wie folgt: "Der Begriff *Gefüge* kennzeichnet die Beschaffenheit der Gesamtheit jener Teilvolumina, von denen jedes hinsichtlich seiner Zusammensetzung und der räumlichen Anordnung seiner Bausteine im Bezug auf ein in den Werkstoff gelegtes Achsenkreuz in erster Näherung homogen ist". [59]

Ein Gefüge bildet sich bei Metallen bei der Erstarrung einer Schmelze: Zunächst bilden sich durch Zusammenlagerung von Atomen zu einem periodischen Kristallgitter einzelne Keime, welche sich durch Hinzufügen weiterer Atome zu *Körnern* weiterentwickeln. Sofern die Wärmeabfuhr bei der Erstarrung durch den Kristall erfolgt, bilden sich Kugelförmige, auch *globulitisch* genannte Körner. Sollte die Schmelze stark unterkühlt sein, kann es vorkommen, dass die Wärme direkt über die Schmelze abgeführt wird und sich langezogene, astartige Körner bilden, welche als *Dendriten* bezeichnet werden [24].

Haben mehrere Körner die gleichen, d.h. homogene Eigenschaften, werden diese als Phase bezeichnet. Dabei kann eine Phase auch aus mehreren Elementen bzw. Bestandteilen bestehen. Eine Phasengrenze trennt zwei verschiedene Phasen, während analog dazu eine Korngrenze einzelne Körner trennt. Ein Gefüge wird somit durch die Gesamtheit aller Phasen und Körner sowie der zugehörigen Phasen- und Korngrenzen charakterisiert. Dazu gehören zusätzlich die Strukturfehler, welche im Kristallgitter vorliegen.

Die drei wichtigsten Phasen des Eisens sind *Ferrit*, *Austenit* und *Martensit*. Ferrit und Austenit unterscheiden sich dabei in ihrer Kristallstruktur. Die kubisch raumzentrierte Modifikation (α) wird als Ferrit bezeichnet, die kubisch flächenzentrierte Modifikation (γ) als Austenit. Bei reinem Eisen ist Ferrit bis zu einer Temperatur von 911°C stabil, in einem Temperaturbereich von 911 - 1392°C der Austenit. Oberhalb von 1392° bis zum Schmelzpunkt von 1536°C ist wieder krz, auch bezeichnet als δ -Eisen, stabil [11]. Durch die Zulegierung verschiedener Elemente existieren allerdings auch Stähle, welche auch bei Raumtemperatur ein rein austenitisches Gefüge aufweisen, z.b. die sogenannten Hochmanganstähle [65].

Martensit entsteht durch athermische Umwandlung von austenitischem Gefüge. Dabei ist die Kristallstruktur stark hexagonal (ϵ) verzerrt [43]. Je nach Stahlsorte kann Martensit in verschiedenen Formen vorliegen. Bei dem in dieser Arbeit hauptsächlich betrachteten Dualphasenstahl DP800 liegt der Martensit an den Korngrenzen der ferritischen Matrix [12].

3.1.2 Statistische Charakterisierung

Für das Ziel einer statistischen Auswertung eines Materials müssen die zu untersuchenden Parameter zuvor definiert werden. In den meisten Fällen handelt es sich hierbei um eine Auswahl der Folgenden:

- Die Kornfläche, bzw. das Kornvolumen [71]
- Die Phasenverteilung [24]
- Die Kornorientierungen [25]
- Nachbarschaftsverhältnisse [24]
- Die Kristallografische Orientierung, bzw. Textur [58]

Eine exakte dreidimensionale Analyse einer Mikrostruktur ist nur durch aufwendige Ansätze möglich. Dazu wird das sogenannte serial-sectioning Verfahren angewandt. Dabei werden Lichtoptische Aufnahmen in Kombination mit EBSD-Aufnahmen gemacht, nach denen eine genau definierte Schicht Material abgetragen wird. Anschließend werden erneute Aufnahmen gemacht. Diese Prozedur wird wiederholt, bis ausreichend Aufnahmen zur Verfügung stehen, welche korrekt übereinander angeordnet eine rekonstruierte Dreidimensionale Mikrostruktur ergeben [62], [37]. In den meisten Fällen findet eine Analyse jedoch anhand zweidimensionaler EBSD-Daten statt. Zur besseren Beschreibung der Eigenschaften der Mikrostruktur werden die Körner hierbei als Ellipse angenommen [29]. Wird ein Korn als Ellipse betrachtet, können zusätzliche Größen statistisch erfasst und zu einer Rekonstruktion des Gefüges genutzt werden. Dazu gehören die Länge der beiden Halbachsen a und b , das Verhältnis (Aspect Ratio) zwischen beiden Halbachsen (ein Maß für die Längung des Korns) und der Winkel der Hauptachse im Bezug zu einem festgelegten Koordinatenursprung. Die Kristallografische Orientierung eines Korns wird meist mithilfe der drei Eulerwinkel angegeben. Es ist zwar möglich, die einzelnen Größen durch einen Mittelwert mit zugehöriger Standardabweichung bzw. Varianz anzugeben, eine genauere Klassifikation wird jedoch durch die Verwendung von Wahrscheinlichkeitsdichtefunktionen erreicht [24]. Zu den gebräuchlichsten Verteilungen gehören die Logarithmische Normalverteilung, die Gammaverteilung und die Rayleighverteilung [8], [31], [44].

Die Logarithmische Normalverteilung ist eine Abwandlung der Klassischen Normalverteilung. Beschrieben wird die Verteilung einer Zufallsvariable X wenn $Y = \ln(X)$ normalverteilt ist. Charakterisiert wird sie durch den Mittelwert M und die Standardabweichung S . Die Dichtefunktion der Verteilung ist:

$$f(x) = \frac{1}{S * \sqrt{2\pi}} * \exp\left(-\frac{(\ln(x) - M)^2}{2S^2}\right) \quad (1)$$

Die logarithmische Normalverteilung ist nur für positive Werte von x definiert. Um mittels dieser Dichtefunktion die Verteilung von Gefügeparametern zu bestimmen müssen die Parameter S und M an die realen Histogramme angepasst werden. Eine von *Berbennier et al.* durchgeführte Studie zu dem Element Zirkonium liefert eine Spanne von 0,33 bis 0,76 für die Standardabweichung S bei einem mittleren Korndurchmesser von $100\mu m$. In Bild 1 sind die beiden Extremfälle 0,33 und 0,76 dargestellt.

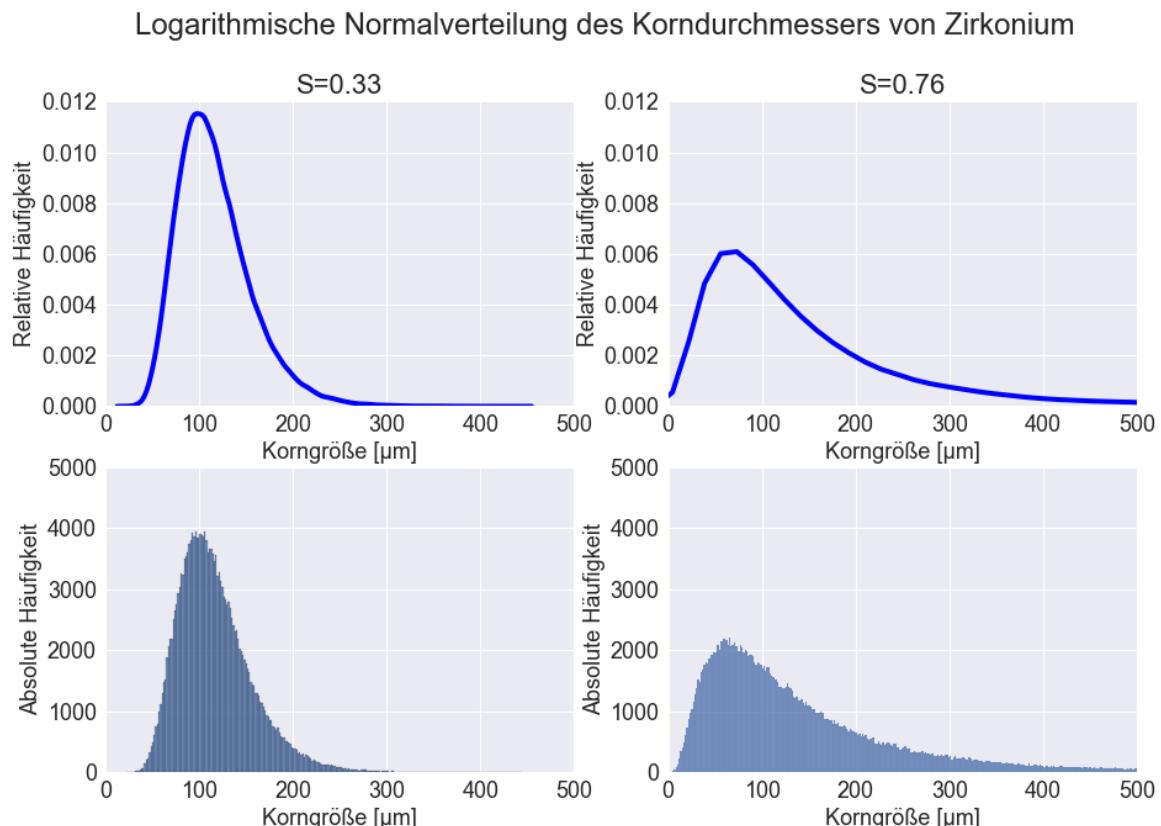


Bild 1: LogNormal Verteilung für Zirkonium mit Standardabweichung 0,33 (links) und 0,76 (rechts) nach [8].
(Verteilungen eigenständig erzeugt - 200000 generierte sample)

Die Gammaverteilung ist eine Verallgemeinerung der Exponentialverteilung. Statt durch Mittelwert und Standardabweichung ist sie durch einen Formparameter a und einen Skalenparameter b charakterisiert.

$$f(x) = \frac{b^a}{\Gamma(a)} * x^{a-1} * \exp(-bx) \quad (2)$$

Γ = Gammafunktion

Durch die passende Wahl der Parameter a und b nimmt die Gammafunktion eine ähnlich Gestalt an wie die logarithmische Normalverteilung in Bild 1. Es gibt Hinweise darauf, dass die Gammaverteilung grundsätzlich bessere Ergebnisse als die logarithmische Normalverteilung liefert, da die Gammaverteilung nicht symmetrisch ist [20]. Die logarithmische Normaverteilung ist bei logarithmierter x-Achse symmetrisch. Weiterhin kann auch die Rayleigh-Verteilung zur Charakterisierung von Gefügeparametern verwendet werden.

$$f(x) = \frac{x}{\sigma^2} * \exp\left(-\frac{x^2}{2\sigma^2}\right) \quad (3)$$

Im Gegensatz zu den bisher vorgestellten Verteilungen hat die Rayleighverteilung nur einen Parameter, σ . (Hierbei ist σ ein Parameter, nicht die Standardabweichung der Verteilung)

Die ausschließliche Beschreibung der Gefügeparameter über eine der vorgestellten Verteilungen hat jedoch den Nachteil, dass Interdependenzen zwischen den einzelnen Parametern nicht abgebildet werden [49]. In Bild 2 sind die Abhängigkeiten der Parameter Kornfläche (Area), Kornneigung (Slope) und Achsenverhältnis (Aspect Ratio) für einen Dualphasenstahl DP800 dargestellt. Diese Darstellung verdeutlicht, dass die Abhängigkeiten der Parameter untereinander ebenso berücksichtigt werden müssen, wie die eindimensionale Verteilung der Parameter. Zur Abbildung dieser multidimensionalen Verteilungen können z.b. die in Kapitel 3.2.1 vorgestellten neuronalen Netze verwendet werden. Mittels eines passenden Generative Adversarial Networks (Kapitel 3.2.2) kann synthetische Mikrostruktur direkt aus realen Daten produziert werden [49].

3.1.3 EBSD

Electron Backscatter Diffraction (Elektronen-Rückstreubeugung), kurz EBSD ist ein Verfahren zur Analyse von Kristallstrukturen. Es basiert auf dem 1912 von Max von Laue entdeckten Prinzip, dass Röntgenstrahlen an einem Kristallgitter gebeugt werden können [13]. Ein EBSD-System arbeitet nach dem in Bild 3 dargestellten Prinzip. Eine Elektronenquelle emittiert einen kontinuierlichen Strahl von Primärelektronen,

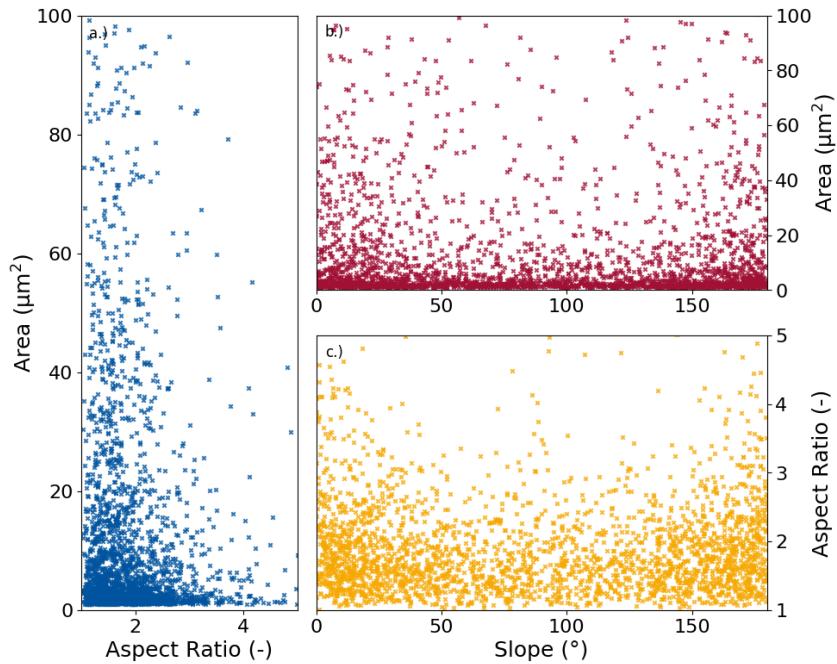


Bild 2: Verteilung von Kornparametern für einen Dualphasenstahl DP800 [49]

welche auf die zu untersuchende Probe treffen. Zur Erzeugung einer besseren Beugung wird die Probe dabei in einem Winkel von 70° angestrahlt. Im Material treffen die Elektronen auf Gitterebenen. Falls nun an den Gitterebenen die *Bragg-Gleichung* erfüllt ist, kommt es zu konstruktiver Interferenz (d.h. die Wellen verstärken sich) [13]:

$$n * \lambda = 2 * d * \sin(\Theta) \quad (4)$$

$n * \lambda$ = Ganzzahliges Vielfach der Wellenlänge

Θ = Eintrittswinkel

d = Netzebenenabstand

Diese Gleichung ist erfüllt, wenn der Elektronenstrahl auf eine Schar von Netzebenen (mit Abstand d zueinander trifft). Die gebeugten Elektronen werden von einem Schirm mit einer dahinter platzierten Kamera aufgefangen. Auf einem so erzeugten Bild ist ein charakteristisches Muster von Beugungslinien zu sehen, die sogenannten *Kikuchi-Linien* (Bild 3) [33]. Ein so erzeugtes Beugungsbild korreliert direkt mit den Gitterebenen. Dabei entsprechen die Winkel zwischen mehreren Linien den Winkel zwischen Gitterebenen. Mittels einer von einer Software durchgeföhrten Hough-Transformation (Ein Verfahren zur Bestimmung von geometrischen Mustern in einem binären Bild) können die Orientierungen bestimmt werden [60] [15]. Dazu wird die Geometrie des EBSD-Bildes mit Netzebenen in der Elementarzelle des Kristalls des zu untersuchenden Material verglichen. Ein Algorithmus berechnet über den Vergleich von verschiedenen Kombinationen von Kikuchi-Linien und theoretischen Daten die wahrscheinlichste Orientierung für die Ebenen [68]. Dabei werden die Orientierungen

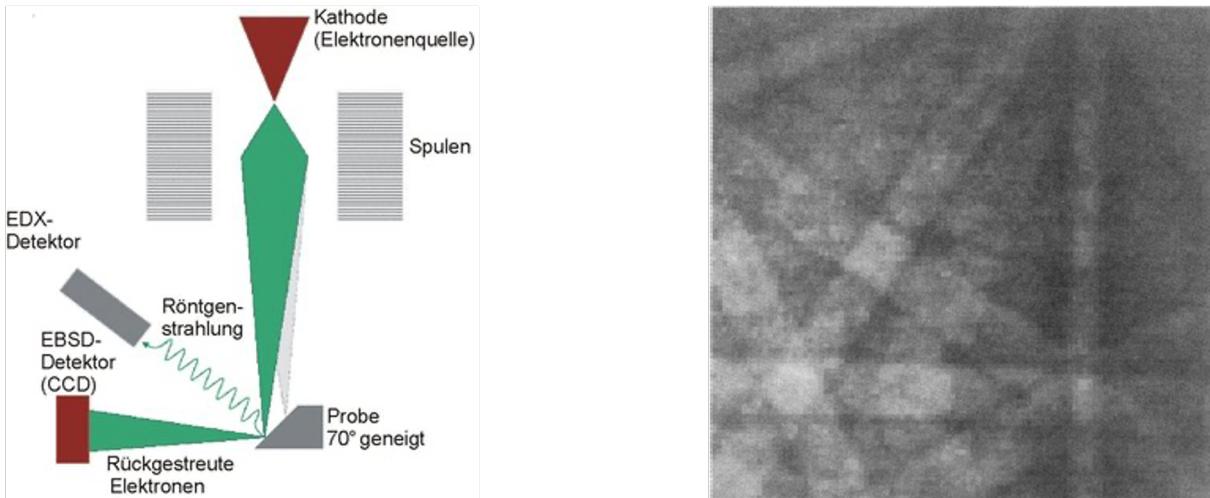


Bild 3: Schematischer Aufbau eines EBSD-Systems [1] (links) und charakteristische Kikuchi Linien (rechts)

entweder in Form von Miller-Indizes oder Eulerwinkeln angegeben. Die Summe aller Orientierungen wird als kristallografische Textur bezeichnet.

Grundsätzlich müssen die Rohdaten einer EBSD-Aufnahme einem Postprocessing unterzogen werden. Hauptgrund hierfür ist, dass die oben genannten Informationen bzw. Orientierungen punktweise für jeden Pixel vorliegen. Um diese in Flächenbasierter Informationen (d.h. die Körner) umzuwandeln, werden Orientierungsunterschiede herangezogen. Alle Bereiche, welche weniger als 15° voneinander abweichen, werden zu einem jeweiligen Korn zusammengefasst. Der Orientierungsunterschied von 15° entspricht einer Großwinkelkorngrenze [24]. Diese Annahme wird getroffen, da Spannungs- und Dehnungsakkumulation nur an Großwinkelkorngrenzen stattfindet. Zudem können diese als Hindernis für die Rissausbreitung bzw. Schädigung angenommen, wohingegen Kleinwinkelkorgrenzen von einem Riss leichter überwunden werden können [70], [69]. Die Spannungsakkumulation ist besonders für das zur Simulation der RVE's verwendeten *Crystal-Plasticity*-Modell von hoher Bedeutung.

3.2 Machine Learning

Der Begriff *Maschinelles Lernen* bezeichnet eine Gruppe von Algorithmen, die der Generierung von Wissen aus vorhandenen Daten dient. Gelernt wird mithilfe von Beispielen, die erlernten Erkenntnisse können auf andere Daten angewandt werden. D.h. es werden nicht einfach Beispiele "auswendig gelernt" sondern reale Muster (engl. Pattern) in den Daten erkannt. Beispiele für die Anwendung von Maschinellem Lernen sind z.B. Bild- und Spracherkennung oder die Erkennung von Kreditkartentrug. Im Rahmen dieser Arbeit werden Verfahren des Maschinellen Lernens zur Identifikation von Mustern und Zusammenhängen in den Materialdaten genutzt.

Besonderer Vorteil ist hierbei, dass im Gegensatz zu den in Kapitel 3.1.2 vorgestellten Wahrscheinlichkeitsdichtefunktionen auch die mehrdimensionalen Abhängigkeiten der Parameter untereinander abgebildet werden können.

Grundsätzlich wird zwischen zwei verschiedenen Kategorien unterschieden, dem *überwachten (supervised)* und dem *unüberwachten (unsupervised)* Lernen. Ersteres funktioniert durch vorgegebene Input-Output Paare, wodurch ein Algorithmus lernt, anhand eines vorgegebenen Inputs den passenden Output vorherzusagen (Beispielsweise, ob aufgrund des Zahlungsverkehrs einer Kreditkarte (Input) Betrug vorliegt oder nicht (Output)). Beim unüberwachten Lernen versucht ein Algorithmus ein statistisches Modell aus einem Datensatz zu Generieren. Hierzu gehören z.B. Clustering-Algorithmen, bei denen Teilmengen der Daten aufgrund ähnlicher Eigenschaften zusammengefasst werden.

Eine wichtige Unterkategorie des Maschinellen Lernens ist das sogenannte *Deep Learning* (Vgl. Bild 4), welches vor allem seit 2012 durch die immer weiter ansteigende Leistung von Computern sehr viel Aufmerksamkeit erhält. Gemeinhin gilt die über 77.000x zitierte Veröffentlichung “ImageNet Classification with Deep Convolutional Neural Networks” als Startpunkt des bis heute anhaltenden Interesse am Deep Learning. Beim Deep Learning werden *kiinstliche neuronale Netze* verwendet, welche den Neuronen des menschlichen Gehirns nachempfunden sind. Mit diesen können viele Aufgaben aus dem supervised und unsupervised Learnings deutlich genauer ausgeführt werden, indem die Algorithmen auf sehr großen Datensätzen trainiert werden.

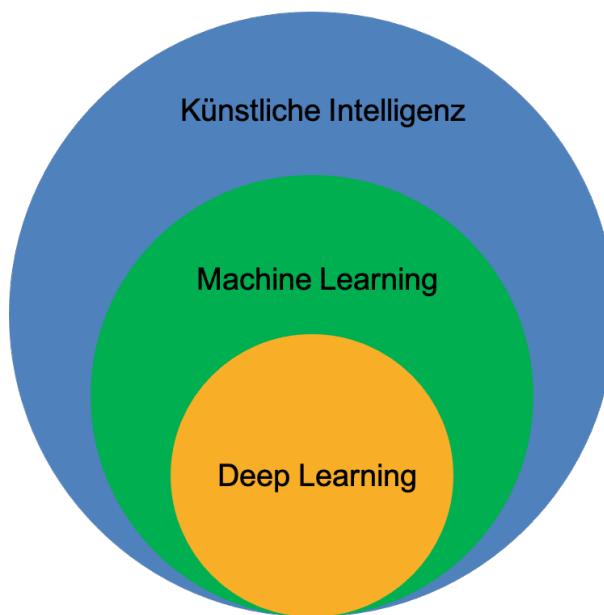


Bild 4: Abstufung verschiedener Bereich der Künstlichen Intelligenz [3]

Da der Gesamte Themenbereich rund um das Deep Learning sehr umfangreich ist,

werden im Folgenden nur für diese Arbeit relevante Themen beleuchtet. In dieser Arbeit wird im Wesentlichen ein Algorithmus/Verfahren verwendet, das Wasserstein-Generative Adversarial Network. Dieses wird in dieser Arbeit eingesetzt, um synthetische Mikrostruktur für die Erzeugung von RVE's zu generieren (Kapitel 3.2.2 und 3.2.3). Da dieser Algorithmus zum Teilbereich des Deep Learnings gehört und auf Neuronalen Netzen basiert, werden diese in Kapitel 3.2.1 näher beleuchtet.

3.2.1 Grundlagen Neuronaler Netze

Künstliche Neuronale Netze oder KNN's sind - wie der Name bereits andeutet - den realen Neuronen im Menschlichen Gehirn nachempfunden. KNN's gehören dabei zum sogenannten Deep Learning, welches einen Teilbereich der Künstlichen Intelligenz bildet. Neuronale Netze kommen sowohl in der Bilderkennung, Objekterkennung (z.b. [52]), Sprachverarbeitung und seit kurzer Zeit auch in Generativen Verfahren zur Erzeugung von Daten zum Einsatz.

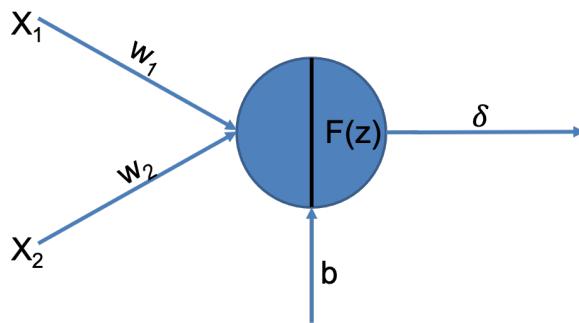


Bild 5: Schematische Darstellung eines Perzeptrons. X bezeichnet den Input, W die Gewichte, b den Bias, F(z) die Aktivierungsfunktion und δ den Output - Eigene Anfertigung

Die einfachste Form eines Neuronalen Netzes ist das sogenannte *Perceptron*, eine abstrahierte Darstellung des realen Neurons. Der *Input* ist ein Vektor x , mit der Länge d . Das Neuron selbst verfügt über einen Gewichtungsvektor w , welcher mit dem Inputvektor multipliziert wird. Ein Neuron ist Beispielhaft in Bild 5 zu sehen.

$$z = b + \sum_{i=1}^d w_i x_i \quad (5)$$

b = sog. Bias des Neurons

Gleichung 5 macht deutlich, dass durch das Neuron eine lineare Transformation des Inputvektors durchgeführt wird. Um auch nichtlineare Zusammenhänge abzubilden wird der Wert z an eine sogenannte Aktivierungsfunktion übergeben. Die heutzutage gebräuchlichste Aktivierungsfunktion ist die Rectified Linear Unit (ReLU) [27]:

$$\delta = \max(0, z) \quad (6)$$

Weitere Aktivierungsfunktionen sind zum Beispiel die Sigmoid, Tangens-Hyperbolicus oder Swish-Funktion. Die Suche nach passenden Aktivierungsfunktionen ist Bestandteil der aktuellen Forschung mit dem Ziel, das Training Neuronaler Netze schneller und stabiler zu gestalten [51].

Der Wert δ ist der Ausgabewert des Neurons. In der einfachsten Form dient ein einzelnes Neuron als Klassifikator. Frank Rosenblatt, welcher das Perceptron 1958 erstmals beschrieb, konnte zeigen, dass eine Schicht zur Klassifizierung verwendeter Neuronen verschiedene Funktionen der Aussagenlogik wie UND, ODER und NICHT darstellen kann [54]. Allerdings ist eine einzelne Schicht von Neuronen nur dazu in der Lage, Daten mittels einer Gerade zu separieren. Dadurch kann z.B. der XOR-Operator (Bild 6) in dieser Form nicht aufgelöst werden, da die Ergebnisse nicht linear separierbar sind [39]. Exclusive-or bedeutet, dass nur bei einem Input von (0,1) oder (1,0) einen Output von 1 erzeugt wird. In Bild 6 ist zu sehen, dass keine Gerade die Outputs True und False voneinander trennen kann.

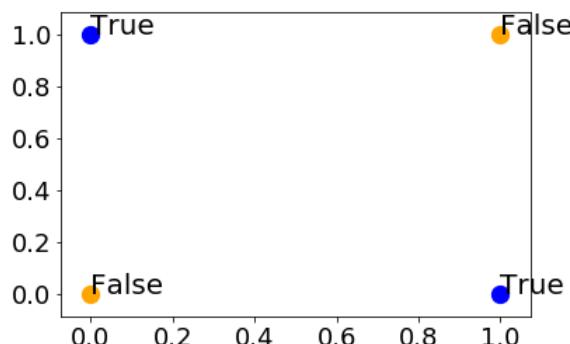


Bild 6: Grafische Darstellung der XOR-Klassifikation. Der Ausgangswert ist True (blau)

Dieses Problem wird behoben, indem das neuronale Netz in die Tiefe erweitert wird.

Dazu werden zwischen einem Input-Layer und einem Output-Layer weitere sog. Hidden-Layer eingefügt. In der einfachsten Form sind alle Neuronen mit jedem Neuron der nächsten Schicht verbunden, daher werden diese als Feed-Forward-Netze oder Multi-layer Perceptron (MLP) bezeichnet [22]. Die Verbindung besteht darin, dass von einem Neuron in Schicht eins der Ausgabewert δ an alle Neuronen der zweiten Schicht übergeben werden. Für spezielle Aufgaben existieren besondere Netze, z.b. Convolutional Networks für die Bildverarbeitung [7] und Recurrent Networks für die Sprachverarbeitung [30]. Moderne Neuronale Netze für die Bilderkennung können bis zu einhundert Schichten tief sein [28].

Das Training eines Neuronalen Netzes geschieht in **zwei** Schritten. Der Einfachheit halber soll hier nur das überwachte (engl. supervised) Lernen betrachtet werden, da in dieser Arbeit solche Verfahren zum Einsatz kommen:

Schritt Eins ist der Vorwärtspass durch das neuronale Netz. Hier wird für einen gegebenen Input ein Output berechnet, je nach Aufgabe des Netze z.b. die Wahrscheinlichkeit, mit der ein Bild oder ein anderer Input zu einer bestimmten Klasse gehört. Da das supervised Learning wie beschrieben auf vorgegeben Input-Output Paaren basiert, werden mittels einer sogenannten Ziel-/Lossfunktion der durch das Netz berechnete Output und der durch die Trainingsdaten vorgegebene Output verglichen. Anschaulich gesagt, wird geprüft ob durch das Netz für ein Bild die korrekte Klasse vorausgesagt wird.

Mathematisch existieren verschiedene Funktionen um die Abweichung zwischen Vorhersage und vorgegebenem Output zu berechnen. Die bekannteste und intuitiv verständlichste Zielfunktion ist der *mean-squared* bzw. *mean-absolute Error*, der vor allem bei Problemen im Bereich der Regression eingesetzt wird. Da die meisten Neuronalen Netze jedoch zur Klassifikation von Daten verwendet werden, der Output also eine berechnete Wahrscheinlichkeit auf dem Intervall $[0,1]$ ist, eignet sich die sogenannte Kreuzentropie am besten [55].

$$L = - \sum_{i=1}^n \sum_{j=1}^m (y_j \log(\hat{y}_j)_i) \quad (7)$$

n = Anzahl der Beobachtungen

m = Anzahl der Klassen

\hat{y}_i = Vorhergesagter Output

y_i = vorgegebener Output

Für genauere Erläuterungen zum Konzept der Entropie in der Informationstechnik siehe Kapitel 3.3.

Schritt Zwei, der eigentliche Lernprozess wird als *Backpropagation* bezeichnet [56]. Backpropagation bedeutet, dass die Gewichte mit dem Gradienten des Loss in Abhängigkeit zu den Gewichten aktualisiert werden. Der Fehler wandert sozusagen zurück durch das Netz bis zu den Gewichten. Ziel ist die Minimierung der Lossfunktion:

$$w_i^{k+1} = w_i^k - \eta * \frac{\partial L}{\partial w_i} \quad (8)$$

k = Aktueller Iterationsschritt

η = Lernrate

Wie in Gleichung 7 zu sehen ist, kann das Differential $\frac{\partial L}{\partial w_i}$ nicht direkt gebildet werden, da die Gewichte in der Lossfunktion nicht direkt enthalten sind. Daher wird die Kettenregel für Differenziale angewandt, um den Gradienten auf Umwegen zu berechnen: (hier in allgemeiner Schreibweise ohne Laufvariablen)

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \delta} * \frac{\partial \delta}{\partial z} * \frac{\partial z}{\partial w} \quad (9)$$

Das in Gleichung 9 beschriebene Vorgehen wird als Gradientenabstiegsverfahren bezeichnet (eine erste Version dieses Verfahrens erschien 1951 in [53]). Durch dieses Verfahren bewegt sich der Gradient an der Lossfunktion in Richtung des Minimums herab und stoppt dort, da bei einem Minimum der Gradient gleich Null ist. Allerdings erfolgt auch an einem Sattelpunkt ein Stop, da auch hier der Gradient gleich Null ist [14]. Allerdings wird in den meisten Fällen ein lokales Minimum erreicht, welches ebenfalls eine sehr gut Genauigkeit liefert.

Für das Training eines Neuronalen Netzes wird die Gesamtheit der Trainingsdaten in sog. *Batches* geteilt. Dies bedeutet, dass für einen Batch der Größe n mit Gleichung 7 ein akkumulierter Loss für n Datensätze berechnet und die Backpropagation für

diesen Gesamtloss durchgeführt wird. Ein vollständiger Durchlauf durch die Gesamtheit der Trainingsdaten wird als *Epoche* bezeichnet. Ist eine vorher festgelegte Anzahl von Epochen durchlaufen, endet das Training. Die Anzahl der Epochen, die Batchgröße n sowie die Lernrate (vgl. Gleichung 8) werden als *Hyperparameter* bezeichnet. Die sorgsame Auswahl geeigneter Hyperparameter ist für den Erfolg des Trainings entscheidend [10], [9].

3.2.2 Generative Adversarial Networks

Generative Adversarial Networks (im Folgenden: GAN) sind eine Erfindung des Amerikanischen Mathematikers Ian Goodfellow aus dem Jahre 2014. [23] Hierbei handelt es sich um ein *generatives* Verfahren, d.h. es sollen keine bekannten Daten klassifiziert, sondern neue Daten anhand eines bekannten Datensatzes erzeugt werden. Dabei besteht ein GAN aus zwei Teilen: Einem *Generator* (kurz G), welcher falsche Datensätze generiert, und einem *Discriminator* (kurz D), welcher zwischen echten und falschen Daten unterscheidet. In der einfachsten Version eines GAN's handelt es sich bei beiden Bestandteilen um die in Kapitel 3.2.1 vorgestellten MLP's. Generator und Discriminator verhalten sich zueinander gegenläufig und minimieren jeweils eine eigene Lossfunktion. [23] Bild 7 zeigt den Aufbau eines GANs schematisch. Bei z handelt es sich um einen n -dimensionalen Vektor aus Zufallszahlen (üblicherweise aus einer Normalverteilung), auch *noise* genannt, der vom Generator als Grundlage für die zu erzeugenden Daten genutzt wird. Ziel ist es, dass die Verteilung p_G des Generators der Verteilung der Realen Daten p_{Daten} entspricht. Beim einem GAN handelt es sich weder um ein rein supervised noch um ein rein unsupervised Verfahren, da sowohl reale Daten als Grundlage dienen, als auch selbständig Daten aus dem Zufallsvektor erzeugt werden. Man spricht daher von einem *semi-supervised*-Algorithmus.

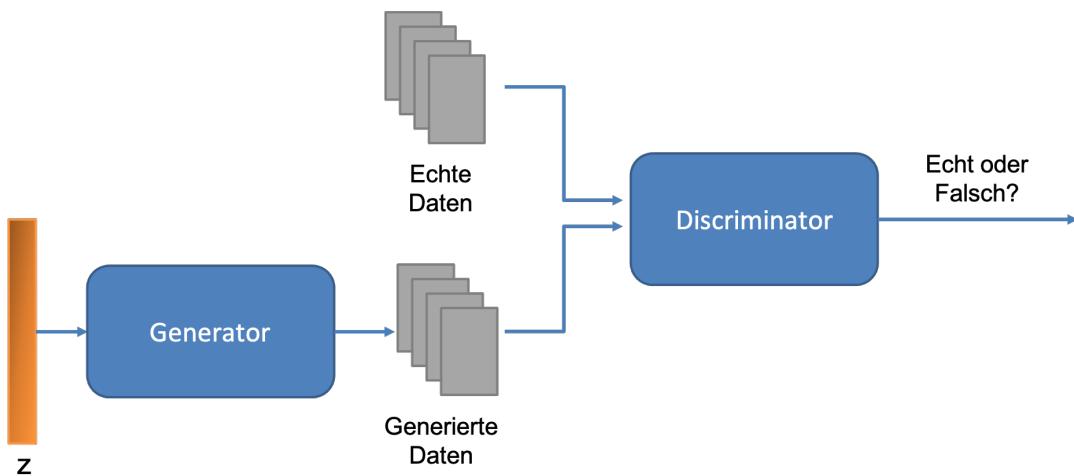


Bild 7: Schematischer Aufbau eines Generative Adversarial Networks (nach [2])

Zunächst soll der Generator betrachtet werden. Dieser erhält als Input einen Vektor z ,

der Output sind generierte falsche Daten. Ziel des Generators ist es, möglichste gute falsche Daten zu erzeugen, welche der Discriminator nicht von den realen Daten unterscheiden kann. Die Lossfunktion lautet wie folgt:

$$\min L_G = \log(1 - D(G(z))) \quad (10)$$

D = Discriminator

G = Generator

z = Vektor von Zufallszahlen

Goodfellow stellte allerdings fest, dass diese Funktion zum Trainieren des Generators ungeeignet ist. Daher wird diese Funktion umgedreht, sodass der Generator seine Zielfunktion maximiert. Das liegt daran, dass das Maximieren der umgedrehten Lossfunktion größere Gradienten während der Backpropagation ergibt. Somit verbessert es den Lernprozess [23]. Die neue Zielfunktion lautet:

$$\max L_G = \log(D(G(z))) \quad (11)$$

D = Discriminator

G = Generator

z = Vektor von Zufallszahlen

Zur Gültigkeit der Gleichungen 10 und 11 ist es erforderlich, dass der Output des Discriminators als eine Wahrscheinlichkeit auf dem Intervall $[0, 1]$ definiert ist.

Das Ziel des Discriminators ist es, möglichst gut zwischen echten und falschen Daten unterscheiden zu können. Dazu wird folgende Zielfunktion maximiert:

$$\max L_D = \log(D(x)) + \log(1 - D(G(z))) \quad (12)$$

x = Realer Datenpunkt

Da die Gleichungen 10 und 12 gegenläufig zueinander laufen, handelt es sich beim gesamten Algorithmus um ein Nullsummenspiel. Die Zielfunktion des Nullsummenspiels als Kombination der Gleichungen 10 und 12 lautet: [23]

$$\min_{G} \max_{D} V(D, G) = E_{x \sim p_{\text{Daten}}(x)}[\log(D(x))] + E_{z \sim p(z)}[\log(1 - D(G(z)))] \quad (13)$$

$p_{\text{Daten}}(x)$ = Reale Verteilung der Daten

$p(z)$ = Verteilung des Zufallsvektors

Zu beachten ist die Ähnlichkeit zu Gleichung 7, der Kreuzentropie. Da das Training nicht in einem Schritt ablaufen kann, wird es in zwei Teile geteilt. Zunächst wird der Discriminator trainiert, anschließend der Generator. Das Trainingsschema ist in Algorithmus 1 zu sehen.

Algorithm 1: Algorithmus eines GAN's (nach [23])

Data: Initiale Gewichte des Discriminators: w_0 , Initiale Gewichte des Discriminators: θ_0

Input : Hyperparameter: Anzahl der Epochen: e, Lernrate: η , Batchgröße: m, Iterationen von D: k, Iterationen von G: l

Output: Trainierte Netze G (Generator) und D (Discriminator)

```

1 for Anzahl der Epochen do
2     Generiere Batch künstlicher Daten
3     Wähle Batch realer Daten
4     Aktualisiere Discriminator mittels Gradientenabstiegsverfahren:
5          $g_w \leftarrow \frac{1}{m} \sum_{i=1}^m \log(D(x)) + \log(1 - D(G(z)))$ 
6          $w \leftarrow w + \eta * g_w$ 
7
8     Generiere Batch künstlicher Daten mit G aus  $p_z$ 
9     Aktualisiere Generator mittels Gradientenabstiegsverfahren:
10     $g_w \leftarrow \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z)))$ 
11     $\theta \leftarrow \theta + \eta * g_w$ 
12 end

```

Auch wenn das ursprüngliche (oder auch *vanilla* GAN) bereits hervorragende Ergebnisse, unter anderem bei der Generierung von Bildern, liefern kann, hat der obige Algorithmus mehrere Schwachstellen [23], [57]. Die wesentlichen Schwachpunkte sind die Folgenden:

1. Der Generator lernt nicht, da der Discriminator zu schnell zu gut ist und somit alle generierten Datensätze durch den Discriminator sofort "verworfen" werden [23]. Dies kann dem oben beschriebenen Wechsel von Gleichung 10 zu Gleichung 11 teilweise behoben werden.
2. Grundsätzliche Instabilität im Hinblick auf die verwendeten Hyperparameter [50]. Dies bedeutet, dass ein GAN auch bei gleichen Inputdaten bereits bei einer leichten Variation der Hyperparameter versagen kann.
3. *Moden Kollaps*: Dies bedeutet, dass der Generator nur wenige *Moden* der Realen Verteilung p_{Daten} erzeugt, trotz des veränderlichen Zufallsvektors. Die Generierung der immer gleichen ein bis zwei Bilder bei der Erzeugung von Bildern ist ein einfaches Beispiel für dieses Verhalten [23], [64].

Grundsätzlich sind GAN's schwierig zu trainieren, auch sind hierfür noch nicht alle Gründe genau verstanden [4].

Es existieren verschiedene Versionen, um eines (oder mehrere) der oben genannten Probleme zu beheben, das Training zu stabilisieren und die Qualität der synthetischen Daten zu verbessern. Dazu gehören unter anderem das DCGAN, das DRAGAN, das Fisher-GAN und das WGAN (mit der Erweiterung WGAN-GP) [50], [35], [42], [5], [26].

Eine weitere Modifikation der Generative Adversarial Networks ist das *Conditional* GAN (C-GAN) [40]. Mit diesem ist es möglich, Daten aus verschiedenen Klassen zu generieren, indem das ursprüngliche Label der Klasse beim Trainingsprozess des Discriminators verwendet wird. Zur Generierung wird eine Repräsentation des Klassenlabels, ein sogenanntes *Embedding* an den Zufallsvektor angehängt. Ebenso wird das Klassenlabel im Trainingsprozess des Discriminators mithilfe eines Embeddings verwendet. Auf die Stabilität des Trainingsprozesses oder auf die Qualität der Generierten Daten haben diese Modifikationen keinen nennenswerten Einfluss [40].

3.2.3 Wasserstein Generative Adversarial Networks

Wasserstein Generative Adversarial Networks sind eine Erweiterung der zuvor beschriebenen GAN's des Mathematikers Martin Arjovsky [5]. Diese erzielen besondere Verbesserungen im Hinblick auf das Problem des Modenkollapses und der Stabilität des Trainingsprozesses. Weiterhin existiert eine Weiterentwicklung mit der Bezeichnung WGAN-GP (Gradient Penalty) zur zusätzlichen Verbesserung des Wasserstein GAN's. Die theoretischen Grundlagen des WGAN-GP's werden im Folgenden erläutert, da in dieser Arbeit ein *Conditional-WGAN-GP* zur Erzeugung synthetischer Mikrostruktur verwendet wird.

Arjovsky hatte bereits in einer vorherigen Arbeit ([4]) festgestellt, dass der Hauptgrund der fehlenden Konvergenz von GAN's im fehlerhaften Verhalten der Lossfunktionen begründet liegt. Verschiedene Versionen der GAN's (Kapitel 3.2.2) haben verschiedene Lossfunktionen, dazu gehört etwa die *Kullback-Leibler Divergenz* zweier Verteilungen p_{Daten} und p_G :

$$KL(P_{Daten}, P_G) = \int \log \left(\frac{p_{Daten}(x)}{p_G(x)} \right) * p_{Daten}(x) * d\mu x \quad (14)$$

Zusätzlich auch die *Jensen-Shannon-Divergence*, welche sich aus der Kullback-Leibler-Divergenz ergibt:

$$JS(P_{Daten}|P_G) = KL(p_{Daten}|p_M) + KL(p_G|p_M) \quad (15)$$

$$p_M = (P_{Daten} + P_G)/2$$

Zudem kann auch die namensgebende *Wasserstein-Distanz* verwendet werden, um die Distanz zwischen zwei Verteilungen zu berechnen (Die genaue mathematische Formulierung ist hierbei zunächst nicht relevant). Dabei gilt, dass die Wasserstein-Distanz im Vergleich zu den Gleichungen 14 und 15 ein besseres Konvergenzverhalten aufweist, auch bei recht simplen Verteilungen. Als Beispiel hierzu dient die Differenz zwischen zwei 2-dimensionalen Verteilungen:

$$\begin{aligned} P_0 &= (0, Z) \\ P_1 &= (\theta, Z) \end{aligned}$$

$Z = U[0,1]$ (Gleichverteilt)

θ = Reellwertiger Parameter

(Bei beiden Verteilungen handelt es sich um eine Linie mit $x = \theta$ oder 0 und gleichverteilten Werten zwischen 0 und 1 auf der y-Achse)

Die Wasserstein-Distanz für diese beiden Verteilungen ist $|\theta|$, die Jensen-Shannon-Divergenz ist $\log 2$ für alle $\theta > 0$ und sonst 0. Dies zeigt die bessere Eignung der Wasserstein-Distanz für das Lernen dieses Verteilungsunterschiedes auf, da mit der Jensen-Shannon-Divergenz der Parameter θ unberücksichtigt bleibt [5]. Vertiefte Analysen der Eigenschaften und Grundlagen der Wasserstein-Distanz sind im Kapitel 3.3.1 zu finden.

Die Wasserstein-Distanz liegt in ihrer Grundform als Minimierungsproblem vor, welches als schwer lösbar gilt und hier nicht weiter betrachtet werden soll. Somit wird das Optimierungs- bzw Minimierungsproblem mittels der *Kantorovich-Rubinstein-Dualität* in eine besser lösbare Form überführt. Die Dualität ist eine grundlegende Eigenschaft eines Optimierungsproblems. Zu jedem Linearen Optimierungsproblem existiert ein duales Problem der in Tabelle 1 dargestellten Form

Tabelle 1: Primale und duale Form eines Linearen Optimierungsproblems

Primale Form		Duale Form
min:	$z = c^T * x$	max: $z = b^T * y$
mit:	$Ax = b$	mit $A^T \leq c$
	$x \geq 0$	

Die duale Formulierung ist dementsprechend nach Tabelle 1 ein Problem mit dem Ziel der Maximierung und lautet wie folgt:

$$W(P_{Daten}, P_G) = \sup_{\|f\|_L \leq 1} E_{x \sim p_{Daten}}[f(x)] - E_{x \sim p_G}[f(x)] \quad (16)$$

Der Ausdruck $\|f\|_L \leq 1$ in Gleichung 16 bedeutet, dass die Gleichung für alle 1-Lipschitzstetigen Funktionen gültig ist. Die Lipschitzstetigkeit ist eine verschärfe Form der allgemeinen Stetigkeit Mathematischer Funktionen. Allgemein liefert die Lipschitzstetigkeit eine obere Schranke des Änderungsverhaltens einer Funktion durch die Lipschitzkonstante K :

$$|f(x_1) - f(x_2)| \leq K * |x_1 - x_2| \quad (17)$$

Die Gleichung 15 ist grundsätzlich für alle K -Lipischitzstetigen Funktionen gültig. Dies ist dadurch begründet, dass jede K -Lipischitzstetige Funktion auch 1-Lipischitzstetig ist, wenn sie durch K dividiert wird (Analog dazu wird W mit K multipliziert) [5]

Im Kontext des Wasserstein-GAN's entspricht die Funktion f in Gleichung 17 dem Discriminator, da ein neuronales Netz im Mathematischen Sinne ebenfalls eine Funktion ist. Um die Lipschitzstetigkeit eines neuronalen Netzes sicherzustellen existieren verschiedene Möglichkeiten [47], [72]. Die einfachste Möglichkeit, welche auch in der Ursprünglichen Veröffentlichung von Arjovsky vorgeschlagen wurde, ist das Abschneiden (engl. clippen) der Gewichte des Discriminators nach der Backpropagation auf die Lipschitzkonstante K . Allerdings wirkt sich dies negativ auf die Qualität des GAN's aus, da das Abschneiden der Gewichte den Discriminator auf das Lernen vergleichsweise einfacher Verteilungen limitiert [5]. Eine bessere Möglichkeit besteht im sogenannten *Gradient Penalty* [26]: Bei diesem Verfahren wird ausgenutzt, dass die Norm der Gradienten einer 1-Lipschitzstetigen Funktion stets 1 ist [26]. Daher wird der folgende Strafterm an die Lossfunktion (Gleichung 16) angefügt:

$$GP = \lambda * (\|\nabla_{\hat{x}} f(\hat{x})\|_2 - 1)^2 \quad (18)$$

∇ = Nabla-Operator

$\|\dots\|_2$ = 2-Norm einer Matrix

λ = Ganzzahliger Strafparameter (10 lt. WGAN-GP Paper)

Die 2-Norm, oder auch Frobenius-Norm einer Matrix ist die Summe der Quadrate aller Einzelteile einer Matrix. Der Einsatz von Gleichung 18 ermöglicht den Verzicht auf das Abschneiden der Gewichte.

Algorithm 2: Algorithmus eines WGAN-GP (nach [26] und [5])

Data: Initiale Gewichte des Discriminators: w_0 , Initiale Gewichte des Generators: θ_0

Input : Hyperparameter: Anzahl der Epochen: e, Lernrate: η , Batchgröße: m, Reelwertiger Strafparameter: λ , Iterationen von D: k

Output: Trainierte Netze G und D

```

1 for Anzahl der Epochen do
2   for Anzahl Schritte k do
3     Wähle  $x \sim p_{Daten}$ ,  $z \sim p_z$ , eine Zufallszahl  $\epsilon \sim U[0, 1]$ 
4      $\tilde{x} \leftarrow G(z)$ 
5      $\hat{x} \leftarrow \epsilon x + (1 - \epsilon)\tilde{x}$ 
6      $g_w \leftarrow \frac{1}{m} \sum_{i=1}^m D(x) - \frac{1}{m} \sum_{i=1}^m D(G(z)) + \lambda * (\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2$ 
7      $w \leftarrow w + \eta * g_w$ 
8   end
9   Wähle  $z \sim p_z$ 
10   $g_w \leftarrow -\frac{1}{m} \sum_{i=1}^m D(G(z))$ 
11   $\theta \leftarrow \theta - \theta * g_w$ 
12 end

```

Im Vergleich der Algorithmen 1 und 2 werden die Unterschiede zwischen WGAN-GP und vanilla-GAN deutlich. Zusätzlich zur veränderten Lossfunktion ist die größte Änderung, dass der Discriminator häufiger trainiert wird als der Generator (z.B. im Verhältnis fünf zu eins). Dies ist möglich, da der Generator auch Lernen kann, wenn der Discriminator schon nahe am Optimum ist. Zudem muss ein weiterer Hyperparameter (λ) im Prozess berücksichtigt werden. Laut [26] kann dieser für Fälle der Bildgenerierung als 10 angenommen werden.

Im Vergleich zum ursprünglichen GAN löst das WGAN-GP einige Probleme, darunter:

- Es wird nahezu kein Moden Kollaps mehr beobachtet.
- Der Generator lernt auch, wenn der Discriminator schon austrainiert ist. Dies liegt daran, dass der Discriminator durch die verwendete Wasserstein-Distanz immer ausreichende Gradienten für das Training des Generators liefern kann.
- Durch den Austausch des Abschneidens der Gewichte durch das Prinzip "Gradient Penalty" kann die Stabilität weiter erhöht werden.

Ziel dieser Arbeit ist es, mittels eines Conditional-WGAN-GP zweidimensionale synthetische Mikrostruktur für verschiedene Materialien zu generieren. Dazu werden Feed-Foward-Netze als Discriminator und Generator verwendet.

3.3 Optimaler Transport

Dieses Kapitel ist als Erweiterung zu Kapitel 3.2.3 konzipiert. Insbesondere sollen vertiefte mathematische Grundlagen zur Wasserstein-Distanz, welche zum mathematischen Themengebiet des *Optimalen Transports* gehört, erläutert werden. Weiterhin soll eine Methode vorgestellt werden, mit welcher effizient der Transport, bzw. die Distanz zwischen zwei Verteilungen in Form von Punktwolken berechnet werden kann. In dieser Arbeit wird diese Methode dazu genutzt, das C-WGAN-GP auszuwerten, d.h. die reale und die synthetische Mikrostruktur zu vergleichen. Da ein Grundlegendes Verständnis der Wasserstein GAN's auch ohne dieses Kapitel möglich ist, ist es chronologisch nach Kapitel 3.2.3 zu finden.

Das Themengebiet des Optimalen Transports geht auf den Französischen Gelehrten Gaspard Monge zurück [41]. Er stellte die Frage, wie man einen Haufen Sand möglichst *kosteneffizient* (das bedeutet in diesem Fall mit möglichst wenig Aufwand) in ein Loch der selben Größe transportieren kann. Allgemein gesprochen geht es darum, bei einer gegebenen Ausgangsverteilung und einer gegebenen Endverteilung den günstigsten Transport, d.h. die günstigsten Überführung der Ausgangs- in die Endverteilung zu finden. Heute finden sich Konzepte des Optimalen Transports in vielen Feldern der Informatik und Ingenieurwissenschaften wieder, zum Beispiel bei der Bildanalyse und- verarbeitung.

3.3.1 Wasserstein Distanz

Monge formulierte das nach ihm benannte Problem erstmalig im Jahre 1781. Die mathematische Darstellung funktioniert wie folgt: Sandhaufen und Loch werden als zwei Wahrscheinlichkeitsdichtefunktionen μ und ν dargestellt. Die Überführung von μ in ν geschieht mithilfe des sogenannten "Push-Forwards" ($s_{\#}\mu = \nu$). Dies bezeichnet einen mathematischen Operator, welcher μ in ν "drückt". Die Gesamtgleichung zur Bestimmung des Optimalen Transports lautet wie folgt: (Die Kostenfunktion ist zunächst unspezifiziert) [41]

$$\inf_{s(x)} = \int c(x, s(x)) * d\mu(x) \quad (19)$$

$$\begin{aligned} s(x) &= s_\# \mu \\ c(x, s(x)) &= \text{Kostenfunktion} \\ \inf &= \text{Infimum (Minimum)} \end{aligned}$$

Gleichung 19 ist durch die Nichtlinearität von $s_\# \mu$ im Allgemeinen nur sehr aufwendig lösbar. [21] Über 150 Jahre später, 1942, schlug Kantorovich eine relaxiert (d.h. vereinfachte) Form des Problems vor. [32] Statt des Push-Forwards betrachtete er sogenannte Transportpläne $\pi \subset P(X, Y)$. Dabei ist $\mu \subset X$ und $\nu \subset Y$. Jeder Transportplan beschreibt eine zulässige Überführung von μ in ν . Das Gesamtset aller Transportpläne $\Pi(\mu, \nu)$ ist ein konvexer Unterraum von $P(X \times Y)$. [21] Für einen gültigen Transportplan müssen zudem jeweils folgende Nebenbedingungen gelten:

$$\pi(A \times Y) = \mu(A) \quad (20)$$

$$\pi(X \times B) = \nu(B) \quad (21)$$

(A und B sind jeweils Teilmengen von X und Y)

Die Verknüpfung von Gleichung 19 mit den obigen Änderungen ergibt die Formulierung des Optimalen Transportproblems nach Kantorovich: [32]

$$\inf_{\pi \in \Pi} = \int c(x, y) * d\pi(x, y) \quad (22)$$

Gleichung 22 ist linear und damit um einiges leichter zu lösen als Gleichung 19. Zudem ist die Formulierung von Kantorovich weniger restriktiv, da sie ermöglicht, dass Masse von *einem* Punkt in X zu *mehreren* Punkten in Y transportiert werden kann. [36] Die ursprüngliche Formulierung von Monge (Gleichung 19) ermöglicht nur den Transport von exakt einer Einheit Masse. Dadurch lassen sich durch diese Formulierung keine Verteilung mit weniger Punkten in eine Verteilung mit mehr Punkten überführen. [48] Mit Gleichung 22 ist dies jedoch möglich.

In beiden Gleichungen ist die Kostenfunktion unspezifiziert. Nun soll eine Distanz als Kostenfunktion betrachtet werden. Bei der Betrachtung des Problems auf dem Raum \mathbb{R}^d ($d = 1$ für z.B. ein Signal oder $d = 2$ für z.B. ein Bild) ergibt sich unter Verwendung der Kostenfunktion $|x - y|^p$ die sogenannte *p-Wasserstein Distanz*: [66] [36]

$$W_p(\mu, \nu) = \left(\inf_{\pi \in \Pi} \int |x - y|^p * d\pi(x, y) \right)^{1/p} \quad (23)$$

$p (\geq 1)$ = Ganzzahliger Parameter (meist 1 oder 2)

Die 1-Wasserstein-Distanz wird in der Informatik und in den Ingenieurwissenschaften häufig auch als *Earth Mover's Metrik* bezeichnet. Diese Bezeichnung nimmt Bezug auf die ursprüngliche Formulierung des Transportproblems von Monge.

3.3.2 Sinkhorn Iterationen - Entropische Regularisierung der Wasserstein Distanz

Es gibt verschiedene Möglichkeiten, das Problem des Optimalen Transportes auf numerischem oder analytischem Wege zu lösen. Eine Möglichkeit ist die Lösung eines linearen Programms. Zur Lösung solcher Programme gibt es verschiedene Verfahren, zum Beispiel das Simplex-Verfahren. [18] Diese Verfahren sind allerdings extrem Ressourcen- und Zeitintensiv. Bei der Berechnung des Optimalen Transport für zwei Punktfolgen der Größe d ist die Laufzeit etwa gleich $\mathcal{O}(d^3 \log(d))$. [46] Zur Erklärung der sogenannten "Big-O-Notation" dient das folgende Beispiel: Betrachtet wird die Funktion $T(d) = 18d^3 + 1000d$: Das Wachstumsverhalten dieser Funktion wird vom Term d^3 bestimmt, daher kann auch geschrieben werden: $T(d) \in \mathcal{O}(d^3)$. Die Notation wird häufig dazu genutzt, die Laufzeit von Algorithmen anzugeben.

Eine numerisch effizientere Lösung des optimalen Transportproblems besteht in der sogenannten entropischen Regularisierung der Wasserstein Distanz, auch Sinkhorn-Distanz genannt. [17] Diese soll im folgenden hergeleitet werden:

Das Konzept der Entropie in der Informationstechnik geht auf den amerikanischen Mathematiker Claude E. Shannon zurück [61]: Im Allgemeinen beschreibt sie den mittleren Informationsgehalt einer Nachricht. Für die in 3.3.1 beschriebenen Transportpläne ist die Entropie wie folgt definiert:

$$H(\pi) = - \sum_{i,j=1}^d p_{ij} * \log_2(p_{ij}) \quad (24)$$

H = Entropie einer Zufallsvariable (hier der Transportplan)

d = Größe der zu betrachtenden Verteilung

$p_{ij} = \pi(x = i, y = j)$

Durch die Kombination aus Gleichung 24 und Gleichung 23 entsteht die entropisch regularisierte Form der Wasserstein-Distanz (Hier in Matrix-Schreibweise):

$$OT_\epsilon = \min_{\pi \in \Pi(\mu, \nu)} \langle \pi C \rangle - \epsilon * H(\pi) \quad (25)$$

ϵ = Regularisierungsfaktor (≥ 0)

$\langle ., . \rangle$ = Frobenius-Skalarprodukt

C = Kostenmatrix (siehe Gleichung 23)

$H(\pi)$ = Entropie

Die Idee zur entropischen Regularisierung stammt aus folgender Beobachtung: Man stellte fest, dass die echten Muster in Transportproblemen nicht den optimalen entsprechen, sondern deutlich diffuser sind [48]. Dies bedeutet, dass die optimale Lösung sich auf insgesamt wenige Routen verlässt als in der Realität. Wird ein solches Problem als Matrix dargestellt, erhält man eine sogenannte *dünnbesetzte Matrix* (engl. sparse matrix), in der der Großteil aller Einträge null ist. Eine solche Matrix besitzt eine höhere Entropie, daher verschiebt die entropische Regularisierung die Lösung zu diffuseren Lösungen. Dies ist auch in Bild 8 zu erkennen. Es soll ein optimaler Transport von den blauen Kreisen zu den Orangenen Kreuzen bestimmt werden. Wird der Regularisierungsparameter größer, werden mehr Verbindungen genutzt, d.h es gibt weniger Einträge in der den Transport beschreibenden Matrix, welche null sind.

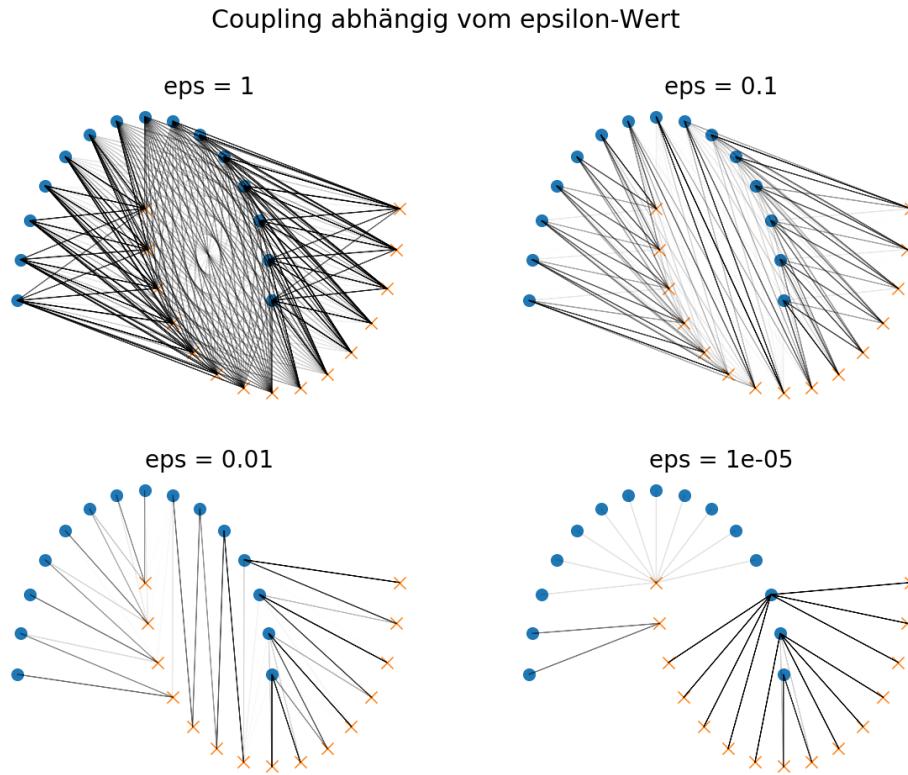


Bild 8: Coupling zweier Datensätze des “MakeMoons” Datensatzes aus der Scikit-Learn Bibliothek von Python mit verschiedenen ϵ . Es ist zu erkennen, dass mit steigendem ϵ die Anzahl der Verbindungen zwischen den Verteilungen zunimmt, das Bild also diffuser wird.

Die Regularisierung der Wasserstein-Distanz führt dazu, dass das gesamte zu lösende Problem nun strikt *konvex* ist, da der Entropieterm strikt konkav ist [36], [17]. Durch die Konvexität besitzt das Problem nun eine eindeutige Lösung. Diese hat folgende Form: [48], [17]

$$P_{ij} = u_i * K_{ij} * v_j \quad \forall (i, j) \in d * d \quad (26)$$

P = Lösung des OT-Problems (sog. "Coupling" Matrix)

d = Länge der zu vergleichenden Verteilungen

$K_{ij} = \exp\left(-\frac{C_{ij}}{\epsilon}\right)$

u/v = (unbekannte) Skalierungsvariablen

Zur Bestimmung der unbekannten Parameter u und v existiert ein Iteratives Berechnungsschema. Es wird nach dem Mathematiker Richard Sinkhorn als Sinkhorn-Iterationen bezeichnet, welcher als erster die Konvergenz der Iterationen nachweisen konnte [63]. Nach ihm wird Gleichung 25 auch als Sinkhorn-Distanz bezeichnet. Die

Parameter u und v werden schrittweise aktualisiert:

$$u^{l+1} = \frac{\mu}{K * v^l} \quad (27)$$

$$v^{l+1} = \frac{\nu}{K^T * u^{l+1}} \quad (28)$$

μ/ν = Zu vergleichende Verteilungen

l = aktueller Iterationsschritt

Die Berechnung endet, sobald ein Abbruchkriterium erreicht worden ist. Dies kann z.B. die Differenz $u^{l+1} - u^l$ sein. Erreicht die Differenz einen kritischen Wert, stoppt die Iteration.

Mit der Python-Bibliothek GeomLoss [19] steht eine effiziente Implementierung der Sinkhorn Distanz zur Verfügung. Mit dieser kann in kurzer Zeit der (regularisierte) Optimale Transport bzw. die Distanz zwischen n-Dimensionalen Punktwolken berechnet werden. Allerdings berechnet die Bibliothek eine leicht abweichende Form der Sinkhorn-Distanz, die sogenannte "unbefangene" (unbiased) Sinkhorn-Distanz. Diese ist nach Feydy [19] wie folgt definiert:

$$S_\epsilon = OT_\epsilon(\mu, \nu) - \frac{1}{2}OT_\epsilon(\mu, \mu) - \frac{1}{2}OT_\epsilon(\nu, \nu) \quad (29)$$

$OT_\epsilon(\mu, \nu)$ = Ursprüngliche Sinkhorn-Distanz (Gleichung 25)

Durch die entropische Regularisierung mit $\epsilon > 0$ ist der Term $OT_\epsilon(\mu, \mu)$ im Normalfall größer als 0.

Kapitel 4 Entwicklung der Algorithmen

Die vorliegende Arbeit lässt sich vereinfacht in zwei Teile gliedern: Teil Eins befasst sich mit der Weiterentwicklung eines Tools zur Generierung statistisch Repräsentativer zweidimensionaler Mikrostruktur auf Basis von EBSD-Aufnahmen. Diese dienen zur Erzeugung ebenfalls statistisch repräsentativer Volumenelemente (S-RVE), mithilfe derer z.b. Schädigungsprozesse oder Umformprozesse simuliert werden können. Solche Weiterführenden Untersuchungen sind jedoch nicht Teil dieser Arbeit.

Im zweiten Teil wird ein Verfahren entwickelt, mit welchem es möglich ist, Eigenschaften dreidimensionaler Mikrostruktur aus zweidimensionalen EBSD-Aufnahmen herzuleiten. Dies ist wichtig, um genauere Rekonstruktionen einer Mikrostruktur zu erhalten.

Im zweiten Abschnitt ist besonders darauf zu achten, dass diese Arbeit die Entwicklung eines Verfahrens bzw. Algorithmus zum Ziel hat. Daher gibt es keine trennscharfe Abgrenzung zwischen Entwicklung des Algorithmus und den Ergebnissen und der Diskussion dieser. Somit werden in Kapitel 4.2 die ersten Ansätze erläutert, weitere Verbesserungen und Änderungen auf Basis der Diskussion der ersten Ergebnisse werden in Kapitel 5 beschrieben.

4.1 Zweidimensionale Mikrostruktur

Ein am IEHK entwickeltes WGAN zur Erzeugung zweidimensionaler synthetischer Mikrostruktur liegt bereits vor. Mit diesem System ist es möglich, Mikrostruktur für RVE's zu generieren. (Für eine breitere Diskussion der Ergebnisse siehe Pütz-*et-al* [49]) Für dieses System wurden allerdings einige Schwächen identifiziert, welche im Rahmen dieser Arbeit behoben werden sollen. Zum einen ist es nicht möglich, in einem einzigen Trainingsdurchlauf das WGAN auf mehrere Materialien gleichzeitig zu trainieren. Wenn synthetische Mikrostruktur für mehrere Materialien erzeugt werden soll, ist jeweils ein Trainingsdurchläufe nötig. Dies verbraucht sowohl Zeit als auch Ressourcen auf dem verwendeten Computersystem. Weiterhin kann das aktuelle Verfahren keine Texturen erzeugen. Zudem lässt auch die allgemeine Qualität der Daten noch zu wünschen übrig, sodass auch hier Verbesserungen notwendig werden.

Auch die Auswertung des Systems bedarf einer Überarbeitung. Sobald das System für

die vorgegebene Anzahl an Epochen trainiert ist, wird zur weiteren Verwendung nur noch der Generator benötigt, der Discriminator wird nur im Trainingsprozess verwendet. Am Ende eines Trainingsprozesses liegen somit viele verschiedene Versionen des Generators vor (Die Versionen unterscheiden sich in ihren Gewichten und Bias). Die Auswertung beinhaltet die Auswahl der geeignetsten Version. Da hier je nach Trainingslänge viele tausend Versionen entstehen können, da theoretisch für jede Iteration des Generators eine einzigartige Version vorliegt, ist eine schnelle und effiziente automatische Auswertung nötig.

4.1.1 Aufbau des Conditional Wasserstein Generative Adversarial Network - Gradient Penalty

Zur Behebung der im vorherigen Abschnitt genannten Probleme wird das vorhandene WGAN zu einem Conditional WGAN mit Gradient Penalty erweitert. Die theoretischen Grundlagen dieses Algorithmus finden sich in Kapitel 3.2.3. Die benötigten Operatoren, Funktionen und Klassen können zwar direkt mit der gewählten Programmiersprache Python geschrieben werden, es stehen allerdings umfangreiche Bibliotheken zur Implementierung diverser Methoden und Konzepte des maschinellen Lernens zur Verfügung. Für diese Arbeit wurde das Framework *PyTorch* verwendet, welches von Facebook AI entwickelt wurde. [45] Da Python unter anderem durch die dynamische Typisierung von Variablen eine relativ langsame Ausführungsgeschwindigkeit besitzt, ist PyTorch in C++ geschrieben und ermöglicht mit der CUDA-API die Auslagerung von Berechnungen auf Grafikprozessoren. Dies ist hilfreich, da der Trainingsprozess hochgradig parallelisierbar ist und sich mit Grafikprozessoren deutliche Geschwindigkeitsvorteile erzielen lassen. Im Gegensatz zu TensorFlow (von Google) bietet PyTorch einige Vorteile: Zum eine ist die Erstellung von Modellen insgesamt näher an direkter Programmierung mit Python, was den Einstieg erleichtert. Zum anderen wird der Berechnungsgraph (engl. computational graph) im Gegensatz zu TensorFlow dynamisch zur Laufzeit des Programms erstellt. Dies erleichtert das Debugging.

Die vorhandenen Daten werden aus EBSD-Aufnahmen extrahiert. Mit dem Programm MTEX [6] werden die EBSD-Aufnahmen analysiert und die Körner als Ellipsen berechnet. Dabei werden nur Körner berücksichtigt, welche eine Fläche von mehr als 1 haben. Dies entspricht bei der verwendeten Auflösung des EBSD-Systems etwa einem Pixel. Für die Validierung des CWGANGP werden zu jedem Korn sechs Features bestimmt:

- Die Kornfläche (Area)
- Das Halbachsenverhältnis (Aspect Ratio)

- Die Neigung des Korns (Slope)
- Die Orientierung in den drei Eulerwinkeln

Andere Parameter, welche in Kapitel 3.1.2, wurden zwar ebenfalls mit MTEX bestimmt, wurden jedoch aus dem Trainingsprozess herausgelöst, da diese für die weitere Verwendung der Daten weniger relevant sind. In Bild 9 ist eine grafische Darstellung des DP800 in Form eines Pairplots, bei welchem alle Parameter eines Materials jeweils gegeneinander dargestellt werden, zu sehen. Zu erkennen ist die in Kapitel 3.1.1 erläuterte Verteilung von Area und Aspect Ratio, welche Ähnlichkeit zu einer Gammaverteilung hat. Die zwei Hochpunkte in der Verteilung des Slope-Parameters liegen bei etwa 0 und 180°, was bedeutet, dass die meisten Körner nicht stark geneigt sind. Die Verteilungen der Orientierungswinkel folgen keinem spezifischen Muster. Dies ist daran begründet, dass das Material keine stark ausgeprägte Textur hat.

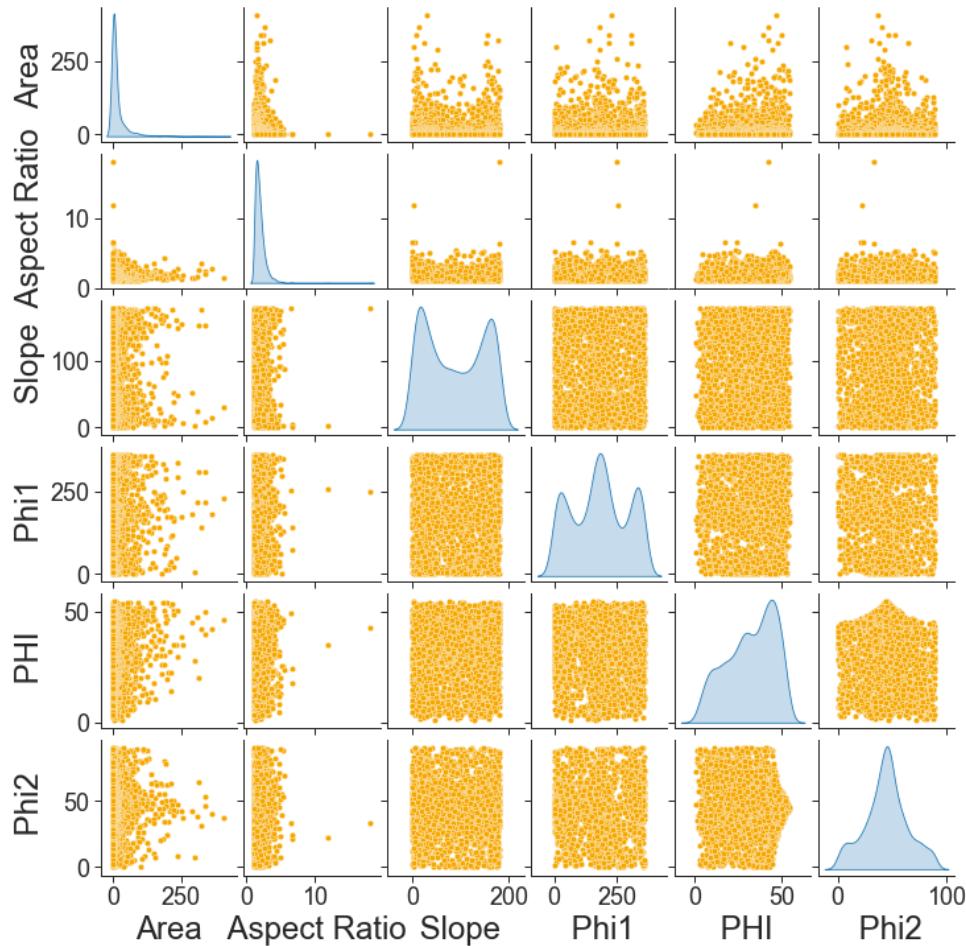


Bild 9: Ein Pairplot des DP800

Bei der Erstellung des WGANs wurde Wert auf eine möglichst einfache Benutzbarkeit gelegt. Zudem soll ein hoher Automatisierungsgrad möglich sein, damit Parameterstudien einfach durchgeführt werden können. Zudem wurde die Anzahl der wählbaren Hyperparameter erweitert. Der Trainings- und Evaluierungsprozess

läuft folgendermaßen ab: Zunächst wird ein Objekt der CWGANGP Klasse mit allen Hyperparametern instanziert. Im Anschluss wird die *train()* Methode aufgerufen und das Training beginnt. Nach Abschluss des Trainings wird die *evaluate()* Methode aufgerufen, welche alle im Laufe des Trainings erzeugten und abgespeicherten Versionen des Generators im Vergleich zu den realen Daten analysiert. Dabei ist zu beachten, dass die Analyse für jedes Material einzeln durchgeführt wird. Somit existiert für jedes Material eine optimale Generatorversion.

Die gewählten Hyperparameter sind für den Erfolg des Trainingsprozess entscheidend. Zusätzlich zu den bereits in der Ersten Version vorhandenen Hyperparameter wurden mehrere Hyperparameter zusätzlich hinzugefügt. Einige ergeben sich aus der veränderten Architektur, andere wurden zur weiteren Qualitätsverbesserung hinzugefügt.

1. *width-g*: Netzbreite des Generators. Jedes Hidden Layer des Generators hat diese Größe
2. *width-d*: Netzbreite des Discriminators.
3. *depth*: Tiefe beider Netzwerke, bzw. Anzahl der Hidden Layer
4. *num-features*: Anzahl der Features des Inputs. Bei den oben beschriebenen Datensätzen gleich sechs.
5. *n-classes*: Anzahl der Klassen/Materialien. Strenggenommen sind n-classes und num-features keine richtigen Hyperparameter, da sie von den Inputdaten abhängen.
6. *embed-size*: Größe der Labelcodierung. (default: 2)
7. *d-loop*: Anzahl an Iteration, die der Discriminator trainiert wird, vor einem Durchlauf des Generators
8. *gen-iters*: Iterationen des Generators bis zum Ende des Trainings. Dieser Parameter ist zur Steuerung des Trainings besser geeignet als die übliche Epochenzahl.
9. *p*: Dropout-Wahrscheinlichkeit. Prozentsatz, zu welchem ein Neuron des Discriminators während einer Iteration im Trainingsprozesses deaktiviert wird. Dies verhindert Overfitting.
10. *batch-size*: Anzahl der Daten, für die ein aggregierter Loss berechnet und eine Backpropagation durchgeführt wird.
11. *z-dim*: Länge des zur Generierung genutzten Zufallsvektors
12. *learning-rate*: Lernrate beider Netze

13. *lambda-p*: Strafparameter λ . (Vgl. Gleichung 18) Dieser Parameter ist für den Erfolg des Trainings entscheidend, da bei einem zu großen λ die Zielfunktion ignoriert wird und nur nach dem Strafterm trainiert wird.
14. *activationg/activationd* Aktivierungsfunktion von Generator bzw. Discriminator. Dieser Parameter ist nicht kontinuierlich, alle zu testenden Funktionen müssen zunächst implementiert werden.
15. *learning-rate*: Lernrate beider Netze. Bei (Wasserstein) Generative Adversarial Networks ist dieser Parameter typischerweise sehr klein.
16. *n-eval*: Schritte, nachdem die Version des Generators evaluiert wird. Der Quotient aus gen-iters und n-eval ist die Anzahl der Versionen, die evaluiert wird.
17. *optimizer*: Optimierungsalgorithmus beider Netze. Es existieren verschiedene Modifikationen der Gleichung 8, welche einen beschleunigten Trainingsprozess zum Ziel haben. (Auch dieser Parameter ist nicht kontinuierlich)
18. *beta1, beta2*: Parameter des *Adam-Optimierungsalgorithmus*. Genauere Informationen hierzu sind in Kapitel 5 zu finden.
19. *normalize*: Boolean Flag, welche anzeigt, ob Batch-Normalization im Generator verwendet wird. Dabei handelt es sich um eine Technik zur Stabilisierung und Beschleunigung des Trainings neuronaler Netze.

Da bereits in einer vorherigen Arbeit die Auswirkungen von Teilen des Parametersatzes auf die Ergebnisse untersucht wurden und die Ergebnisse trotz des Wechsels der Architektur zumindest begrenzt übertragbar sind, soll in dieser Arbeit der Fokus auf zusätzlichen Hyperparametern liegen. Dazu zählen im Besonderen die Aktivierungsfunktionen und die Optimizer und zugehörigen Parameter.

Weiterhin ist beim Studium der Hyperparameter zu beachten, dass einige hiervon auch Einfluss auf die Laufzeit des Trainings haben. Daher ist auch ein akzeptabler Kompromiss zwischen Laufzeit und Qualität zu finden.

4.1.2 Prinzipien der Auswertung

Im Vergleich zu vorherigen Arbeiten wurde die Auswertemethodik im Rahmen dieser Arbeit grundlegend überarbeitet. Die Auswertung wurde bisher in drei Schritten durchgeführt: Zunächst wurde eine Berechnung der Abweichungen zwischen den jeweiligen Randverteilungen (d.h. für jedes Feature einzeln) über die Differenz der Kerndichteschätzer (KDE's) durchgeführt. Da so die Interdependenzen der Randverteilungen untereinander unberücksichtigt bleiben, wurden zusätzlich eine Clusteranalyse und eine Funktionenanalyse durchgeführt. Bei beiden Verfahren ist allerdings die manuelle optische Bewertung der Ergebnisse nötig. Dies ist für eine Bewertung von unter Umständen mehreren hundert Versionen nicht praktikabel. Um alle geforderten Punkte bei der Auswertung abzudecken wird nun die in Kapitel 3.3.2 theoretisch erläuterte *Unbiased-Sinkhorn-Distance* verwendet. Durch die Verwendung von GPU und effizienten Algorithmen in der *GeomLoss*-Bibliothek können so auch viele Versionen schnell hintereinander direkt nach dem Trainingsprozess ausgewertet werden.

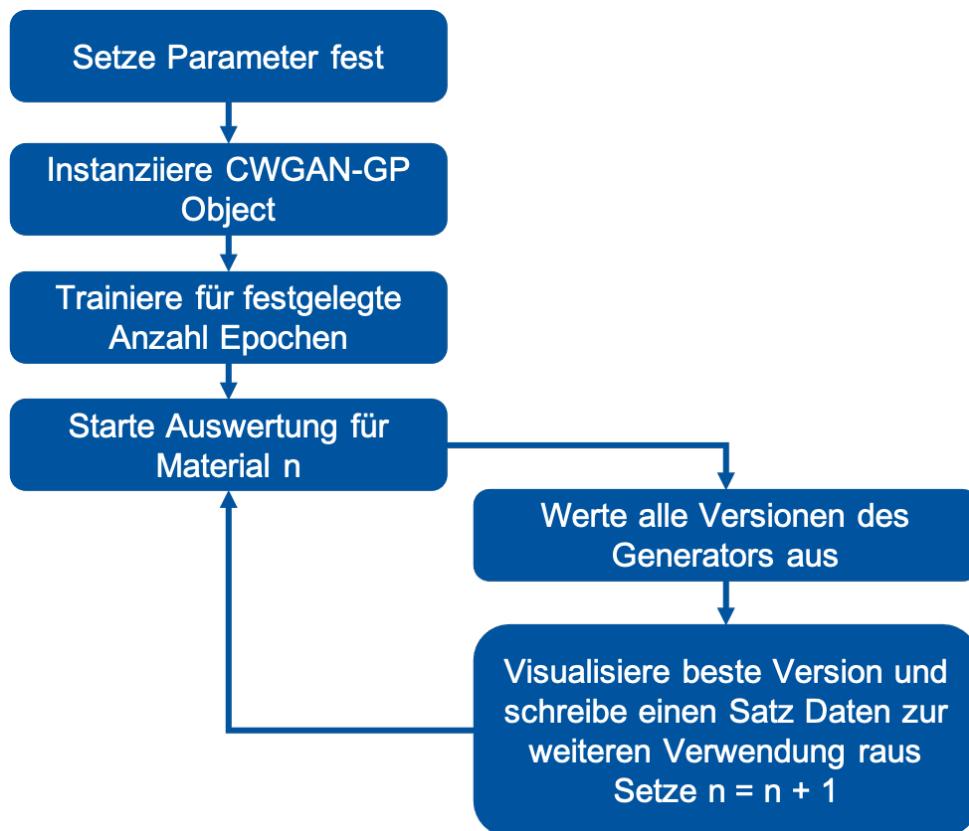


Bild 10: Grafische Darstellung des Trainings- und Auswerteprozesses des CWGAN

Bild 10 verdeutlicht den Auswertevorgang. Es wird deutlich, dass nach einem einzelnen, zusammenhängenden Trainingsprozess für jedes Material eine einzelne Auswertung durchgeführt wird. Dadurch wird ermöglicht, dass für jedes Material die beste Version des Generators gespeichert wird.

Das in dieser Arbeit verwendete C-WGAN GP basiert auf vielen Stellen auf Zufallszahlen bzw. zufällige Reihenfolgen von Prozessen und Abfolgen. Daher sind auch die Ergebnisse von Zufallszahlen abhängig. Da sich im Rahmen von Testläufen herausgestellt hat, dass der Sinkhorn-Loss sehr sensibel auf kleinste Änderungen reagiert, ist es wichtig, alle Einflüsse des Zufalls zu eliminieren. Dies ermöglicht zudem eine exakte Reproduzierbarkeit der Ergebnisse. Ermöglicht wird diese Reproduzierbarkeit durch das Setzen der "Seeds" der Random Number Generators (RNG's) der einzelnen Module. Das Setzen der RNG's bedeutet eine exakte Reproduzierung der Zufallszahlen in mehreren unabhängigen Durchläufen. Dazu werden im Code bei der Initierungs-Methode des CWGAN-GP folgende Befehle eingefügt.

Listing 4.1: Setzen der Random Number Generators

```

1      torch.backends.cudnn.deterministic = True
2      random.seed(0)
3      torch.manual_seed(0)
4      torch.cuda.manual_seed(0)
5      np.random.seed(0)

```

Zeile 1 im Codebeispiel bedeutet, dass das cudNN-Backend (ein Befehlsatz für die Berechnung neuronaler Netze auf Grafikprozessoren) ausschließlich deterministische Befehle verwenden soll. Mit den restlichen Befehlen werden die RNG's aller verwendeten Module auf null gesetzt.

Um die Auswertung zu präzisieren, wird vor dem Prüfen jedes einzelnen Generators der Seed von PyTorch wieder auf null gesetzt. Dadurch wird zur Generierung eines synthetischen Datensatzes, welcher mit den Realen Daten verglichen wird, immer der exakt gleiche Zufallszahlenvektor verwendet. Dies eliminiert jeglichen Einfluss des Zufalls. Auch wenn der Zufallsvektor aus einer Normalverteilung mit Mittelwert 0 und Standardabweichung 1 erzeugt wird, sind bei einer Länge im Bereich 128 - 512 auch sichtbare Abweichungen möglich. Für das Training werden echte (Pseudo)-Zufallszahlen verwendet.

4.1.3 Materialien

Zur Validierung muss das CWGAN in der Lage sein, verschiedene Materialien zu erkennen und synthetische Mikrostruktur zu generieren. Dazu wurden im Rahmen dieser Arbeit insgesamt sieben Materialien analysiert:

1. Ein Dualphasenstahl DP800 von thyssenkrupp (tk) (Bild 9)
2. Ein DP800 von Arcelor-Mittal (AM)
3. Ein Einsatzstahl (16MnCrS5)

4. Ein Hochmanganstahl mit rein austenitischem Gefüge
5. Derselbe Hochmanganstahl analysiert nach einer Dehnung von 35%
6. Ein Interstitial-Free (IF-Stahl)
7. Ein Dualphasenstahl DP1000

Die ausgewählten Materialien decken dabei ein breites Spektrum ab. Es soll somit getestet werden, ob das System sowohl sehr feine Unterschiede, wie den unterschiedlichen Hersteller beim DP800, als auch komplett verschiedene Materialien erkennen kann. Da die Materialien insgesamt keine stark ausgeprägte Textur aufweisen, wurde ein bereits um 35 Prozent gedeckter Hochmanganstahl hinzugefügt. Eine genauere Analyse der Materialien wie Zusammensetzung oder Herstellungsprozess ist in dieser Arbeit nicht nötig, da die Materialien nur als Validierung für das CWGAN dienen und größtenteils nicht weiteruntersucht werden sollen. Aus Platzgründen können die EBSD-Aufnahmen der Materialien im Anhang nachgeschlagen werden.

Die Datengrundlage bilden die EBSD-Aufnahmen. Aus diesen Aufnahmen können mit dem MATLAB-Addon MTEX die in Kapitel 3.1.3 beschrieben Parameter extrahiert und in einem tabularen Datenformat (.csv oder .h5) gespeichert werden.

4.2 Dreidimensionale Mikrostruktur

Ziel ist die Herleitung von Parametern dreidimensionaler Mikrostruktur aus zweidimensionalen EBSD-Aufnahmen. Dadurch kann die aufwendige Analyse dreidimensionaler Mikrostruktur durch das in Kapitel 3.1 beschriebene serial-sectioning-Verfahren vermieden werden.

4.2.1 Verfahrensentwicklung

Als Grundlage dienen EBSD-Aufnahmen, welche aus unterschiedlichen Richtungen angefertigt werden. In Bild 11 sind die Richtungen in einem Blechelement zu sehen. Dabei ist "RD" die Walzrichtung, "ND" die Blechnormale (auch BN) und "TD" die Quer-richtung. Somit existieren drei verschiedene EBSD-Aufnahmen:

1. $RD \times TD$ - Blick von oben
2. $RD \times BN$ - Blick von der Seite
3. $TD \times BN$ - Blick von vorne

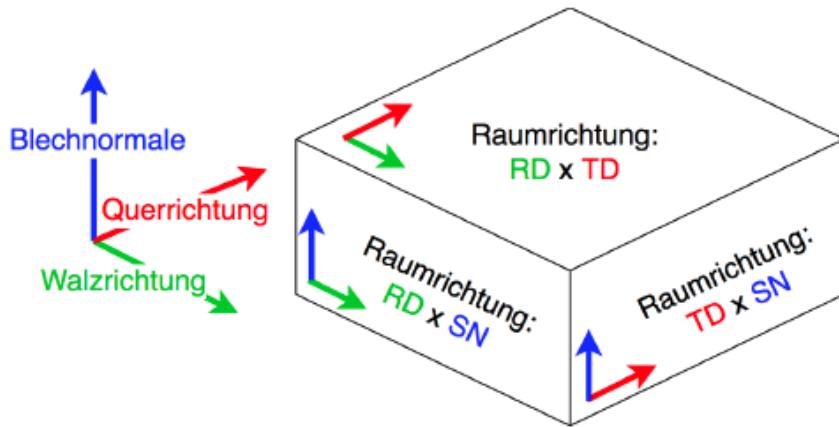


Bild 11: Richtungen an einem Blechelement

Gesetzt die Annahme, dass ein Korn näherungsweise als Ellipsoid mit dem Achsentupel (a,b,c) beschrieben werden kann, liegt jede Achse entlang einer der drei Richtungen. Die genaue Zuweisung kann willkürlich gewählt werden. Für diese Arbeit wurde a in Transverse Direction, b in Normal Direction und c in Rolling Direction gewählt. Das zweidimensionale Gegenstück zum Ellipsoid ist die auf den einzelnen EBSD-Aufnahmen zu sehende Ellipse, die einem Schnitt durch die Körner entspricht. Somit sind auf der EBSD-Aufnahme TDxBN die Achsen a und b zu sehen, auf der Aufnahme RDxTD die Achsen c und a und auf der Aufnahme RDxBN die Achsen c und b .

Zu beachten ist, dass durch solche Aufnahmen kein Korn von allen Richtungen betrachtet wird, d.h. auf allen Aufnahmen sind unterschiedliche Körner zu sehen. Um ein Korn genau zu bestimmen, müsste durch jedes einzelne Korn exakt mittig geschnitten werden, was durch eine einzelne EBSD-Aufnahme für mehrere Körner unmöglich ist. Daher ist nur eine möglichst statistisch repräsentative Bestimmung der fehlenden Dimension möglich. In der Theorie kann die Bestimmung folgendermaßen erfolgen, falls zu jedem Korn die Haupt- und Nebenachse des Ellipsenfits bekannt sind: (In diesem Beispiel c aus a hergeleitet)

1. Wähle zufälliges Korn mit a und b aus TDxBN-Aufnahme
2. Wähle Korn mit gleichem a aus RDxTD-Aufnahme und bestimme passendes c
3. Füge Tupel (a,b,c) der Kornliste hinzu und gehe zu Schritt 1.

Dies bedeutet im Wesentlichen, dass in zwei Aufnahmen eine gleiche Länge a gesucht wird und mit den jeweils zugehörigen Längen b und c ein Ellipsoid bildet.

Dieses Verfahren birgt allerdings einige Probleme: Zuvordest kann es zu einer definierten Länge eines Korns a auch mehrere korrelierende c 's in der zweiten Aufnahme geben, da die Korngrößen zwar einer Verteilungsfunktion folgen, allerdings ein einzeln herausgegriffener Wert nicht zwangsläufig statistisch repräsentativ sein muss.

Diesem Problem kann durch die doppelte Bestimmtheit jeder Achse entgegengewirkt werden. Im obigen Verfahren bleibt die RDxBN (mit den Achsen c und b) Aufnahme unberücksichtigt. Da mit dem beschriebenen Verfahren auch c aus b aus dieser Aufnahme hergeleitet werden kann, können beide Verfahren sich gegenseitig bestätigen. Dies gelingt wie folgt:

1. Wähle zufälliges Korn mit a und b aus TDxBN
2. Wähle Korn mit gleichem a aus RDxTD und bestimme passendes c
3. Wähle Korn mit gleichem b aus RDxBN und bestimme passendes c
4. Sind beide c's gleich, dann füge das Tupel (a,b,c) zur Kornliste hinzu und gehe zu Schritt 1

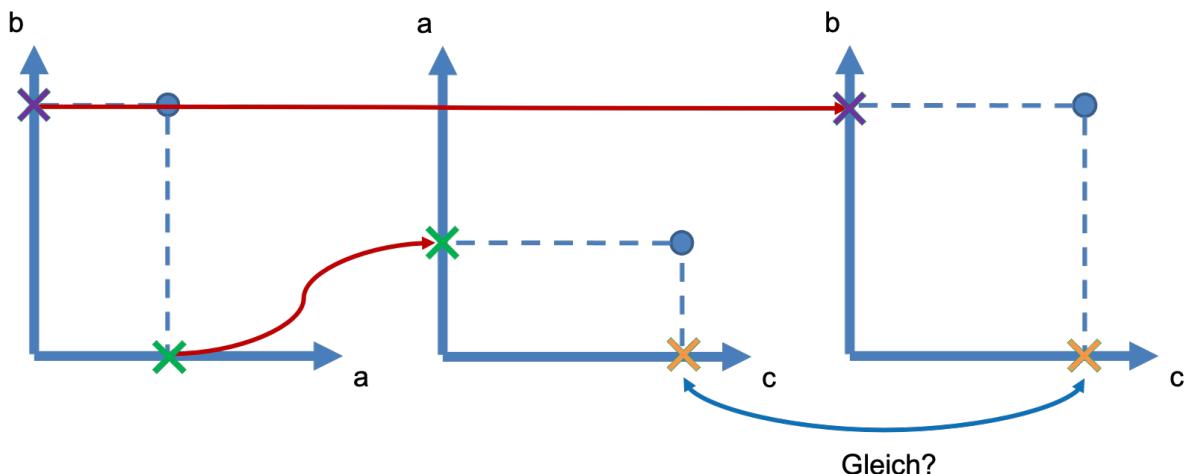


Bild 12: Darstellung des Prinzips zur Rekonstruktion dreidimensionaler Körner aus zweidimensionalen Aufnahmen

Mit dieser Erweiterung werden die beiden c's gegenseitig validiert. Dies steigert die Genauigkeit der rekonstruierten Körner, da statistische Ausreißer entfernt werden. Das vollständige Prinzip ist in Bild 12 zu sehen. Die gleichen Werte für die Halbachse a sind durch das grüne Kreuz dargestellt, die Werte für b in Rot und die Werte für c in Orange. Die drei Graphen symbolisieren die jeweiligen a x b / c x a und c x b Aufnahmen. Mittels der roten Pfeile werden die Punkte 2 und 3 des obigen Vorgehens illustriert. Liegen nun beide so berechneten c-Werte (orange Kreuze) innerhalb eines vorher festgelegten Intervalls, werden alle drei Mittelwerte zu einem Ellipsoid zusammengefasst.

Als letztes Problem bleibt die Neigung der Körner. Bild 13 zeigt eine EBSD-Aufnahme eines Hochmanganstahl mit zusätzlich dargestellten Ellipsen. Deutlich wird, dass die Ellipsen nicht gerade im Raum liegen, sondern gegenüber dem Koordinatenursprung geneigt sind. Dieser Winkel wird als Slope bezeichnet (vgl. Kapitel 3.1.2). Werden aus

einer solchen Aufnahme mithilfe der Software MTEX die Parameter der Ellipsen extrahiert, liegen diese in der Form (Hauptachse, Nebenachse, Neigung) vor. Dies bedeutet, dass die größere der Achsen immer in der ersten Spalte liegt, unabhängig davon, ob sie in x oder y-Richtung liegt. Bei einer Aufnahme, bei welcher a in x-Richtung und b in y-Richtung geht, müssten die Parameter Hauptachse und Nebenachse bei einem Winkel von 90° getauscht werden um die korrekten Achsenzuweisungen zu erhalten.

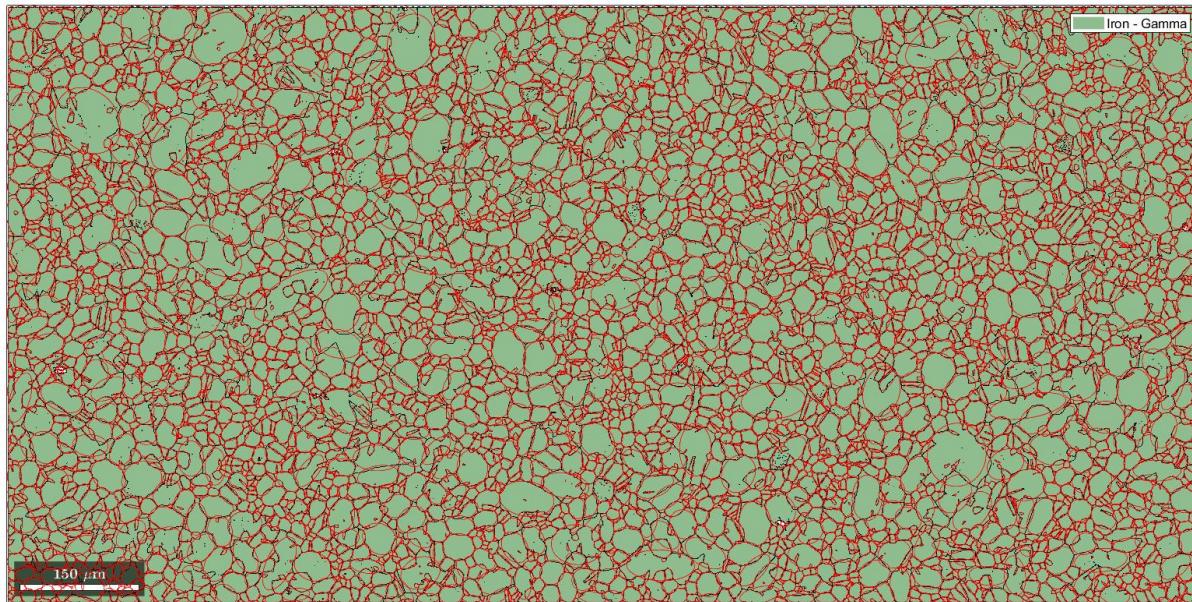


Bild 13: Ellipsenfit durch MTEX am Beispiel des Hochmanganstahls

Ein einfaches Tauschen ist allerdings nur bei einer Drehung von 90° möglich. Bei einem Winkel zwischen $0 - 90^\circ$ und $90 - 180^\circ$ muss ein anderer Ansatz gewählt werden. Eine Möglichkeit ist es, ein Intervall von Winkeln auszuwählen, bei welchem ein Tausch der Achsen durchgeführt wird, statt nur bei einer exakten Neigung von 90° . Dies Intervall kann z.b. von $45^\circ - 135^\circ$ gewählt werden. Dieser Ansatz ist allerdings recht willkürlich und stark von der Wahl des Intervalls abhängig. Es ist zwar naheliegend, dass bei einer Neigung im Bereich $80 - 100^\circ$ ein Tausch der Halbachsen sinnvoll ist, darüber hinaus herrscht allerdings keine Eindeutigkeit.

Sinnvoller ist die Charakterisierung eines zweidimensionalen Korns anhand seiner Anteile in x- und y-Richtung. Mittels des Neigungswinkels kann jede der Halbachsen in seine jeweiligen Komponenten zerlegt werden. Schematisch ist dies in Bild 14 dargestellt. Die Addition der Bestandteile x_1 und x_2 und y_1 und y_2 ergibt den x-, respektive y-Anteil der Ellipsenachsen.

Die Gleichungen zu diesem Verfahren lauten wie folgt.

$$\begin{aligned} \text{sum}_x &= |\cos \alpha| * a + \sin \alpha * b \\ \text{sum}_y &= \sin \alpha * a + |\cos \alpha| * b \end{aligned}$$

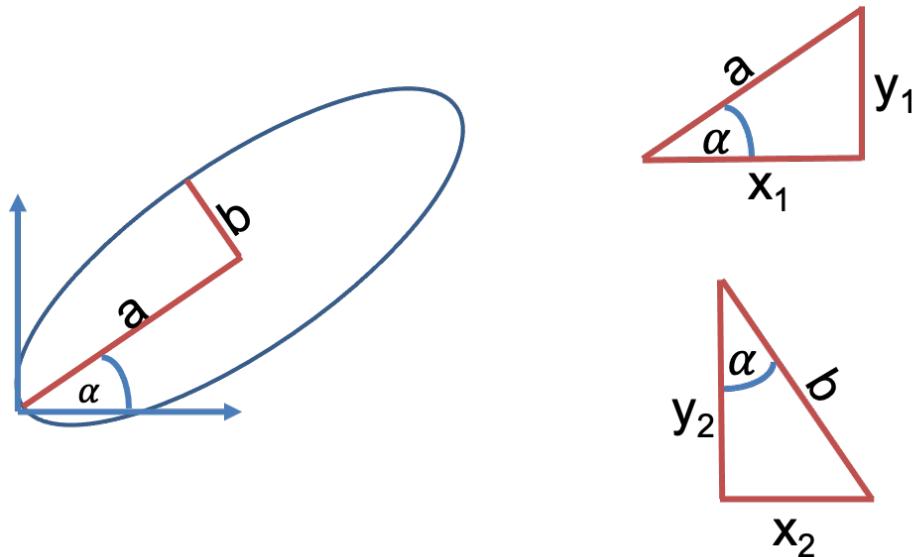
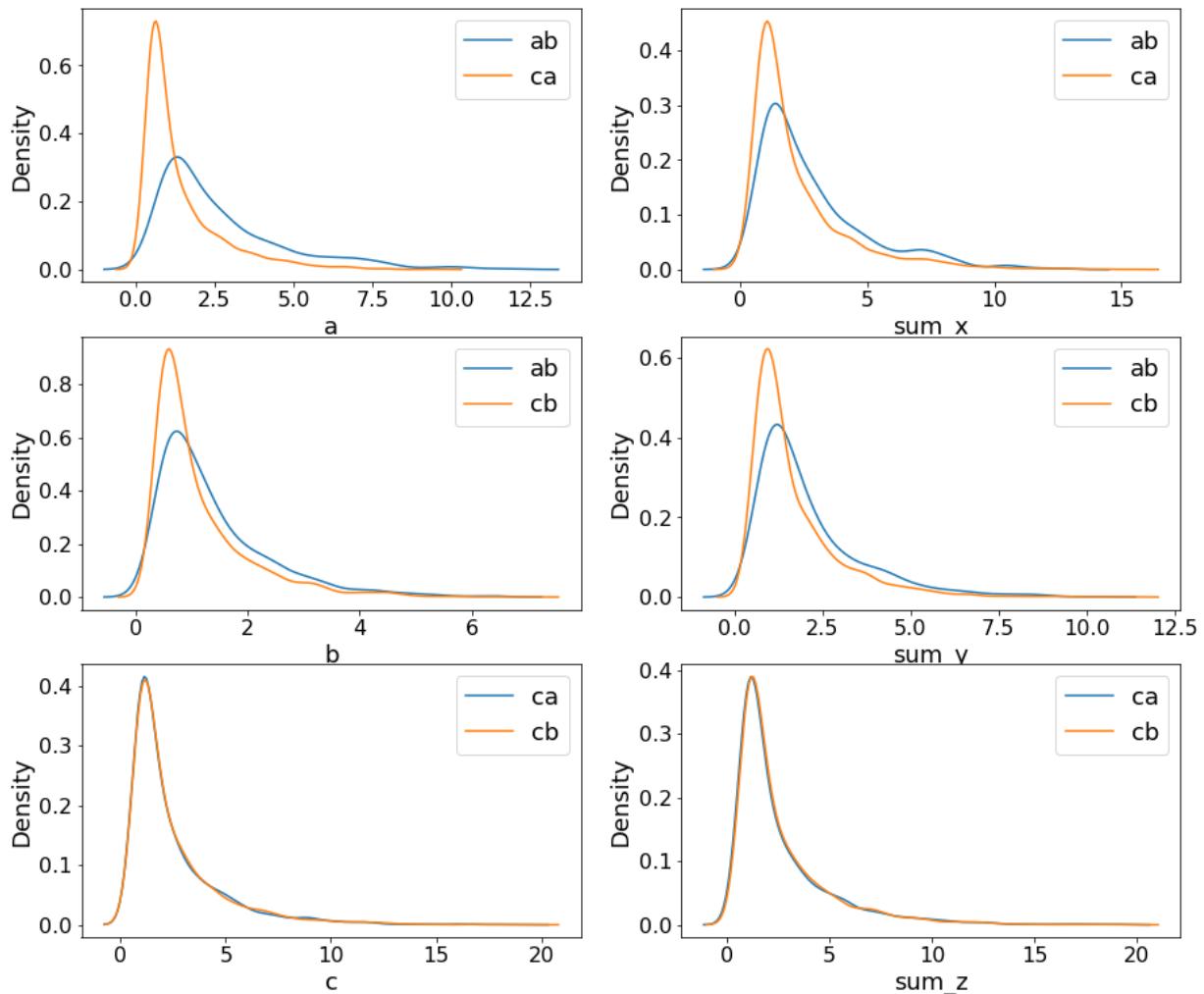


Bild 14: Illustrierung der Winkelbeziehungen einer mit dem Winkel α geneigten Ellipse

Es ist nötig, den Betrag von $\cos \alpha$ zu nehmen, da z.B. 0° und 180° für diese Rechnungen analog sind, mathematisch allerdings 1 und -1 ergeben. Mit diesen Gleichungen ist bei einem Winkel von 0° (oder auch 180° analog) die x-Komponente gleich a und analog bei einem Winkel von 90° die x-Komponente gleich b. Dies entspricht dem zuvor diskutierten Tauschen der Achsen bei 90° . Wird nun jedes Korn auf diese Weise zerlegt, sind in dieser Charakterisierung beide Achsen sowie die Neigung enthalten und somit alle relevanten Informationen. Zusätzlich kann die Sinnhaftigkeit der Achsenumrechnung an Bild 15 verdeutlicht werden. Es sind die jeweiligen Halbachsen (links) und die Summen der Bestandteile (rechts) als Kerndichteschätzungen aufgetragen. Zu erkennen ist, dass die Verteilungen aus den unterschiedlichen Aufnahmen zwar voneinander abweichen, allerdings insgesamt recht ähnlich verlaufen. Auch wird deutlich, dass bei der Verwendung der Summierten Halbachsenbestandteile (unter Berücksichtigung der Neigung) die Verteilungen besser aufeinander liegen (Diagramme auf der rechten Seite). Besonders deutlich wird dies für Halbachse a im linken oberen Diagramm: Die Abweichung ist hier deutlich stärker ausgeprägt als beim Vergleich der zerlegten Achsen rechts daneben. Hauptgrund dafür ist die Darstellung der Achsen durch MTEX: Während bei der Aufnahme/dem Datensatz axb a immer die größere der beiden Achsen ist, ist a bei cxa immer die kleinere der beiden Achsen. Dies erklärt, warum die Verteilung von a aus axb sowohl ihr Maximum bei höheren Werten hat, als auch der Peak zu höheren Werten verschoben ist.

Zuletzt müssen zwei *Eingangsparameter* für den Algorithmus definiert werden. Diese beiden sind die *Range*, in welcher ein passendes a bzw. b im zweiten/dritten Bild liegen darf und die *Abweichung (Thershold)*, die die beiden gefundenen c's zueinander haben dürfen. Beide Parameter sind nötig, da es nahezu unmöglich ist, in einer anderen Aufnahme einen exakt gleichen Wert zu finden, da wie bereits erläutert, verschiedene

Achsen und Achsenbestandteile

**Bild 15:** Achsen und Achsenbestandteile der einzelnen EBSD-Aufnahmen

Körner in jeder Aufnahme zu sehen sind. Eine Möglichkeit ist, die Werte aufsteigend zu sortieren und anschließend n Punkte vor und nach dem Wert zu wählen und als *nahe genug* für ein Korrelieren der Körner zu betrachten. Schwachpunkt dieses Ansatzes ist, dass, insbesondere wenn es nur wenige Körner in einer gewissen Größenordnung gibt, die Unterschiede sehr groß und nicht mehr vernachlässigt werden können. Besser ist es, beide Werte als prozentuale Abweichung zu definieren. Zusammengefasst ergeben all diese Erläuterungen den Algorithmus in Darstellung 3:

Algorithm 3: Vollständiger Algorithmus zur Rekonstruktion dreidimensionaler Mikrostruktur aus zweidimensionalen EBSD-Aufnahmen

Data: tabulare Datensätze mit Hauptachse, Nebenachse und Neigung aus allen drei Richtungen (TDxBN als axb , RDxBN als cxb , RDxTD als cxa)

Anzahl der zu rekonstruierenden Punkte: $npoints$

Akzeptable Abweichung beider c-Werte: $threshold$

Akzeptable Abweichung von a bzw. b: $range$

- 1 Rauslöschen verwendeter c Werte aus cb/ca: $drop$ (ja oder nein)
- 2 **Result:** Liste mit Körnern in der Form $(x,y,z, \alpha, \beta, \gamma)$
- 3 Vor dem Start:
- 4 Berechne für jeden Datensatz sumx und sumy aus den Halbachsen.
- 5 Setze dann $x \leftarrow a$ | $y \leftarrow b$ | $z \leftarrow c$
- 6 Speichere dann die Datensätze neu als xy, zy, zx
- 7 $i \leftarrow 0$
- 8 $its \leftarrow 0$
- 9 **while** $i <= npoints$ **do**
- 10 Wähle Punkt $(x, y) \sim xy$
- 11 Suche mit x alle (z_1, x) mit x in $[range, range]$ aus zx
- 12 Suche mit y alle (z_2, y) mit y in $[range, range]$ aus zy
- 13 Berechne die Differenz $d^i \leftarrow |z_1^i - z_2^i|$ für alle i
- 14 **if** $min(d) <= threshold$ **then**
- 15 $z_{neu} \leftarrow mean(z_1, z_2)$
- 16 Füge Tupel (x, y, z_{neu}) der Kornliste hinzu
- 17 **if** $drop = True$ **then**
- 18 Lösche verwendete Punkte (z, y) und (z, x) aus zy und zx
- 19 **else**
- 20 behalte Punkte
- 21 **end**
- 22 $i \leftarrow i + 1$
- 23 $its \leftarrow its + 1$
- 24 **else**
- 25 $its \leftarrow its + 1$
- 26 **end**
- 27 Lösche Punkt (x, y) aus xy
- 28 **end**

Dieser Algorithmus arbeitet mit der Extrapolation von z aus x und y. Dies ist nur ein Beispiel, jede Kombination ist möglich. Grundsätzlich ist im Idealfall der Datensatz

als Grundlage der Extrapolation zu wählen, welcher die wenigsten Körner besitzt, da dies der limitierende Faktor ist. Je mehr Körner in den Datensätzen vorhanden sind, desto genauer werden die Zuweisungen.

In dieser Arbeit werden die Verfahren am Beispiel eines Dualphasenstahls DP800 (tk) erläutert. Per EBSD wird hier allerdings nur die ferritische Matrix analysiert, die Analyse der Martensitbänder/Insel erfolgt in anderen Studien außerhalb dieser Arbeit. Um möglichst viele Körner zu erhalten wurden teilweise mehrere EBSD-Aufnahmen aus derselben Richtung zusammengefasst. Als Basis liegen alle Daten in einer .csv-Datei vor:

1. 750 Körner aus TDxBN (axb)
2. 2500 Körner aus RDxTD (cxa)
3. 2900 Körner aus RDxBN (cxb)

Auch hier werden - analog zu Kapitel 4.1.1 - nur Körner mit einer Fläche von größer $1\mu\text{m}^2$ berücksichtigt.

4.2.2 Validierung des Algorithmus

Der im vorhergegangenen Kapitel vorgestellte Algorithmus muss auf seine Eignung hin überprüft werden, die reale Dreidimensionale Mikrostruktur nachzubilden. Für eine solche Validierung stehen grundsätzliche mehrere Vorgehensweisen zur Verfügung:

1. Die Validierung über ein reales, per serial-sectioning-Verfahren erzeugten Gefüges
2. Die Validierung mittels Simulationen von RVE's
3. Die Validierung mittels einen synthetisch erzeugten Gefüges

Punkt 1 ist die direkte Validierung ohne Umwege und somit grundsätzlich vorzuziehen. Es ist darauf zu achten, ein genügend großes Gefügebestandteil zu analysieren, um statistische Repräsentativität zu gewährleisten. Allerdings liegt für den DP800 kein solcher Datensatz vor, daher muss eine Validierung anderweitig erfolgen. Bei einer wie in Punkt 2 gestalteten Validierung würden Zugversuche durchgeführt und Kraft-Weg-Kurven aufgenommen. Dazu werden RVE's simuliert und geprüft, ob RVE's mit der durch den Algorithmus erzeugten Mikrostruktur bessere Ergebnisse liefern als RVE's mit einer dreidimensionalen Mikrostruktur aus Rotationsellipsoiden (mit der Annahme $a = b$). Dazu müssen allerdings die anderen für eine Simulation von hoher Qualität notwendigen Parameter stimmen, z.B. die Schädigung oder die

korrekte Beschreibung anderer Phasen (Im Falle des DP800 insb. der Martensit). Da auch dieses im Rahmen dieser Arbeit nicht gewährleistet werden kann, scheidet auch diese Valdiierungsmethode aus.

Punkt 3 beschreibt im Wesentlichen eine Validierung wie in Punkt 1 über den Umweg einer künstlichen Gefügeerzeugung. Da von einem selbständig erzeugten Gefüge alle Eigenschaften (Ellipsoidgröße, Neigung etc.) bekannt sind, können von einem solchen Gefüge "Quasi-EBSD"-Aufnahmen angefertigt werden, auf welche nun Algorithmus 3 angewandt wird. Die so erzeugte Mikrostruktur kann nun mit der synthetischen Struktur verglichen werden. Das künstliche Gefüge besteht hier aus einem mit Ellipsoiden verschiedener Größen und Neigungen gefüllten dreidimensionalen Grid. Die Erzeugung des Gefüges ist der Erstellung der Dreidimensionalen RVE's aus *Henrich et-al.* nachempfunden, das Kornwachstum wurde allerdings weggelassen [29].

Auch dieser Algorithmus wurde in Python mit Rückgriff auf PyTorch und anderen Bibliotheken geschrieben. Zunächst wird ein Ellipsoid als diskrete Punktwolke erzeugt und an einer zufälligen Stelle im Grid platziert. Sukzessive werden weitere Ellipsoiden im Raum platziert, dabei wird für jedes Ellipsoid geprüft, ob ein Schnittpunkt mit bereits platzierten Ellipsoiden vorliegt. Dazu wird über einen *intersect*-Befehl abgeprüft, ob der Tensor der bereits belegten Indizes und der Tensor der Indizes des neuen Ellipsoids gleiche Einträge aufweisen. Falls ja, wird das Ellipsoid verworfen, falls nein werden die Indizes des Ellipsoids den belegten Indizes angehängt. Nachteil dieses Vorgehens ist, dass die Rechenzeit mit Hinzufügen eines jeden Korns weiter steigt, da der Tensor der belegten Indizes stetig größer wird. Sind genügend Ellipsoiden platziert, stoppt der Algorithmus und das fertige "Gefüge" liegt als *torch.sparse.FloatTensor* vor. Die Speicherung der Daten als *Dünnbesetzte* Matrix ist besonders im Hinblick auf den Speicherplatz effizient, da auch bei einer Befüllung des Grids mit mehreren tausend Körnern immer noch nur wenige Prozente eines 1000x1000x1000 Grids besetzt sind.

Ist ein Gefüge erzeugt, wird im Anschluss ein "Quasi-EBSD" durchgeführt. Zunächst wird der *sparse Tensor* in einen normalen Tensor umgewandelt, welcher aus Nullen und Einsen besteht. An jeder Stelle, die von einer Ellipse bedeckt ist eine Eins. Aus dem dreidimensionalen Tensor des Grids können "Bilder" herausgeschnitten werden, indem der zweidimensionaler Tensor an einer beliebigen Stelle der x-, y- oder z-Achse ausgewählt wird. Auf diesen drei Bildern (Die Matrix kann einfach als Bild interpretiert werden) können mittels der Bildverarbeitungsbibliothek *openCV* Ellipsen identifiziert und als .csv-Datei gespeichert werden. Auf diese kann der Algorithmus angewandt werden. Abschließend liegt die *synthetische Mikrostruktur*, mit welcher das Grid gefüllt wurde und die *rekonstruierte Mikrostruktur* des Algorithmus vor, welche nach verschiedenen Parametern verglichen werden können.

Für die Bewertung des Algorithmus ist es wichtig, zuvor ein grundsätzliches Problem bei der Nutzung des EBSD-Verfahrens zu erläutern: Wird eine Ellipse aus einem auf einer Aufnahme zu sehenden Korn berechnet, wird implizit angenommen, die Mitte des Korns getroffen zu haben. Diese Annahme ist jedoch unlogisch, da bei mehreren Körnern, welche nicht alle auf einer Ebene liegen, eine zweidimensionale Analyse unmöglich jedes Korn mittig treffen kann. Bild 16 verdeutlicht dieses Problem, die schematische Kamera symbolisiert den Blick von oben auf die fünf Ellipsoiden.

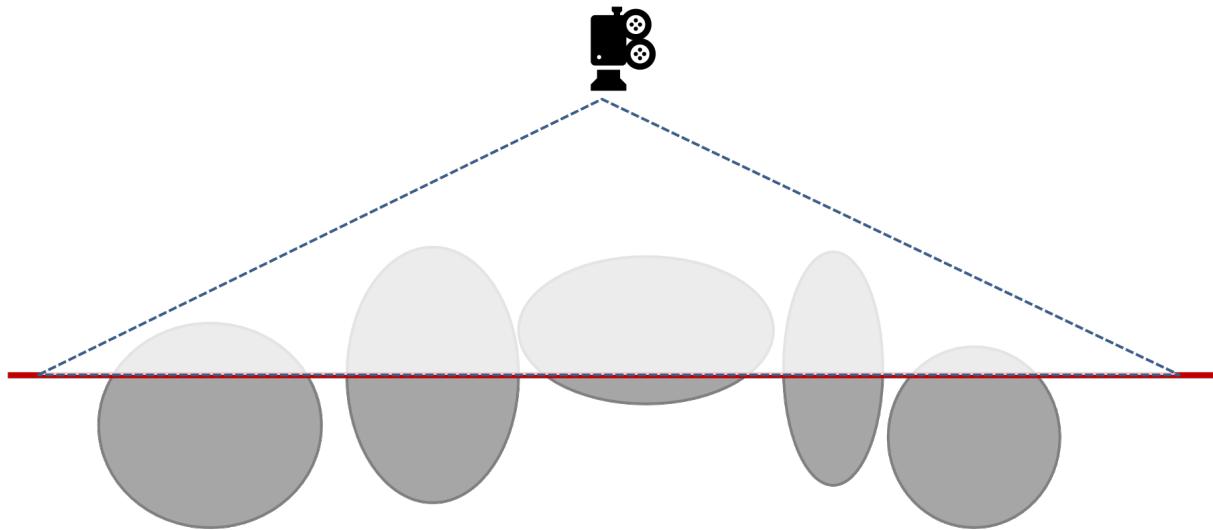


Bild 16: Problem bei der zweidimensionalen Aufnahme dreidimensionaler Körner (Kamera dient als Veranschaulichung)

Wird eine regellose Aufreihung von Kugeln (oder Ellipsoiden) an einer Stelle (rote Linie im Bild) abgeschnitten und von oben aufgenommen (Position der Kamera), sind auf der Aufnahme ausschließlich Ellipsen zu sehen. Diese können allerdings die Realität nur begrenzt abbilden, da z.B. das mittlere Ellipsoid in der Aufnahme von oben viel kleiner als in Wirklichkeit erscheint. Dieses Problem führt zu den folgenden zwei Arbeitshypothesen für die Validierung des Algorithmus:

- Die Korngrößen werden grundsätzlich zu klein geschätzt: Dies liegt nahe, da nur ein exakter Schnitt durch die Mitte die Größe richtig wiedergibt.
- Die Verteilung der Korngrößen wird zu breit geschätzt, d.h. es werden mehr Korngrößen rekonstruiert als in Wahrheit vorliegen. Hierzu führt die Tatsache, dass selbst bei einem Gefüge aus absolut gleichen Kugeln auf einer Aufnahme verschiedene Größen zu sehen sind.

In Kapitel 5 werden die Ergebnisse des Algorithmus speziell unter Berücksichtigung dieser beiden Hypothesen untersucht. Speziell soll geprüft werden, ob der Algorithmus folgende Parameter der synthetischen Mikrostruktur abbilden kann:

- Die Aspect Ratios der einzelnen Halbachsen (a,b,c)

- Die Volumenverteilung der Körner
- Die Verteilung der Halbachsen
- Das mittlere Volumen und zusätzliche Parameter der Verteilung

Weiterhin sollen die Drehungen betrachtet werden. Da der Algorithmus nicht direkt mit den Halbachsen (a, b, c) arbeitet, sondern mit den Achsenkomponenten (x, y, z), müssen diese in die Halbachsen zurückgerechnet werden, bevor das reale mit dem rekonstruierten Gefüge verglichen werden kann. Da die zur Rekonstruktion verwendeten Ansätze zu weiteren Diskussionen der Ergebnisse führen, werden diese Ansätze in Kapitel 5 erläutert.

Kapitel 5 Ergebnisse und Diskussion

Das Kapitel mit der Darstellung der Ergebnisse ist analog zu Kapitel 4 in zwei Teile gegliedert. Zunächst sollen die Ergebnisse des CWGAN-GP diskutiert werden, im zweiten Teil soll der Fokus auf der Anwendung und Validierung des in Abschnitt 4.2.1 vorgestellten Algorithmus zur Rekonstruktion dreidimensionaler Mikrostruktur (Abbildung 3) liegen. Informationen über die mögliche Verknüpfung beider Ansätze befinden sich im Ausblick (Vgl. Kapitel 6).

Eine trennscharfe Abgrenzung zwischen Ergebnissen und Diskussion ist im Rahmen dieser Arbeit oftmals nicht möglich. Zum einen sind die in Kapitel 4 entwickelten Algorithmen bereits ein großer Teil der Ergebnisse dieser Arbeit. Zum anderen basieren viele weitere in diesem Kapitel durchgeföhrten Analysen auf vorhergegangenen Ergebnissen und der Diskussion dieser. Somit wurden Ergebnisse und Diskussion in diesem Kapitel zusammengefasst.

5.1 Generierung zweidimensionaler Mikrostruktur

In diesem Kapitel befinden sich die Ergebnisse des CWGAN-GP zur Generierung synthetischer zweidimensionaler Mikrostruktur. Kapitel 5.1.1 behandelt verschiedene Parameter zur Stabilisierung des Trainings, Kapitel 5.1.2 ist eine Parameterstudie mit verschiedenen Aktivierungsfunktionen. In Kapitel 5.1.3 sollen die Hyperparameter einzelner Optimierungsalgorithmen analysiert werden.

Zu beachten ist, dass jede in dieser Arbeit betrachtete Parameterkonfigurationen Ergebnisse für jedes Material liefert. Aus Übersichtlichkeitsgründen werden nicht immer Darstellungen für jedes Material gezeigt.

5.1.1 Stabilisierung des Trainings

Vorteilhaft für die Suche nach stabilen Parameter für das CWGAN-GP, mittels welcher durch das CWGAN-GP gleichzeitig eine sehr gute synthetische Mikrostruktur für jedes Material generiert werden kann, ist, dass bereits im Rahmen vorheriger Arbeiten Parameterstudien durchgeführt wurden. Daher wurde als Startkonfiguration eine bereits bekannte Konfiguration gewählt. Dies entspricht einer Netzbreite von jeweils

256, einer Tiefe von jeweils 1, Batchgröße und Zufallsvektorgröße von 512 und einem dropout von 0,5 (50%). Als Aktivierungsfunktion wurde die *leaky_relu* gewählt, diese ist eine Abwandlung der Gleichung 6, bei der der Ausgabewert des Neurons bei negativem Input nicht 0, sondern $0.2 * x$ ist. Im Vergleich zur vorhandenen Version kommt beim CWGAN-GP der Strafparameter λ neu dazu. Im Gegensatz zum im originalen Paper vorgeschlagenen Wert von 10 wurde dieser Parameter auf 0,1 gesetzt, da Versuche mit $\lambda = 10$ gezeigt haben, dass in diesem Fall der Strafterm die eigentliche Zielfunktion (Gleichung 16) dominiert und die Ergebnisse nicht mehr verwertbar sind. Insgesamt wurden 150.000 Iterationen des Generators trainiert (Für die Beschreibung der Hyperparameter siehe Kapitel 4.1.1).

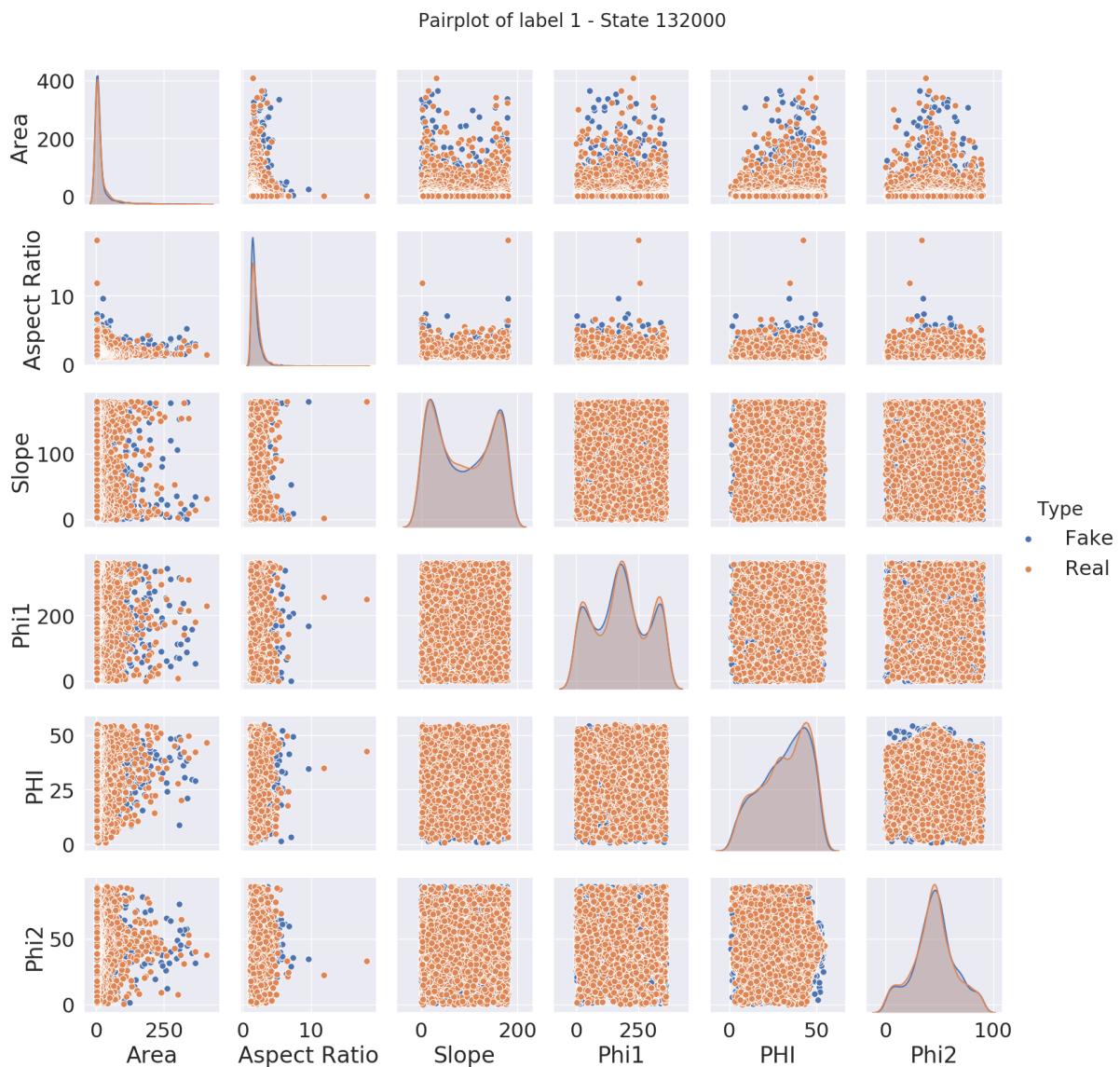


Bild 17: Pairplot echter und synthetischer Daten für den DP800 (tk) bei der optimalen Iteration 132000. “Label 1” steht für das zweite Material.

Für den DP800 (von thyssenkrupp) sind die Ergebnisse des Trainings in Bild 17 zu sehen. Die synthetischen Daten (Fake) sind in blau dargestellt, die Realdaten (Real) in Orange. Die Optimale Iteration 132000 wurde über die minimale Sinkhorn-Distanz

zwischen echten und synthetischen Daten bestimmt. Der Pairplot zeigt eine sehr gute Übereinstimmung der synthetischen und echten Daten. Da die optische Auswertung allerdings nicht die Interdependenzen untereinander erfassen kann, sind die Werte des SinkLoss für alle sieben Materialien und der Mittelwert in Tabelle 2 zu sehen. Diese Materialien können in den Überschriften der Pairplots den Labeln 0 (DP800 tk) bis 6 (HMS 35%) zugewiesen werden.

Tabelle 2: SinkLoss für alle Materialien und jeweils optimale Iteration bei einer Tiefe von 1 und einer Breite von 256

Material	Iteration (in 1000)	Loss
DP800 (AM)	59	0,026975
DP800 (tk)	132	0,032653
DP1000	145	0,045525
Einsatzstahl	80	0,026384
HMS	109	0,019121
Interstitiell Free	146	0,021569
HMS 35% gedehnt	146	0,016541
Mittel	–	0,026967

Die Loss- bzw. Distanzwerte sind nicht trivial zu interpretieren, da es keine prozentualen Abweichungen oder anderweitig genormte Werte sind, sondern absolute Distanzwerter im mehrdimensionalen Raum. Eine Distanz von 0 entspricht der exakten Gleichheit zweier Verteilungen bzw. Punktwolken. Zur weiteren Einordnung dieser Werte kann die Distanz zwischen bekannten Verteilungen dienen. Beispielsweise ist die Distanz zwischen zwei normalverteilten Zufallsvariablen mit jeweils 39300 Werten (Alle Materialien zusammen haben in etwa 39300 Werte) in 6 Dimensionen bei einem Mittelwert von 1 und einer Standardabweichung von 0,5 ca. 0,042, bei einer Standardabweichung von 0,4 ca. 0,026 und bei einer Standardabweichung von 0,3 etwa 0,014. Die Größenunterschiede in Tabelle 2 befinden sich in etwa in der gleichen Größenordnung wie die verschiedenen Normalverteilungen. Z.b. entspricht der Verteilungsunterschied beim HMS 35% etwa dem zweier Normalverteilungen mit $\sigma = 0,3$ und der Verteilungsunterschied des DP1000 etwa dem einer Normalverteilung mit $\sigma = 0,5$. Dies bedeutet, dass jeweils beide Verteilungen als quasi identisch angesehen werden können. Im Gegensatz zu diesen Werten ist die Differenz zwischen zwei Normalverteilungen mit $\sigma = 0,5$ und Mittelwerten von 1 und 2 etwa 3,0, also deutlich höher.

Die größte Distanz (also das schlechteste Ergebnis) ist beim DP1000 zu finden, die geringste Distanz beim um 35% gedehnten Hochmanganstahl. Ein Vergleich der beiden Pairplots zeigt allerdings keine nennenswerten optischen Abweichungen (Bilder 18 und 19).

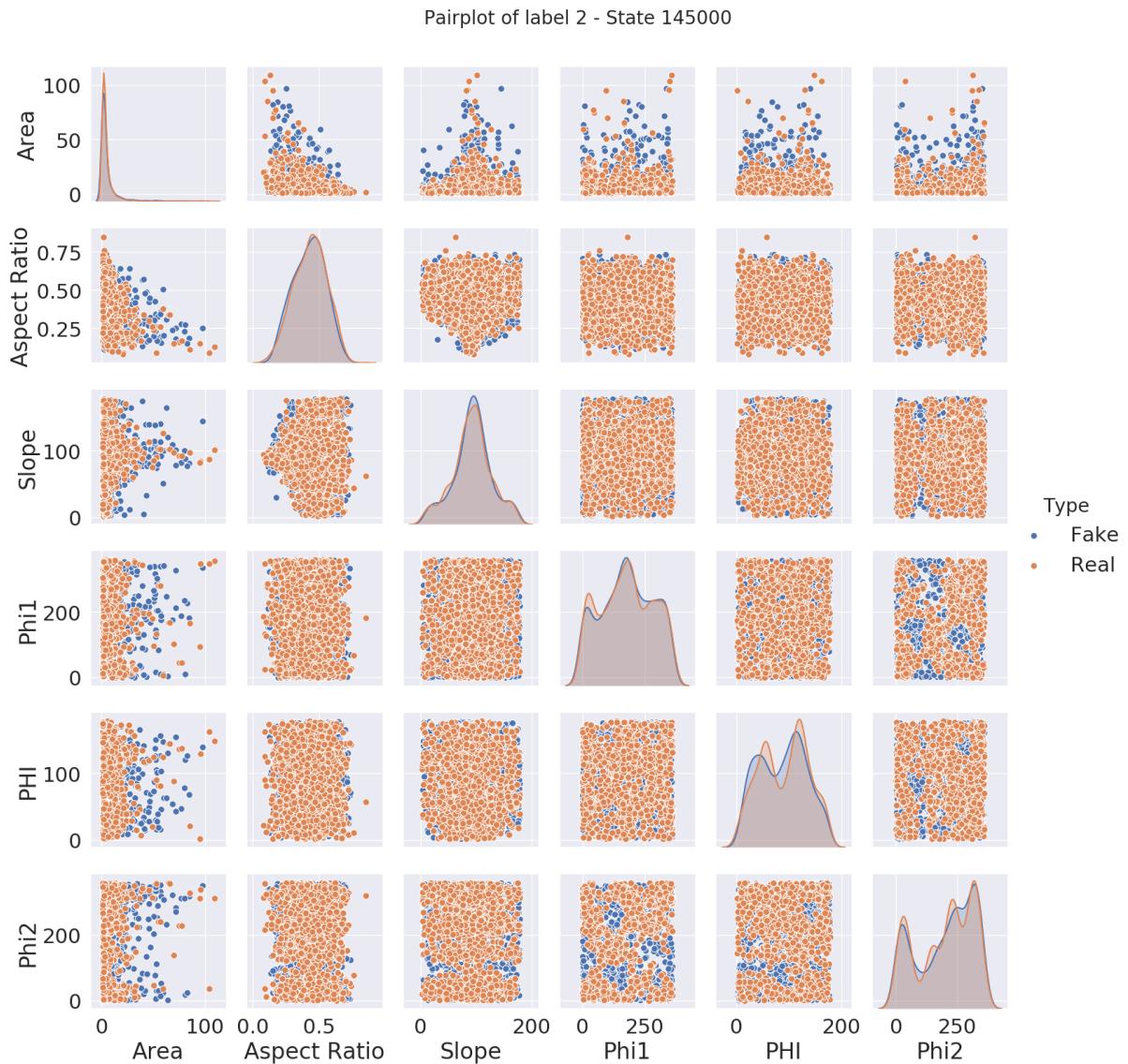


Bild 18: Pairplot echter und synthetischer Daten für den DP1000 bei der optimalen Iteration 145000

Ein Grund für den höheren Loss könnte die Tatsache sein, dass das GAN die Interdependenzen zwischen den Eulerwinkeln Phi2 und Phi1 beim DP1000 nicht optimal erfassen kann. In Bild 18 sind diese in Form der "Lücken" in den Plots Phi1 vs Phi2 und umgekehrt zu sehen. Da das GAN diese Lücken nicht korrekt darstellen kann (die Lücken sind mit blauen Punkten gefüllt), kommt ein höherer Loss zustande. Beim HMS sind diese Lücken nicht vorhanden, daher hat das GAN weniger Schwierigkeiten und der Loss ist deutlich geringer. Die Lücken können lose auf eine Textur des Materials hinweisen, da sie bedeuten, dass verschiedene Kombinationen von Eulerwinkeln (d.h. Orientierungen) nicht vorkommen.

Insgesamt sind die Ergebnisse dieser Konfiguration bereits gut, besonders die Ergebnisse beim DP1000 sind allerdings Verbesserungswürdig. Eine naheliegende Verbesserung ist, beide Netze des GAN's in die Tiefe zu erweitern und dafür die Breite zu

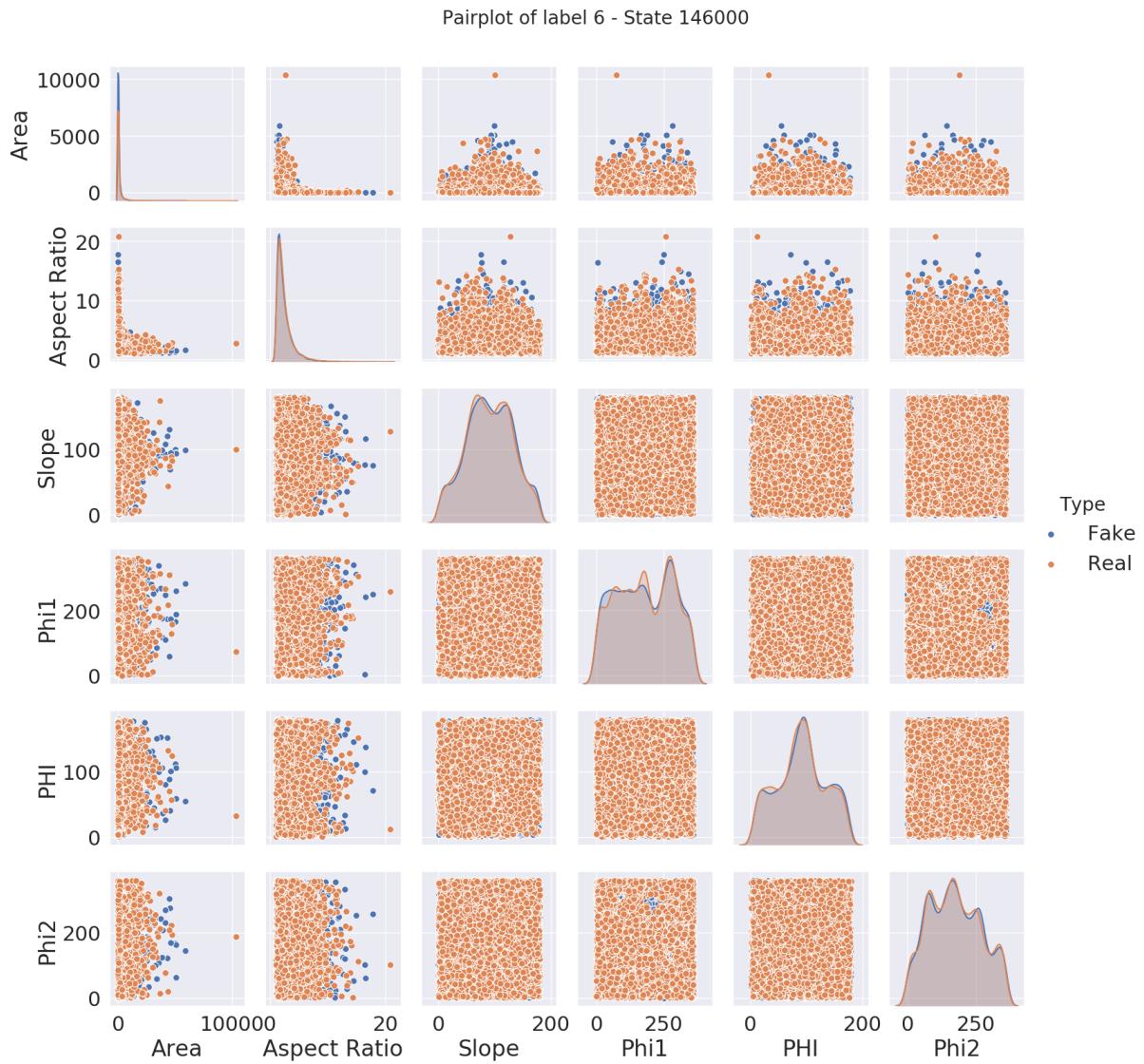


Bild 19: Pairplot echter und synthetischer Daten für den HMS bei der optimalen Iteration 146000

reduzieren. Somit wird eine Tiefe von 2 Hidden Layer und im Gegenzug eine Breite von 128 gewählt. Alle anderen Parameter bleiben gleich. Die Losses sind in Tabelle 3 dargestellt.

Diese Änderung bietet allerdings keine Verbesserung, sondern eine Verschlechterung des mittleren Losses um ca. 0.002. Eine Betrachtung des im Trainingsprozess berechneten Wasserstein-Loss des Discriminators zeigt, dass dieser im Laufe des Trainings stark schwankt (Vgl. Bild 20). Zu beachten ist, dass dieser Wert auch sehr stark von der Batchsize abhängt. Eine bereits recht große Batchsize von 512 reduziert das "Zittern" des Wasserstein-Loss im Vergleich zu einer Batchsize von 32 (Vgl. Bild 53 im Anhang) bereits deutlich. Ein solches Zittern des Loss im Trainingsprozess deutet allerdings auf eine gewisse Instabilität des Trainingsprozesses hin. Um diese Instabilität zu beseitigen, wird der Dropout-Parameter von 50% auf 0% reduziert. Die Sinkhorn-Distanzen

Tabelle 3: SinkLoss für alle Materialien und jeweils optimale Iteration bei je einer Breite von 128 und einer Tiefe von 2

Material	Iteration (in 1000)	Loss
DP800 (AM)	93	0,027568
DP800 (tk)	81	0,034526
DP1000	126	0,053856
Einsatzstahl	126	0,026286
HMS	101	0,020809
Interstitiell Free	129	0,022173
HMS 35% gedehnt	142	0,017481
Mittel	–	0,028957

sind in Tabelle 4 und der Verlauf des Wasserstein-Loss des Discriminators in Bild 21 zu sehen.

Tabelle 4: SinkLoss für alle Materialien und jeweils optimale Iteration bei 0% Dropout und Tiefe gleich 2, Breite gleich 128

Material	Iteration (in 1000)	Loss
DP800 (AM)	123	0,026747
DP800 (tk)	116	0,032991
DP1000	149	0,045412
Einsatzstahl	107	0,024613
HMS	140	0,019985
Interstitiell Free	150	0,020941
HMS 35% gedehnt	135	0,015498
Mittel	–	0,026598

Der Vergleich der Bilder 21 und 20 zeigt deutlich, wie stark der Einfluss des auf 0% reduzierten Dropout-Wertes auf die Oszillationen der Kurve des Wasserstein-Loss ist. Der eigentliche Wertebereich bleibt jedoch gleich, abfallend von etwa 0.055 bei 100 Iterationen in den Bereich knapp über 0. Weitere Analysen haben gezeigt, dass die Kurve konstant mit abnehmendem Dropout glatter wird, eine Kurve mit 30% Dropout liegt also zwischen den beiden Bildern. Ein Grund für die wesentlich glattere Kurve kann sein, dass der Dropout den Discriminator in jedem Trainingsschritt zu stark verändert, sodass das Feedback an der Generator stark durch die jeweils aktivierten bzw. deaktvierten Neuronen beeinflusst ist. Auch die Tabelle 4 zeigt eine deutliche Verbesserung der mittleren Sinkhorn-Distanz im Vergleich zu Tabelle 3 (ca. 0,0025) und eine leichte Verbesserung zu Tabelle 2 (ca. 0,0004). Da zusätzlich keine Hinweise auf Overfitting durch den auf 0% reduzierten Dropout vorliegen, ist diese Konfiguration anderen Konfigurationen vorzuziehen und dient als Basis für die im folgenden vorgestellte Analyse verschiedene Aktivierungsfunktionen in beiden Netzen. Theoretisch könnten bei einem auf 0% reduzierten Dropout auch die Netzbreite reduziert werden (z.b. auf

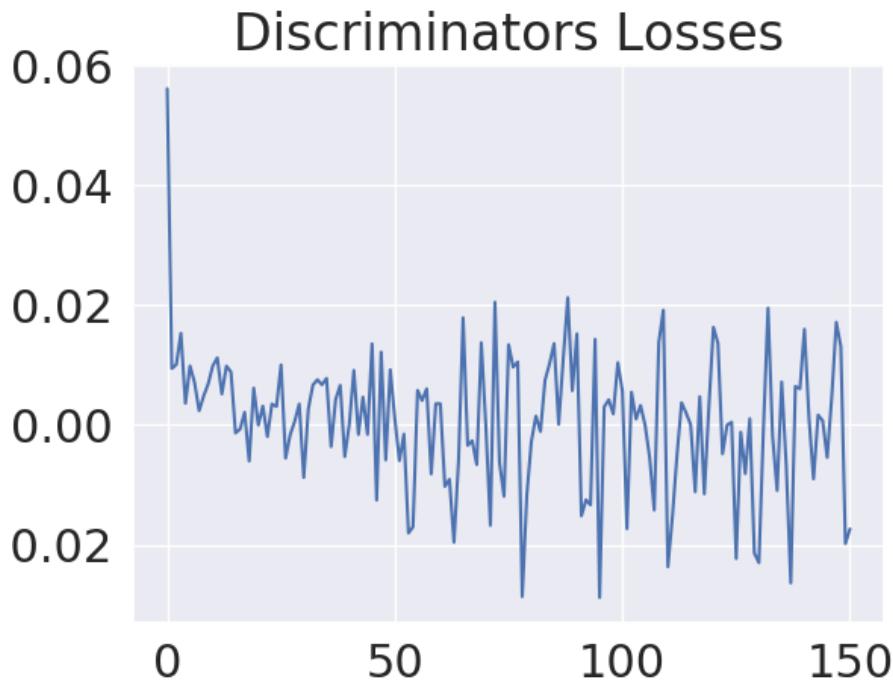


Bild 20: Wasserstein-Loss bei je einer Breite von 128 und einer Tiefe von 2

64 oder 32), da dies jedoch keinen Vorteil in Form einer kürzeren Rechenzeit bringt, wurde davon abgesehen.

5.1.2 Vergleich verschiedene Aktivierungsfunktionen

Das CWGAN-GP hat eine Vielzahl von Hyperparametern (Vgl. Kapitel 4.1.1). Da im Rahmen dieser Arbeit nicht alle verschiedenen Kombinationen bzw. Parameter auf ihre Auswirkung untersucht werden können, wurde der Fokus bei der Untersuchung auf die Analyse verschiedener Aktivierungsfunktionen gelegt. Die Untersuchung verschiedener Konfigurationen von Tiefe und Breite ist zwar ebenso naheliegend, allerdings wurden hier schon Untersuchungen im Rahmen einer vorherigen Masterarbeit und Veröffentlichung durchgeführt, welche wie bereits erwähnt übertragbar sein sollten [49].

Die theoretischen Hintergründe zu Aktivierungsfunktionen sind in Kapitel 3.2.1 zu finden. Für die Analysen in dieser Arbeit wurden vier verschiedene Aktivierungsfunktionen ausgewählt, welche in allen möglichen Kombinationen zwischen Generator und Discriminator getestet wurden. Diese drei Aktivierungsfunktionen sind:

leaky_relu:

$$\delta = \begin{cases} x & x \geq 0 \\ 0.2 * x & x < 0 \end{cases}$$

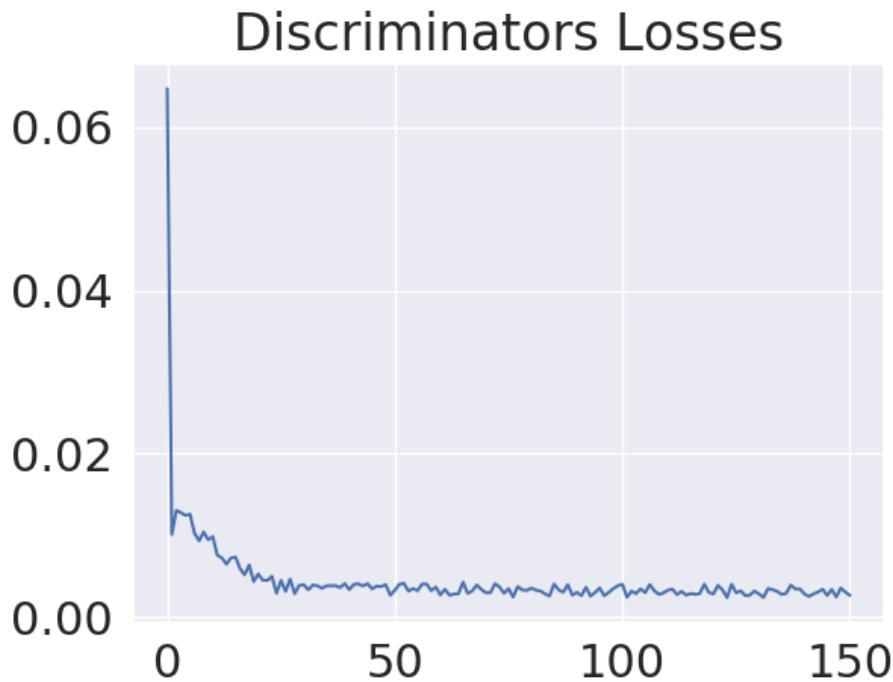


Bild 21: Wasserstein-Loss bei je einer Breite von 128 und einer Tiefe von 2 bei Dropout = 0%

swish:

$$\delta = x * \text{sigmoid}(x)$$

tanh:

$$\delta = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Dazu kommt die ReLU (Gleichung 6). Die *leaky_relu* ist eine Abwandlung der ReLU, welche eine geringere Steigung im negativen Bereich aufweist. Die *swish* weist ein ähnliches Verhalten wie die ReLU-Funktion auf, sie ist nach oben hin unbeschränkt. Allerdings ist der Übergang vom negativen zum positiven Bereich stetiger. Der *Tangens Hyperbolicus* skaliert einen Input x zwischen -1 und 1. Diese Funktion wird häufig dazu auch genutzt, Werte auf das Intervall -1 bis 1 zu skalieren.

Für diese Analysen wurden die Parameter neben den jeweiligen Aktivierungsfunktionen konstant gehalten. Die Konfiguration entspricht der bereits verwendeten Konfiguration in Kapitel 5.1.1, die Gesamtheit aller Hyperparameter ist in Tabelle 5 aufgelistet (Für eine genaue Erklärung der Parameter siehe Kapitel 4.1.1):

Insgesamt wurden 16 Simulationen durchgeführt, das entspricht 4×4 Kombinationen. Diese wurden automatisch mittels *for-Schleifen* unter Nutzung von Ressourcen

Tabelle 5: Hyperparameterkonfiguration zum Testen verschiedener Aktivierungsfunktionen

Hyperparameter	Generator	Discriminator
depth	2	2
width	128	128
z_dim	512	–
learning_rate	0,00005	0,00005
Batch_Size	512	512
Dropout	–	0
Iterations	150.000	–
d_loop	–	5
Optimizer	RMSProp	RMSProp

des RWTH-High Performance Computing durchgeführt. Ein Training von 150.000 Generator-Iterationen benötigt etwa 1,5h bei Nutzung einer Grafikkarte. Dazu kommen wenige Minuten für die Auswertung.

Die Ergebnisse der Simulationen sind in Tabelle 6 zu sehen. Dies ist jeweils der Mittelwert der sieben einzelnen Sinkhorn-Distanzen, die einzelnen Distanzen verhalten sich ähnlich wie in den Tabellen 4, 3 und 2. Das beste und das schlechteste Ergebnis ist jeweils **fett** hervorgehoben.

Tabelle 6: Mittlere Sinkhorn-Distanz für verschiedene Kombinationen von Aktivierungsfunktionen

		G			
		l_relu	swish	tanh	relu
D	G				
	l_relu	0,026598	0,025842	0,024188	0,025550
	swish	0,030393	0,027759	0,028086	0,029211
	tanh	0,026851	0,026416	0,025146	0,025705
	relu	0,026322	0,025500	0,023970	0,025674

Zu sehen ist, dass der Wert in der Tabelle für eine jeweilige Verwendung der leaky_relu exakt dem Mittelwert aus Tabelle 4 entspricht. Da dies zwar exakt dieselben Konfigurationen, aber unabhängig voneinander durchgeführte Berechnungen sind, belegt die Gleichheit der Ergebnisse die sichergestellte absolute Reproduzierbarkeit der Ergebnisse.

Die Ergebnisse zeigen deutlich, dass der *tanh* als Aktivierungsfunktion für den Generator die beste Wahl ist. Die besten drei Ergebnisse kommen bei einer Verwendung des *tanh* beim Generator zustande. Ähnlich verhält es sich mit der *ReLU*-Funktion beim Discriminator. Der Effekt ist hier allerdings nicht ganz so deutlich, die Nutzung des *tanh* beim Generator scheint einen größeren Einfluss zu haben. Weiterhin wird deutlich, dass die *swish*-Funktion (speziell im Discriminator) zu deutlich schlechteren Ergebnissen führt. Alle anderen Ergebnisse liegen im Bereich 0,024 - 0,026, nur mit

swish im Discriminator gibt es Ausreißer bis zu 0,03. Im Gegensatz dazu ist bei der Verwendung der swish-Funktion im Generator keine deutliche Verschlechterung der Ergebnisse zu sehen. Analog zu den obigen Konfigurationen, sind für das beste Ergebnis (G - tanh und D - relu) die Pairplots für den HMS bei 35% Dehnung (niedrigste Distanz von 0,014139) und den DP1000 (höchste Distanz 0,039996) in den Bildern 22 und 23 dargestellt. Besonders beim DP100 konnte eine deutliche Verbesserung im Vergleich zur Tabelle 3 erzielt werden.

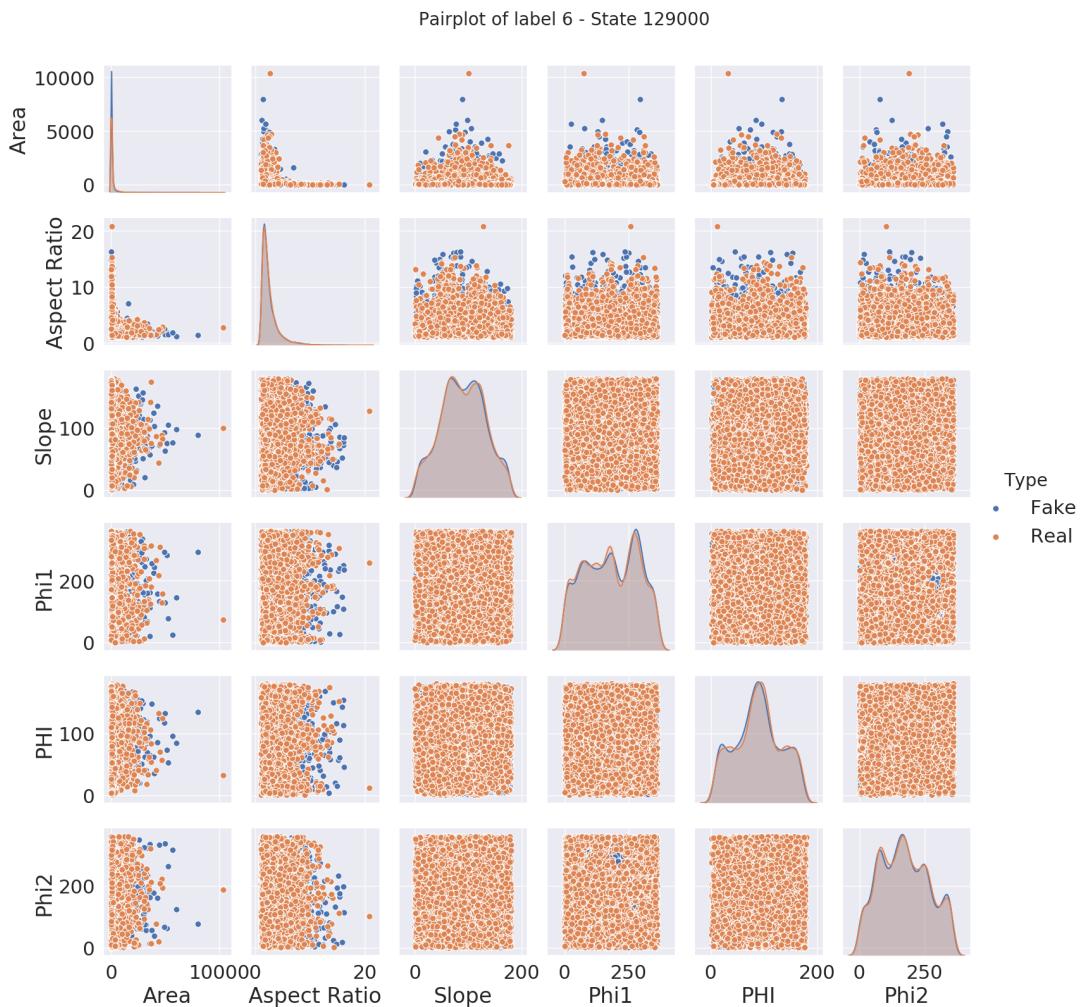


Bild 22: Pairplot echter und synthetischer Daten für den HMS bei der optimalen Iteration 129000 (G - tanh und D - relu)

Die Verbesserungen sind auch beim Vergleich der Bilder 18 und 23 zu sehen. Deutlich wird, dass mit der verbesserten Auswahl der Lossfunktionen auch die "Lücken" im Scatterplot zwischen den Eulerwinkeln erzeugt werden können. Ebenso weisen die Kerndichteschätzungen für Phi2 und PHI eine größere Übereinstimmung auf.

Für die unterschiedlichen Ergebnisse bei Wahl verschiedener Aktivierungsfunktionen gibt es die folgenden Erklärungen: Das der tanh im Generator am besten funktioniert,

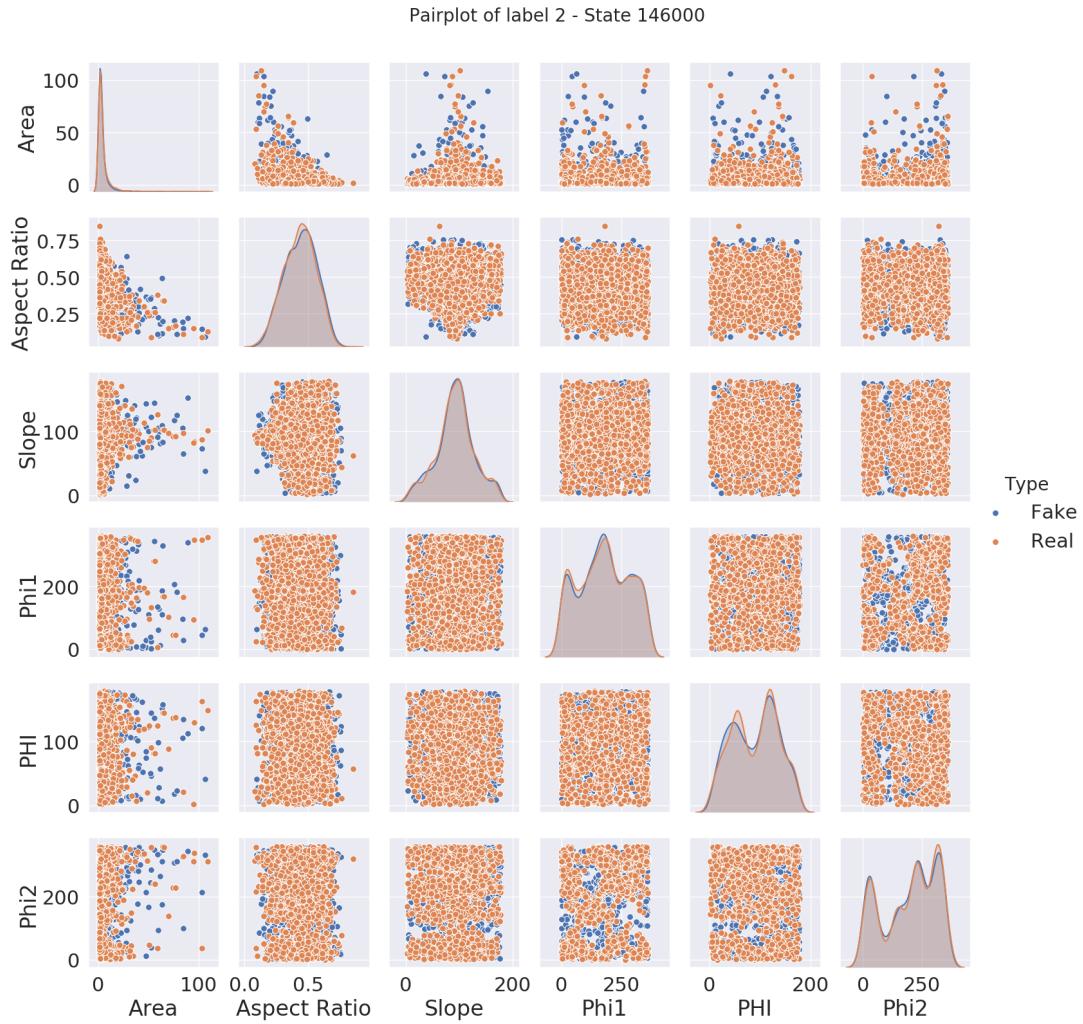


Bild 23: Pairplot echter und synthetischer Daten für den DP1000 bei der optimalen Iteration 146000 (G - tanh und D - relu)

ist durch die Skalierung der Daten begründet. Bevor die Realdaten in den Discriminator (genauer: In den DataLoader) geladen werden, werden alle Datenpunkte auf das Intervall -1 bis 1 skaliert. Da die Realdaten in diesem Intervall liegen, müssen auch die synthetischen Daten aus dem Generator in diesem Intervall liegen. Daten, welche unterschiedlich skaliert sind können unmöglich verglichen werden. Dazu wird ebenfalls ein Tangens Hyperbolicus verwendet, der hinter dem Output-Layer des Generators angebracht wird. Dementsprechend müssen alle Daten, welche auch durch die Aktivierungsfunktionen modifiziert durch den Generator laufen, zum Ende wieder auf das Intervall (-1, 1) skaliert werden. Somit ist naheliegend, die Daten während des Durchlaufs durch den Generator durch die Aktivierungsfunktion direkt nach jedem Hidden-Layer auf dieses Intervall zu skalieren. Dies ist der Grund für den Vorteil des tanh gegenüber anderen Aktivierungsfunktionen.

Weiterhin ist der Unterschied zwischen der ReLU und leaky_ReLU beim Discriminator zu erkennen. Hierfür gibt es keine ebenso naheliegende Erklärung, eine Möglichkeit

ist, dass das insgesamt etwas linearere Verhalten der leaky_ReLU für schlechtere Ergebnisse sorgt. Wie in Kapitel 3.2.1 beschrieben, muss die Aktivierungsfunktion nichtlinear sein, da sonst das Perceptron nur lineare Transformationen durchführen kann.

Die schlechten Resultate der swish-Funktion können nicht ohne weiteres erklärt werden. Diese zeigen besonders in Kombination mit der ReLU-Funktion teilweise abnormales Verhalten. Der Wasserstein-Loss für die Kombination (D-Swish, G-Relu) zeigt starke Ausreißer (Vgl. Bild 24)

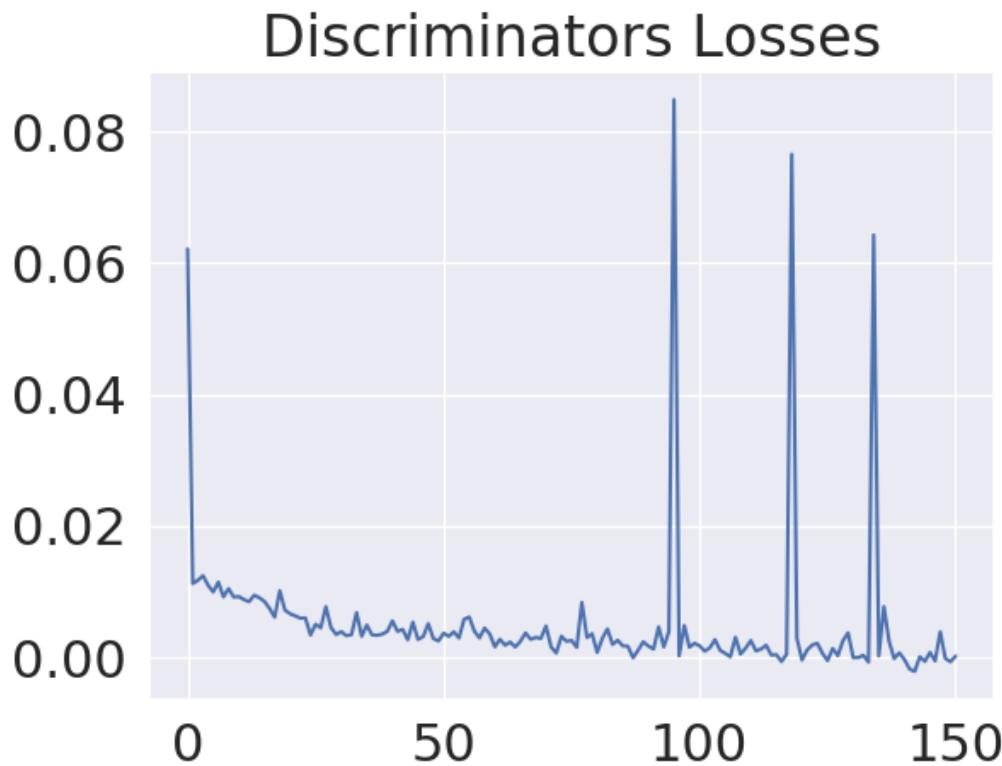


Bild 24: Wasserstein-Loss für die Kombination (D-Swish, G-Relu)

Diese Ausreißer überschreiten auch den "Startwert" von 0.06 bei der 100. Generatoriteration teilweise deutlich. Der erste Ausreißer pflanzt sich auch bis in die Sinkhorn-Distanz fort, deren Verlauf ebenfalls einen heftigen Ausreißer aufweist. Die Kurve ist in Bild 25 zu sehen, beispielhaft wurde die Kurve des Einsatzstahls gewählt.

Besonders bemerkenswert ist dieser Ausreißer, da die Kurve sonst nahezu vollständig schwankungsfrei verläuft. Sowohl das schlechtere Ergebnis, als auch die schwer erklärbaren Ausreißer deuten auf insgesamt fehlerhaftes Verhalten der Netze bei Verwendung der swish-Funktion hin. Da nicht vorausgesagt werden kann, wie sich diese Probleme bei weiteren Materialien oder auch weiteren Eigenschaften/Features auswirkt, sollte die swish-Funktion bei weiteren Analysen außen vorgelassen werden.

Zusammengefasst zeigen die Ergebnisse bei der Nutzung verschiedener Aktivierungsfunktionen erstens mögliche Verbesserungen der Ergebnisse im Vergleich zum Start

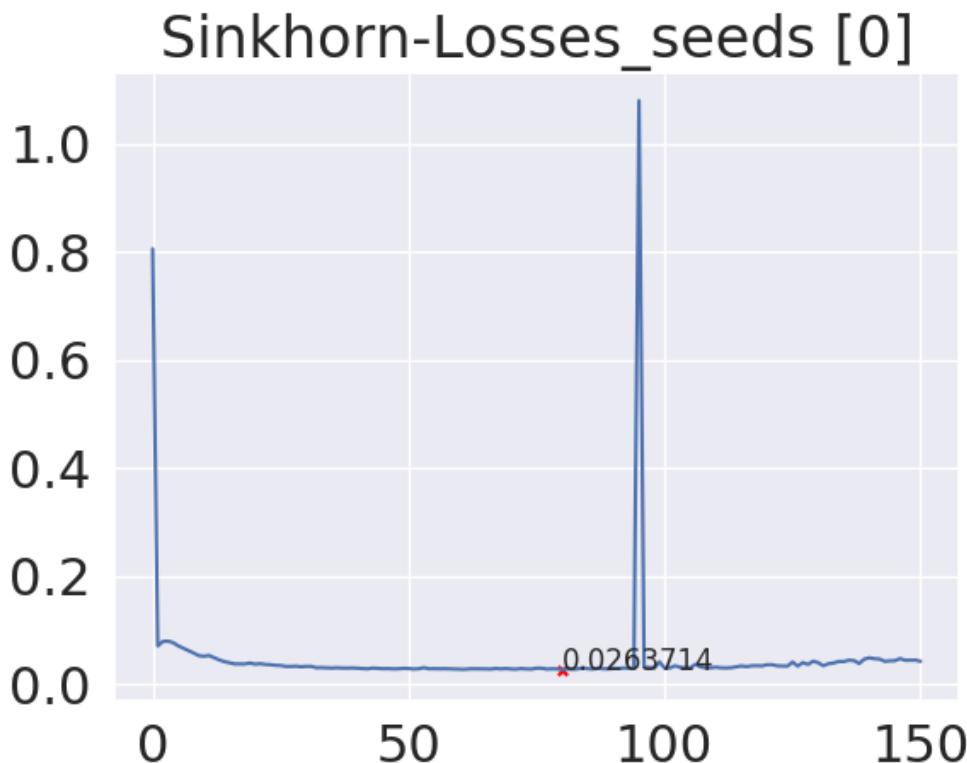


Bild 25: Sinkhorn-Distanz mit Ausreißer für den Einsatzstahl

des Trainings und zweitens wurde aufgezeigt, dass die korrekte Auswahl der Aktivierungsfunktionen die Ergebnisse verbessern. Anzunehmen ist, dass diese Erkenntnisse auch übertragbar sind, falls weitere Materialien und/oder Features hinzugefügt werden. Dies liegt daran, das oben erläuterte Gründe für die Vorteile von tanh und relu gegenüber den anderen Funktionen unabhängig von Features und Anzahl der Klassen sind.

5.1.3 Vergleich verschiedener Optimizer

Optimizer sind im Wesentlichen Modifikationen der Gleichung 9. Ziel dieser Modifikationen ist meist eine Beschleunigung und Stabilisierung des Trainingsprozesses. Die Betrachtung der Optimizer ist in dieser Arbeit aus ähnlichen Gründen wie die Betrachtung der Aktivierungsfunktionen gewählt worden. Sie sollten übertragbar sein, auch wenn Materialien und/oder Features hinzugefügt werden. Interessant ist besonders die Betrachtung des Adam-Optimizers (ein Eigenname, keine Abkürzung) im Vergleich zum bislang verwendeten RMSprop. Grundsätzlich gilt der Adam-Optimizer als Standard, er wird in vielen Referenzarchitekturen verwendet [34]. Dazu gehört auch die Originale Implementierung des WGAN-GP [26]. Im Gegensatz dazu wurde in der ursprünglichen Version des WGAN (ohne Gradient Penalty) der RMSprop vorgeschlagen [5]. In diesem Abschnitt soll untersucht werden, ob der Adam-Optimizer bei Variation seiner beiden Hyperparameter b_1 und b_2 bessere Ergebnisse liefern kann,

als der RMSprop. Hierbei bleiben die Hyperparameter gleich wie in Tabelle 5, als Aktivierungsfunktion wurde in beiden Fällen ein Tangens Hyperbolicus verwendet. Somit ist der Referenzwert der Tabelle 6 zu entnehmen, dieser ist 0.025146.

Der Adam ist ein lernraten-adaptiver Optimizer. Das bedeutet, dass die Lernrate im Laufe des Trainings dynamisch angepasst wird. Bildlich gesprochen bedeutet das, dass der Optimizer die Lernrate erhöht, sofern das Training "in die Richtige" Richtung läuft (In Richtung eines Lokalen Minimums). Dabei kommt ein sogenannter "moving average" zum Einsatz, ein gleitender Mittelwert über die berechneten Gradienten. Es wird ein moving Average für den einzelnen und den quadrierten Gradienten berechnet (g_t ist der Gradient):

$$m_t = \frac{b_1 * m_{t-1} + (1 - b_1) * g_t}{1 - b_1} \quad (30)$$

$$v_t = \frac{b_2 * v_{t-1} + (1 - b_2) * g_t^2}{1 - b_2} \quad (31)$$

Je größer b_1 oder b_2 wird, desto weniger wird der aktuelle Wert des Gradienten berücksichtigt und je stärker der Einfluss des vorherigen Wert des Mittelwerts. Das Prinzip wird als Momentum bezeichnet. Die Modifizierte Gleichung 9 lautet:

$$w_t = w_{t-1} + \eta * \frac{m_t}{\sqrt{v_t} + \epsilon} \quad (32)$$

ϵ ist ein Wert in der Größenordnung $1 * 10^{-7}$ um zu vermeiden, dass durch 0 geteilt wird. Im Gegensatz zum Adam bleibt beim RMSprop die Gleichung 32 weitestgehend gleich, bei diesem Optimizer steht allerdings nicht der gleitende Mittelwert m_t im Zähler, sondern direkt der Gradient g_t . Der RMSprop ist daher im Wesentlichen ein Adam ohne Momentum.

Untersucht wurden verschiedene Kombinationen der Parameter b_1 und b_2 . Die Ergebnisse (mittlere Sinkhorn-Distanz) sind in Tabelle 7 dargestellt. Wieder sind bestes und schlechtestes Ergebnis **fett** hervorgehoben.

Tabelle 7: Mittlere Sinkhorn-Distanz für verschiedene Kombinationen der Hyperparameter des Adam-Optimizer

		b1				
		0,2	0,4	0,6	0,8	0,9
b2	0,5	0,025591	0,025648	0,025631	0,026052	0,027589
	0,8	0,025773	0,025191	0,026031	0,026140	0,027185
	0,9	0,025559	0,024926	0,025034	0,026057	0,026741
	0,99	0,02488	0,024895	0,025499	0,025749	0,026477

Im Vergleich zu der Analyse verschiedener Aktivierungsfunktionen wird hierbei deutlich, dass die Unterschiede nicht so groß sind. Die maximale Differenz zwischen

bestem und schlechtestem Ergebnis ist in diesem Fall ca. 0,0027, bei den Aktivierungsfunktionen ca. 0,0064. Ebenso sind in Tabelle 7 keine so eindeutigen Muster zu sehen, allerdings scheinen größere Werte für b_2 und kleinere Werte für b_1 zu besseren Ergebnissen zu führen. Insgesamt liegen vier Werte der Tabelle unter dem Referenzwert des RMSprop, dies sind die Kombinationen (0,2, 0,99), (0,4, 0,9), (0,4, 0,99) und (0,6, 0,9). Alle diese Werte liegen jedoch noch über dem Wert bei Verwendung von tanh (G) und relu (D), dies lässt die Schlussfolgerung zu, dass Hyperparameter und Optimizer den "Nachteil" der nicht optimal ausgewählten Aktivierungsfunktionen nicht ausgleichen können.

Insgesamt liegen jedoch alle Werte in einem guten Bereich. Exemplarisch ist in den Bildern 26 und 27 das Ergebnis für den DP1000 und HMS35 für die optimale Parameterkombination (0,2, 0,99) dargestellt.

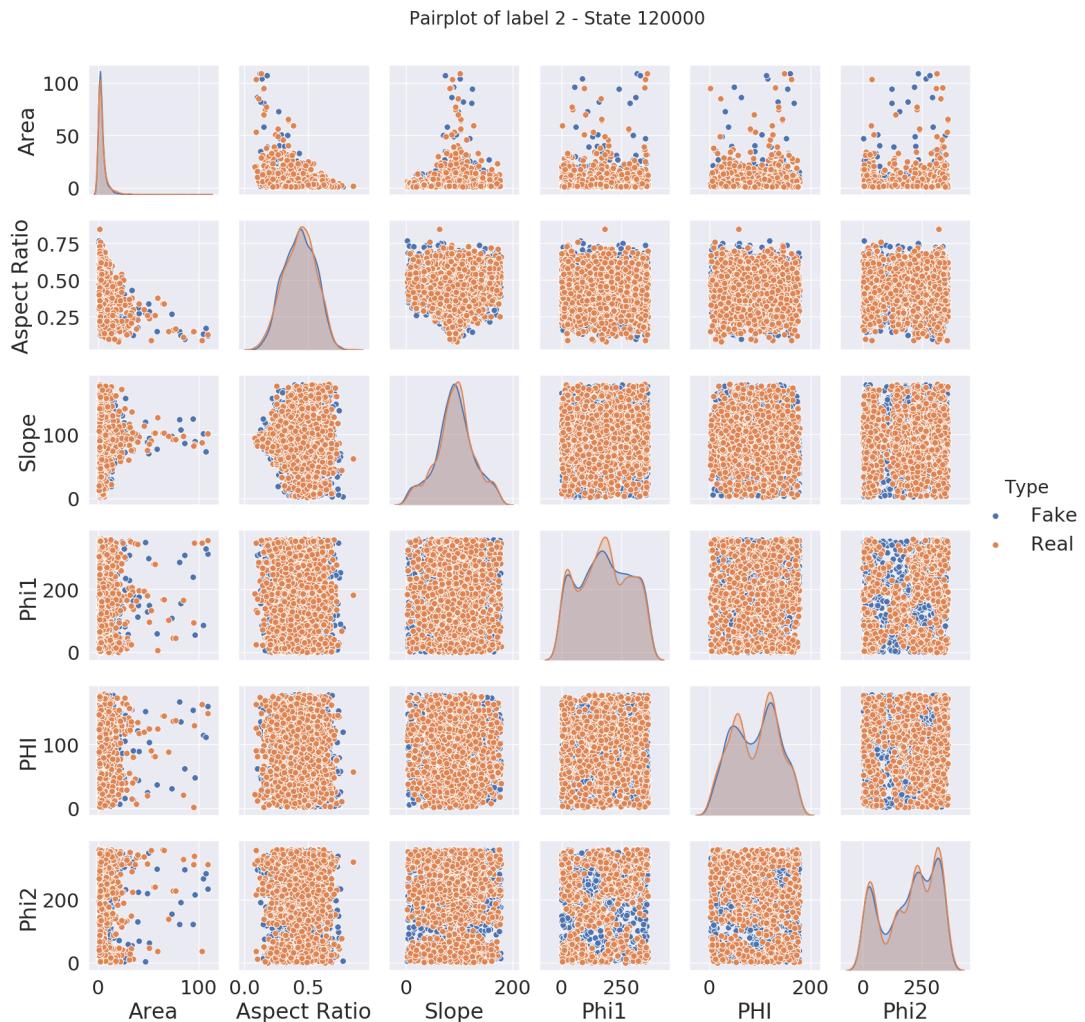


Bild 26: Pairplot echter und synthetischer Daten für den DP1000 bei der optimalen Iteration 120000 (Hyperparameterkonfiguration 0,2, 0,99)

Besonders im Vergleich von Bild 26 mit Bild 23 fällt allerdings auf, dass das Ergebnis mit der Wahl von tanh und relu als Aktivierungsfunktionen insbesondere bei den

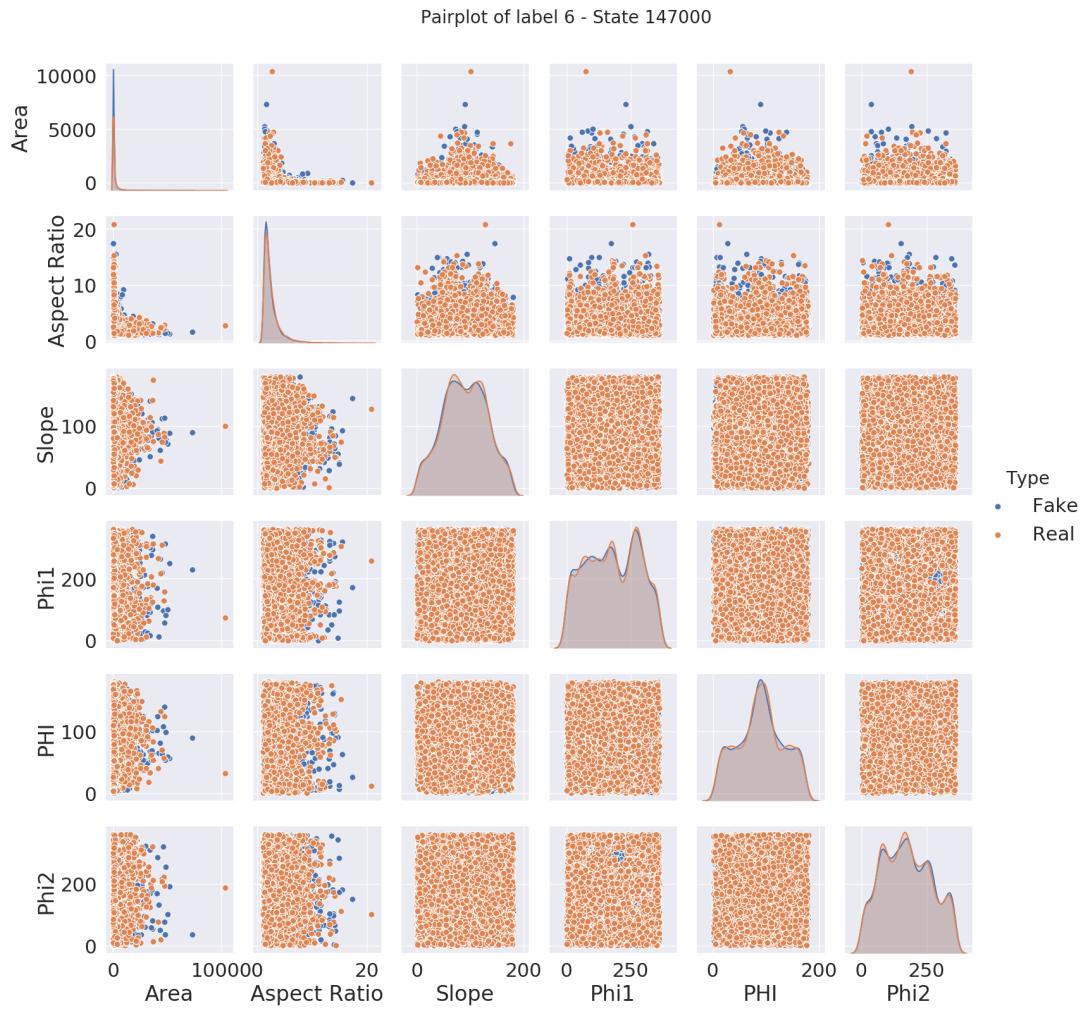


Bild 27: Pairplot echter und synthetischer Daten für den HMS bei der optimalen Iteration 147000 (Hyperparameterkonfiguration 0,2, 0,99)

Eulerwinkel bessere Ergebnisse zeigt. Insgesamt ermöglicht die Variation der Hyperparameter des Adam-Optimizers keine so starken Verbesserungen wie die Variation der Aktivierungsfunktionen.

Generell konnte gezeigt werden, dass die Erweiterung des WGAN hin zum CWGAN-GP sehr gute Ergebnisse bei der Generierung synthetischer Mikrostruktur liefert. Dabei können auch mehrere Materialien gleichzeitig in einem Trainingsprozess dargestellt werden, auch wenn diese Materialien teilweise größere Unterschiede in ihrer Mikrostruktur aufweisen. Trotz insgesamt etwa 40.000 Datenpunkten (alle sieben Materialien) kann der Trainingsprozess in einer vertretbaren Zeitspanne von knapp 100 Minuten mit GPU-Unterstützung durchgeführt werden. Dies ermöglicht die einfache Hinzufügung weiterer Materialien in das System. Weiterhin konnte mit der Nutzung der Sinkhorn-Distanz eine effiziente und praktikable Auswertungsmethode implementiert werden, welche sehr gut mit der optischen Betrachtung der Pairplots korreliert. Als finale Konfiguration wurde somit eine Tiefe von 2, eine Breite von

128, tanh im Generator, relu im Discriminator und RMSprop als jeweiliger Optimizer gewählt.

5.2 Rekonstruktion dreidimensionaler Mikrostruktur

Die Ergebnisse der Rekonstruktion dreidimensionaler Mikrostruktur sind in zwei Abschnitte gegliedert. Zunächst die einfache Anwendung des Algorithmus auf den Dualphasenstahl DP800 zur Überprüfung einer generellen Anwendbarkeit des Ansatzes. Im zweiten Abschnitt wird mittels der in Kapitel 4.2.2 vorgestellten synthetisch generierten Mikrostruktur validiert, dass über den Algorithmus dreidimensionale Mikrostruktur aus zweidimensionalen Aufnahmen hergeleitet werden kann.

5.2.1 Anwendung des Algorithmus

Vor einer vertieften Validierung mittels der synthetischen Mikrostruktur muss geprüft werden, ob der Algorithmus *grundsätzlich* funktioniert, d.h ob es möglich ist, die Teilachsen zu dreidimensionalen Körnern zu verknüpfen. Sollte dies nicht der Fall sein und der Algorithmus versagt, muss der Ansatz überdacht werden. Weiterhin sollen verschiedene *Hyperparameter* (siehe Abbildung 3) überprüft und bewertet werden.

Als Grundlage dienen EBSD-Aufnahmen eines DP800, diese sind in Bild 28 zu sehen. Zu beachten ist, dass hier nur drei bespielhafte Aufnahmen zu sehen sind, da für alle drei Richtungen mehrere Aufnahmen vorliegen. Nach dem in Kapitel 4.2.1 vorgestellten Processing der Daten liegen tabulare Datensätze vor, die als Input dienen. Für diese Überprüfung wurden folgende Zuweisungen gewählt: Kornachse a in TD, Kornachse b in BN und Kornachse c in RD. Daher wird mittels der Aufnahme TDxBN (mit a in x und b in y) die c-Achse durch die Aufnahmen RDxTD und RDxBN hergeleitet. Diese Auswahl wurde aufgrund der begrenzten Anzahl von 750 Körnern in TDxBN getroffen. In den beiden anderen Aufnahmen liegen über 2500 Körnern vor, somit ist die Chance höher, für einen gegebenen a oder b Wert aus TDxBN passende Werte in RDxTD/RDxBN zu finden, als umgekehrt.

Insgesamt sollen drei verschiedene Hyperparameter-Konfigurationen betrachtet werden. Für eine genauere Beschreibung der Hyperparameter siehe Algorithmus 3.

1. threshold und range $\leftarrow 15\%$, dazu $drop \leftarrow False$
2. threshold und range $\leftarrow 15\%$, dazu $drop \leftarrow True$
3. threshold und range $\leftarrow 5\%$, dazu $drop \leftarrow True$

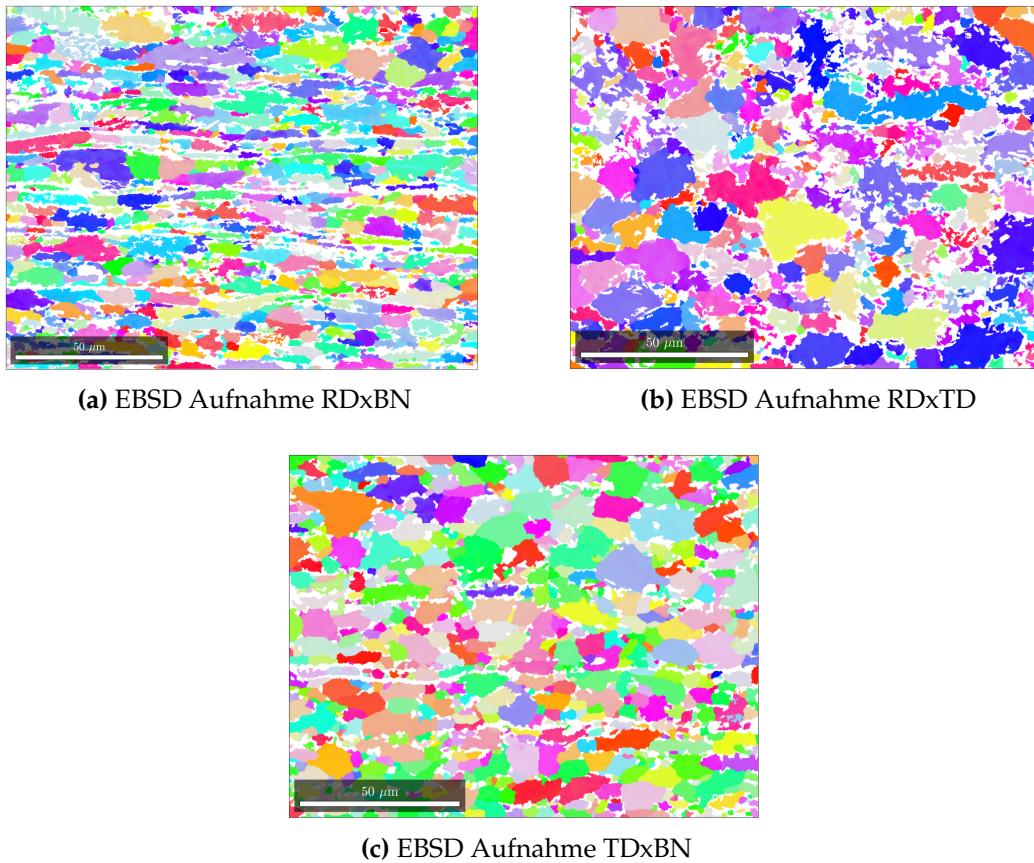


Bild 28: In dieser Arbeit verwendete EBSD-Aufnahmen von drei verschiedenen Seiten (Für alle Seiten wurden mehrere Aufnahmen verwendet, die Abgebildeten sind nur ein einzelnes hiervon)

Mit diesen Konfigurationen wird geprüft, welchen Einfluss zum einen der abzusuchende Radius um einen bestimmten a/b -Wert (*range*) und die erlaubte Abweichung beider bestimmten c -Werte (*threshold*) auf die Performance haben. Dazu soll zudem analysiert werden, welche Einfluss das Rauslöschen von bereits verwendeten c/b und c/a Paaren hat (*drop = True*). Die Anzahl der zu rekonstruierenden Punkte *n_points* wird jedes Mal auf 750 gesetzt, somit wird jedes Mal versucht, alle Punkte aus TDxBN zu verwerten.

Konfiguration eins - threshold / range = 15% & drop=False:

Recht hohe Schwellenwerte und kein Rauslöschen von verwendeten Werten ermöglicht eine maximale Anzahl an verknüpften Punkten bzw. Rekonstruierten Körnern. Dies zeigt sich daran, das 750 von 750 Punkten erzeugt werden konnten.

Zwei Möglichkeiten wurden entwickelt, um den Algorithmus auch ohne die aufwendige Anfertigung von synthetischer Mikrostruktur in erster Näherung zu bewerten. Dazu werden für alle drei Aufnahmen jeweils beide Halbachsen in einem RegPlot der *seaborn*-Library von Python [67] dargestellt. Dieser ist in Bild 29 dargestellt, dabei

sind die neu erzeugten 3D-Körper im oberen Teil des Bildes zu sehen, die jeweiligen Datensätze im unteren Teil. Es handelt sich hierbei um eine Darstellung als Scatterplot mit zusätzlich eingefügter Regressionsgerade.

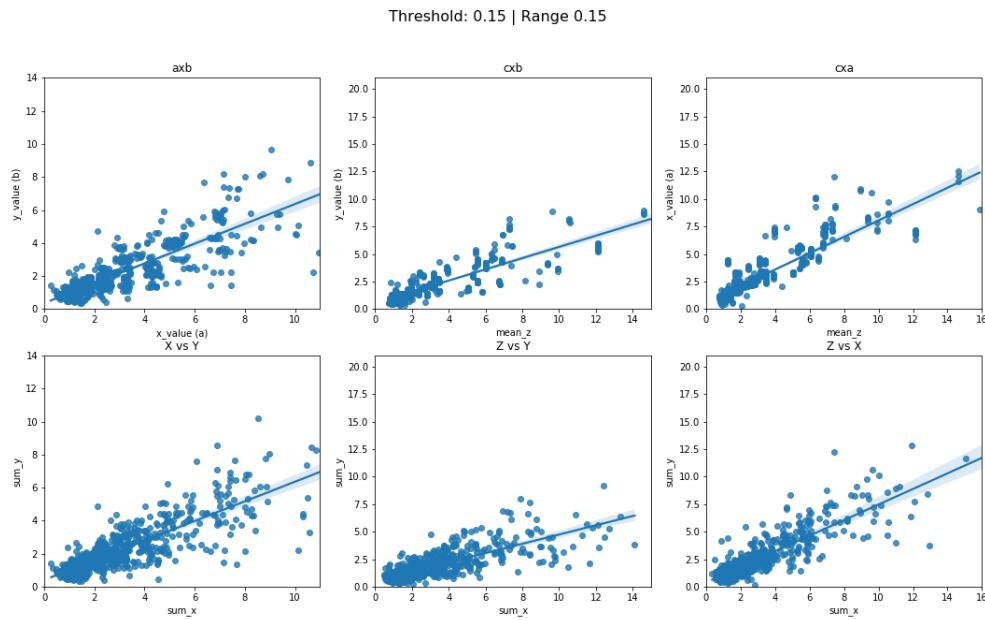


Bild 29: Ein RegPlot für verbundene (oben) und nicht-verbundene Daten (unten) für Konfiguration eins

An diesem Bild ist zu erkennen, dass die obere und untere Reihe sehr ähnlich sind. Dies liefert einen Hinweis darauf, dass durch die Anwendung des Algorithmus keine Verzerrung der Daten stattfindet, da die Regressionsgerade jeweils eine ähnliche Steigung aufweist und somit die Achsenverhältnisse ähnlich sind. Zudem sind durch das fehlende Rauslöschen bereits verwendeter Datenpunkte sehr viele gleiche c bzw. z-Werte zu sehen. Durch einen hohen Prozentwert für den Hyperparameter *range* können sich die Radien, die um einen Punkt betrachtet werden häufiger überschneiden und somit für verschiedene a/b-Werte gleiche c-Werte berechnet werden. Zu beachten ist bei Bild 29, dass für die untere Reihe Plots für eine bessere Vergleichbarkeit zufällig gleich viele Punkte wie in den oberen Plots ausgewählt wurden. Wie bereits beschrieben, liegen für RDxTD und RDxBN deutlich mehr als 750 Punkte vor.

Für eine bessere Analyse können die Steigungen auch berechnet werden (allerdings müssen diese aus den Daten mittels der *scipy*-Library berechnet werden, der RegPlot bietet keine Möglichkeit, den Steigungswert anzeigen zu lassen). Die Steigungen sind in Tabelle 8 zu sehen. In dieser Tabelle sind die jeweiligen Steigungen der oberen (3D) und unteren (2D) Regplots zu sehen. Hier wird ebenfalls deutlich, dass die Steigungen sehr ähnlich sind. Zusammen mit Bild 29 zeigt dies, dass durch die Anwendung des Algorithmus keine "Schiefe" in den Daten erzeugt wird, d.h. dass nicht nur große bzw. kleine Körper im dreidimensionalen erzeugt werden.

Tabelle 8: Zu Bild 29 gehörigen Steigungen

	Rekonstruiert	Real
TDxBN	0,59398307	0,59943982
RDxBN	0,42980443	0,50919326
RDxTD	0,72885119	0,74223649

Die Steigungen der Regplots der jeweiligen Datensätze können auch dazu genutzt werden, theoretisch zu beweisen, dass der Algorithmus funktionieren, die Bilder dementsprechend zusammenhängen: Dazu wird vereinfacht angenommen, dass die Gerade die Punkte in guter Näherung beschreiben (Bei einem R^2 von jeweils ca. 70 - 80% ist diese Annahme zulässig). Damit nun die Daten aufeinanderpassen, müssen die Steigungen *ineinander überführbar sein*. Dazu muss $a(c) == a(b(c))$ sein. Die folgende Rechnung verdeutlicht dies (Werte der Tabelle 8 gerundet und $a/b/c$ statt TD/BN/RD):

$$b = 0,59 * a$$

$$b = 0,43 * c$$

$$a = 0,73 * c$$

damit kann umgeformt werden:

$$0,59 * a = 0,43 * c$$

$$a = \frac{0,43}{0,59} * c$$

0,43/0,59 entspricht 0,73, damit ist bewiesen, dass $a(c)$ exakt gleich dem "Umweg" $a(b(c))$ ist, da $a = 0,73 * c$ (siehe oben). Somit ist gesichert, dass ein Wertepaar (a,b) zu jeweils ähnlichen c-Werten führt. Diese Annahme bildet die Grundlage des Algorithmus und muss gelten, sofern die einzelnen Aufnahmen zu der selben Struktur gehören.

Konfiguration zwei - threshold / range = 15% & drop=True:

Der obige Ansatz liefert bereits gute Ergebnisse. Da jedoch sehr viele gleiche c-Werte erzeugt wurden, wurde im nächsten Schritt überprüft, ob auch bei Rauslöschen bereits verwendeter Werte (d.h. drop = True) ähnlich gute Ergebnisse erzielt werden können. Auch hier ist es möglich, 750 von 750 Körner zu rekonstruieren. Die Zugehörigen Regplots sind in Bild 30 zu sehen.

In dieser Grafik ist im Gegensatz zu Bild 29 zu erkennen, dass c-Werte nicht mehrfach

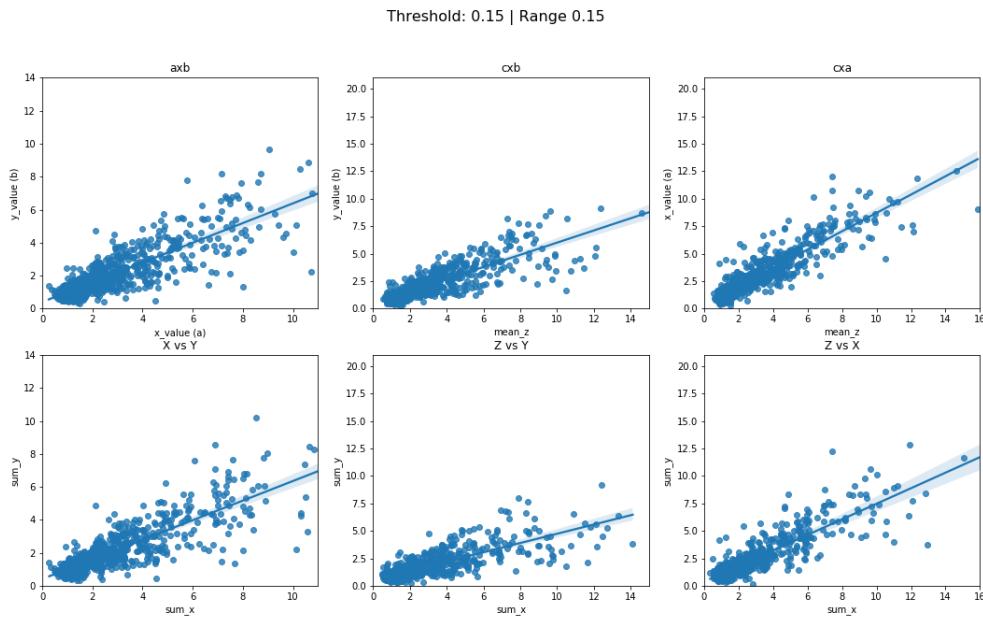


Bild 30: Ein RegPlot für verbundene (oben) und nicht-verbundene Daten (unten) für Konfiguration zwei

vergegeben werden, da sie nach der ersten Verwendung aus den jeweiligen Datensätzen rausgelöscht werden. Auch ohne Mehrfachzuweisung wird deutlich, dass die Daten keine stärkere Verzerrung aufweisen. Die zugehörigen Steigungen sind in Tabelle 9 zu sehen. Im Vergleich zu Tabelle 8 wird deutlich, dass die Steigungen in diesem

Tabelle 9: Zu Bild 30 gehörigen Steigungen

	Rekonstruiert	Real
TDxBN	0,59398307	0,59777309
RDxBN	0,42980443	0,55294198
RDxDT	0,72885119	0,83300065

Fall stärker abweichen. Dies kann durch einen ähnlichen Effekt wie die vielen gleichen Werten in Bild 29 erzeugt werden: Die Radien überschneiden sich durch den hohen Wert für range, dadurch können stärkere Verzerrungen auftreten, welche deutlicher werden, da (unter Umständen sehr gute) Werte für c nicht doppelt verwendet werden können, da sie rausgelöscht wurden.

Konfiguration drei - threshold / range = 15% & drop=True:

Bei einem ausreichend hohen Wert für range ist bei grundsätzlich zueinanderpassenden Datensätzen naheliegend, dass eine hohe Anzahl Körner rekonstruiert werden kann. Daher wurde weiterhin der Algorithmus mit einem rangewert von 5% und eingeschaltetem drop auf die Daten angewandt. Hierbei konnten 745 von 750 Körnern

rekonstruiert werden. Auch bei dieser Konfiguration zeigt der Vergleich der Regplots eine insgesamt gute Übereinstimmung der rekonstruierten Daten und der Basisdaten (Vgl. Bild 31)

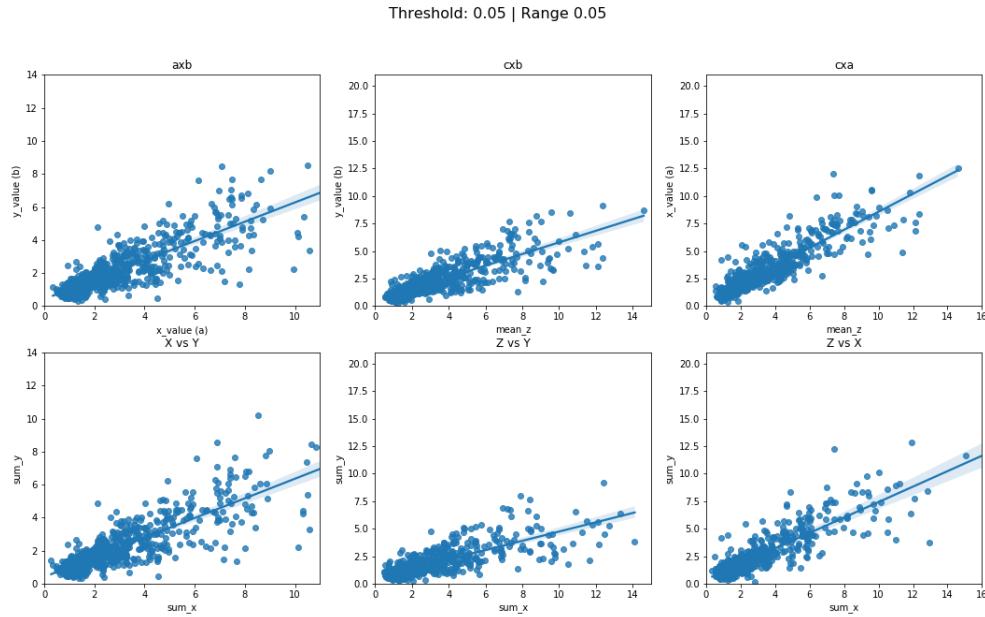


Bild 31: Ein RegPlot für verbundene (oben) und nicht-verbundene Daten (unten) für Konfiguration drei

Auch hier wurden wieder die Steigungen berechnet. Die Steigungen sind insgesamt näher beieinander als bei 15% und Rauslöschen der Punkte, allerdings schlechter als bei 15% ohne Rauslöschen. Diese Konfiguration zeigt jedoch, dass bereits ein sehr

Tabelle 10: Zu Bild 31 gehörigen Steigungen

	Rekonstruiert	Real
TDxBN	0,59398307	0,58449685
RDxBN	0,42980443	0,52663615
RDxTD	0,72885119	0,81741736

kleiner Wert für den Range-Parameter ausreicht, um nahezu alle Körner zu rekonstruieren.

Es ist naheliegend, dass die Parameter auch nach der Anzahl der Körner pro EBSD Aufnahme gewählt werden sollten. Je weniger Körner in den Daten vorliegen, desto größer sollte der range-Parameter gewählt werden. Weiterhin sollte ein Abschalten der drop-Flag die Rekonstruktion erleichtern. Dies ist auch bei der Konfiguration 5/5, drop=False zu sehen, hier können 747 statt 745 von 750 Körnern rekonstruiert werden (Der Vergleich der RegPlots ist im Anhang zu sehen, vgl. Bild 52). Dieser Unterschied ist jedoch marginal, daher sollten zur weiteren Bewertung des Drop-Parameters weitere Analysen durchgeführt werden.

5.2.2 Validierung des Algorithmus

Im vorhergehende Abschnitt wurde bereits gezeigt, dass der Algorithmus grundsätzlich in der Lage ist, eine Dreidimensionale Mikrostruktur aus einzelnen, zweidimensionalen EBSD-Aufnahmen zu rekonstruieren. Ob diese Mikrostruktur allerdings mit dem realen dreidimensionalen Gefüge übereinstimmt, muss zusätzlich validiert werden. Da wie in Kapitel 4.2.2 beschrieben keine reale Dreidimensionale Daten des DP800 vorliegen und eine simulative Validierung ebenfalls nicht in Frage kommt, wurde der Algorithmus mittels einer synthetisch generierten Mikrostruktur validiert. Dazu wurden verschiedene Gefüge erzeugt. Im folgenden sollen verschiedene dieser Gefüge und die mittels Algorithmus daraus rekonstruierte Mikrostruktur analysiert werden.

Gleich große Körner

Das einfachste synthetische Gefüge besteht aus exakt gleichen Körnern ohne Neigungen. Dazu wurde ein Raum der Größe 800x800x800 erzeugt und mit 1539 Körnern der Größe ($a=8$, $b=16$, $c=24$) gefüllt. Das Gefüge ist in Bild 32 zu sehen. Ein Gefüge mit gleichen Körnern wurde als erster Schritt einer Validierung gewählt, da anzunehmen ist, das die in den Arbeitshypothesen (Kapitel 4.2.2) formulierten Problemen in einem einfachen Gefüge am deutlichsten zu sehen sein sollten. Durch mehrere Schnitte durch das Gefüge wurden 281 Rekonstruierte Körner erzeugt. Da keine Neigungen vorliegen, entsprechen die Achsenbestandteile (x,y,z) direkt den Halbachsen (a,b,c). Es ist also nicht nötig, Achsenbestandteile über die Neigungen in die Halbachsen zurück zutransformieren.

Ein erster Indikator für die Qualität der Rekonstruktion sind mittleres Kornvolumen und Standardabweichung. Dies zeigt an, ob die Verteilung nicht gestreckt oder gestaucht und nicht verschoben ist. Für diesen Fall sind die Werte in Tabelle 11 (Werte gerundet) zu sehen.

Tabelle 11: *mean* und *std* für ein Gefüge mit gleich großen Körnern

	Original	Rekonstruiert
Mittelwert	12.868	7.239
Standardabweichung	0	3.658

Hierbei weichen beide Werte stark ab. Eine Kerndichteschätzung der beiden Datensätze zeigt starke Abweichung ebenso (Vgl. Bild 33). Da seaborn für einen Datensatz mit einer Varianz von 0 keinen KDE berechnen kann, wurde der Wert des Volumens des realen Gefüges mittles der durchgezogenen orangenen Linie eingefügt. Auch in diesem Bild ist deutlich zu erkennen, dass durch die Rekonstruktion (im Bild als "2D" gekennzeichnet) die Verteilung der Volumen deutlich zu breit geschätzt wird. Auch der Peak der Volumen ist deutlich geringer als im realen Gefüge.

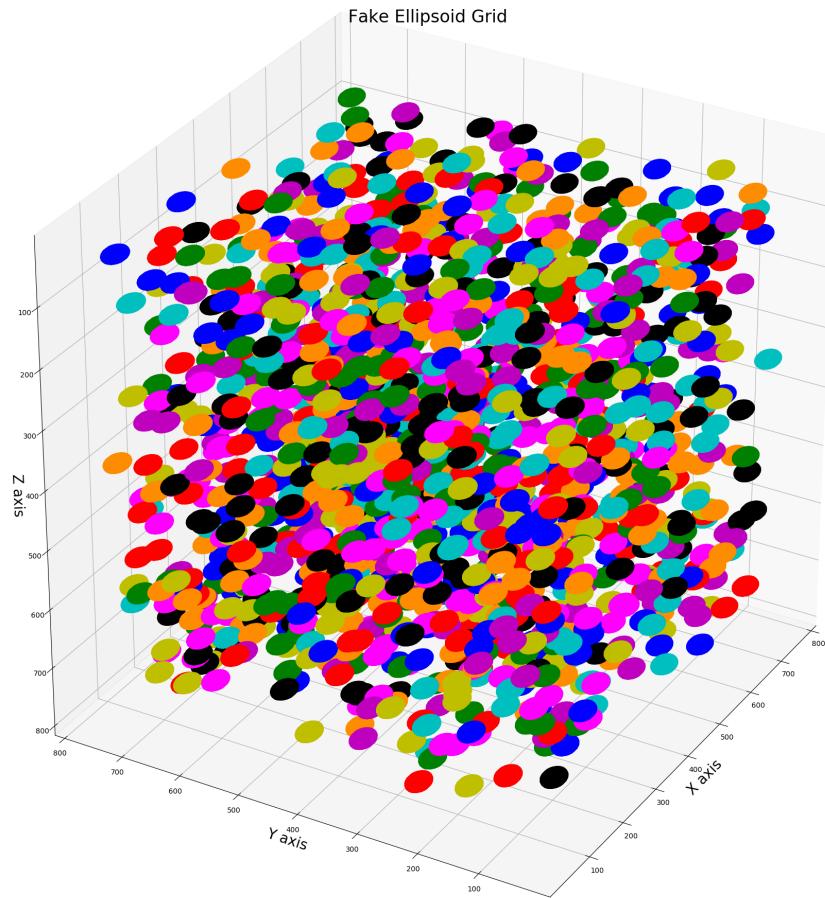


Bild 32: Dreidimensionales Gefüge mit 1539 gleich großen Ellipsoiden

Eine ähnliche Unterschätzung ist auch in den Verteilungen der Halbachsen zu erkennen. Diese Grafik weist eine starke Ähnlichkeit zur Volumenverteilung auf. Die Unterschätzung der Werte ist naheliegend, da ein Ellipsoid bei einem Schnitt exakt in der Mitte getroffen werden muss, um die Größe korrekt zu approximieren. Wird ein Ellipsoid an einer bestimmten Stelle geschnitten entsteht eine Ellipse. Sofern diese Ellipse nicht genau aus der Mitte des Ellipsoids entsteht, werden beide Halbachsen im Vergleich zum Realen Ellipsoid unterschätzt. Da mittels des Algorithmus anhand der beiden Halbachsen die fehlende Halbachse berechnet wird, ist es naheliegend, dass diese Achse ebenso unterschätzt wird. Da das Volumen eines Ellipsoids mit der Formel $\frac{4}{3} * a * b * c * \pi$ berechnet wird, wird die Volumenunterschätzung noch potenziert, falls alle drei Achsen kleiner als im Realen Gefüge sind. Dies ist auch beim Vergleich der Bilder 33 und 34 zu sehen, der Unterschied zwischen realem und rekonstruiertem Volumen ist größer als der Unterschied zwischen dem Equivalient bei den Halbachsen. Die überschätzte Breite der Verteilung kann auch gut durch die jeweiligen Schnittebenen illustriert werden. Diese entsprechen in diesem Validierungsverfahren den EBSD-Aufnahmen in der Realität. Diese Schnittbilder für den Schnitt an der Stelle 180

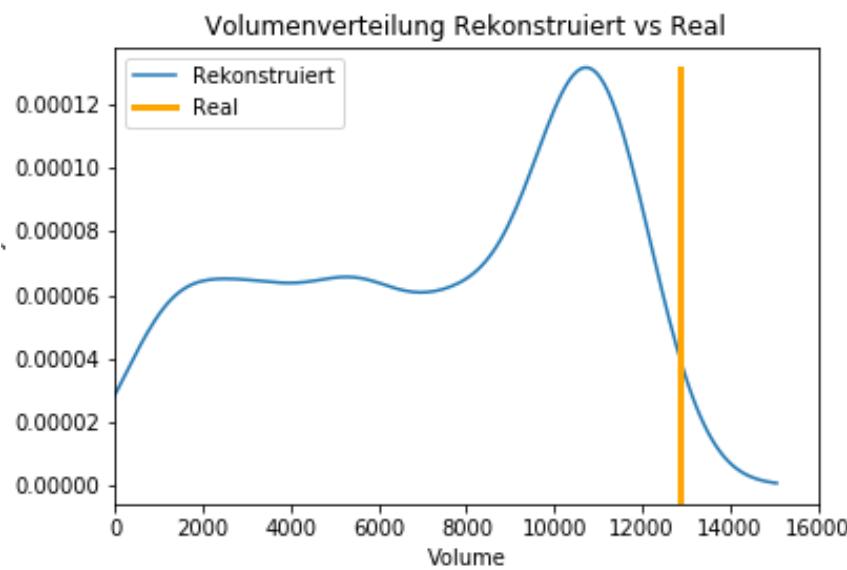


Bild 33: Volumenverteilung für gleich große Körner

von 800 sind in Bild 35 zu sehen. Links ist der Schnitt x-y, in der Mitte der Schnitt y-z und rechts der Schnitt x-z zu sehen. Dabei soll angemerkt werden, dass für die Rekonstruktion mittels der y und z Werte des mittleren Schnittes auf den x-Wert extrapoliert wurde. Dies geschieht aus den in Kapitel 5.2.1 genannten Gründen, da im Schnitt y-z die wenigsten Körner vorliegen.

Eine kritische Eigenschaft des Gefüges sind die Achsenverhältnisse (engl. Aspect Ratios), welche ein Maß für die Längung eines Korns in eine Richtung sind. Bei der Rekonstruktion des Gefüges ist es daher von hoher Wichtigkeit, diese Eigenschaften ebenfalls korrekt rekonstruieren zu können. Zudem sollten diese Ergebnisse nicht von Problemen der Achsenunterschätzung verzerrt werden, ein Korn kann zwar zu klein geschätzt werden, die Längungen können dennoch korrekt sein. Insgesamt können drei Aspect Ratios (mit zugehörigen Kehrwerten) berechnet werden, hier sind die beiden Achsenverhältnisse c/a und b/c exemplarisch dargestellt (Vgl. Bild 36).

Die realen Achsenverhältnisse sind $c/a = 3$ und $b/c = 0.66$ (durchgezogene Orange Linie). Bei diesem Vergleich wird deutlich, dass die rekonstruierten und realen Achsenverhältnisse eine sehr gute Übereinstimmung aufweisen. Abweichungen bewegen sich im Bereich weniger Prozentpunkte. Es ist wahrscheinlich, dass diese Abweichungen durch den diskreten Ellipsenfit der *openCV*-Library begründet sind, da die in Bild 35 zu sehenden Ellipsen nicht exakt bestimmt werden können. Zudem werden die Ellipsoiden nur durch diskrete Punktwolken repräsentiert, somit kann eine Schnittellipse nicht exakt bestimmt werden. Möglicherweise kann dieses Problem durch die Nutzung größerer Ellipsoiden (ggf. in einem größeren Raum) minimiert werden und die Verteilung der Verhältnisse in Bild 36 weiter gestaucht werden. Unabhängig von diesem Problem zeigt die Grafik allerdings, dass es keinen einzigen Ausreißer

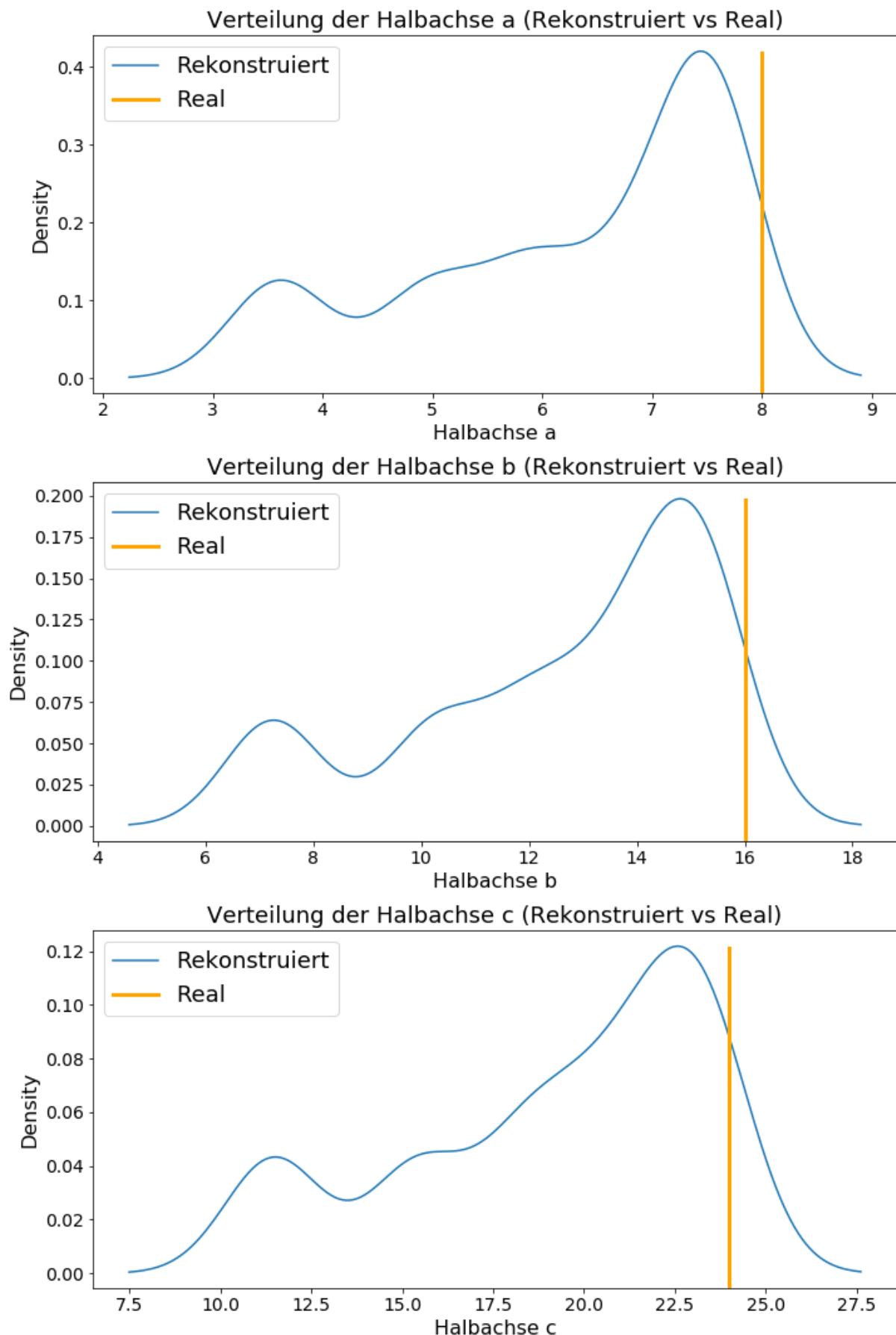


Bild 34: Halbachsenverteilung für gleich große Körner

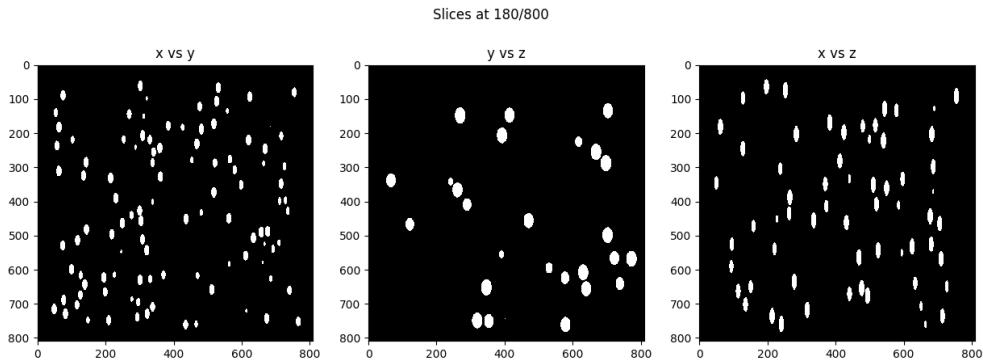


Bild 35: Schnittbilder durch das in Bild 32 zu sehende Gefüges

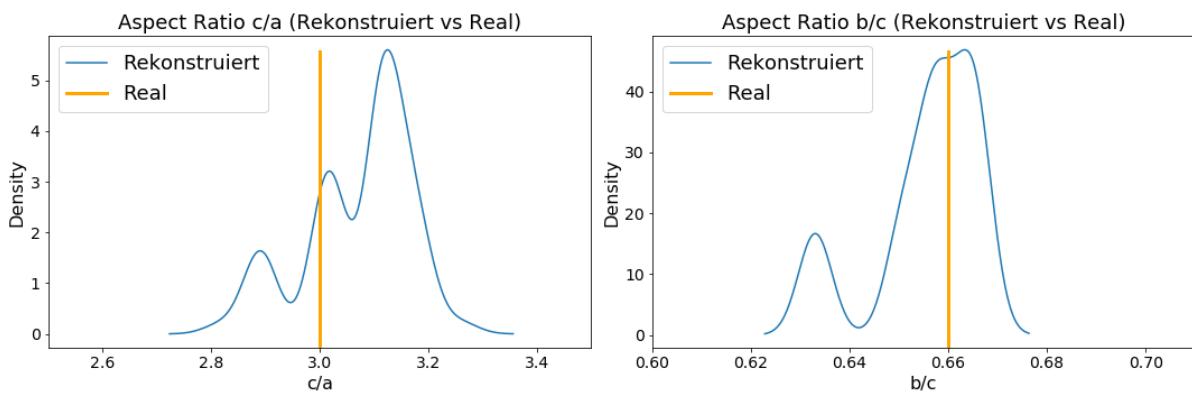


Bild 36: Achsenverhältnisse (Aspect Ratios) des rekonstruierten und realen Gefüges

gibt (die blaue KDE-Kurve wäre ansonsten bis zu diesem Punkt durchgezogen). Dies zeigt, dass durch den Algorithmus keine falschen Zuweisungen von x zu einem Paar (y,z) gemacht werden und somit alle Körner in den korrekten Größenverhältnissen rekonstruiert werden.

Diese Tatsache belegt das grundsätzliche Funktionieren des Algorithmus. Die Fehlerhaft geschätzten Verteilungen von Volumen und Halbachsen sind wie beschrieben nicht auf ein fehlerhaftes Verhalten des Algorithmus, sondern auf den grundsätzlichen Verlust von Informationen bei der Reduktion von dreidimensionalen auf zweidimensionale Daten zurückzuführen.

Realitätsnahe Gefüge

Ein Gefüge ausschließlich bestehend aus gleich großen Ellipsoiden/Körnern ist keinesfalls eine brauchbare Approximation für die reale Mikrostruktur des DP800. Da als nächste Näherung an die dreidimensionale Mikrostruktur die EBSD-Aufnahmen vorliegen, wurde ein Gefüge mit rekonstruierten Körnern durch Anwendung des Algorithmus in der Konfiguration 5/5%, drop=True erzeugt. Dies ergibt 745 Körner.

Es wurden die Achsenkomponenten (x, y, z) als Halbachsen (a, b, c) verwendet, da zum Zeitpunkt der Validierung keine Methode zur Verfügung stand, um Achsenkomponenten in Halbachsen zurückzurechnen (Vgl. Kapitel 5.2.4). Um Ellipsoiden in der passenden Größenordnung für ein $1000 \times 1000 \times 1000$ Gitter zu erzeugen, wurden die Halbachsen mit dem Faktor fünf skaliert und auf die nächste Ganzzahl gerundet. Die Skalierung ist notwendig, um diskrete Ellipsoiden in hinreichender Genauigkeit zu erzeugen.

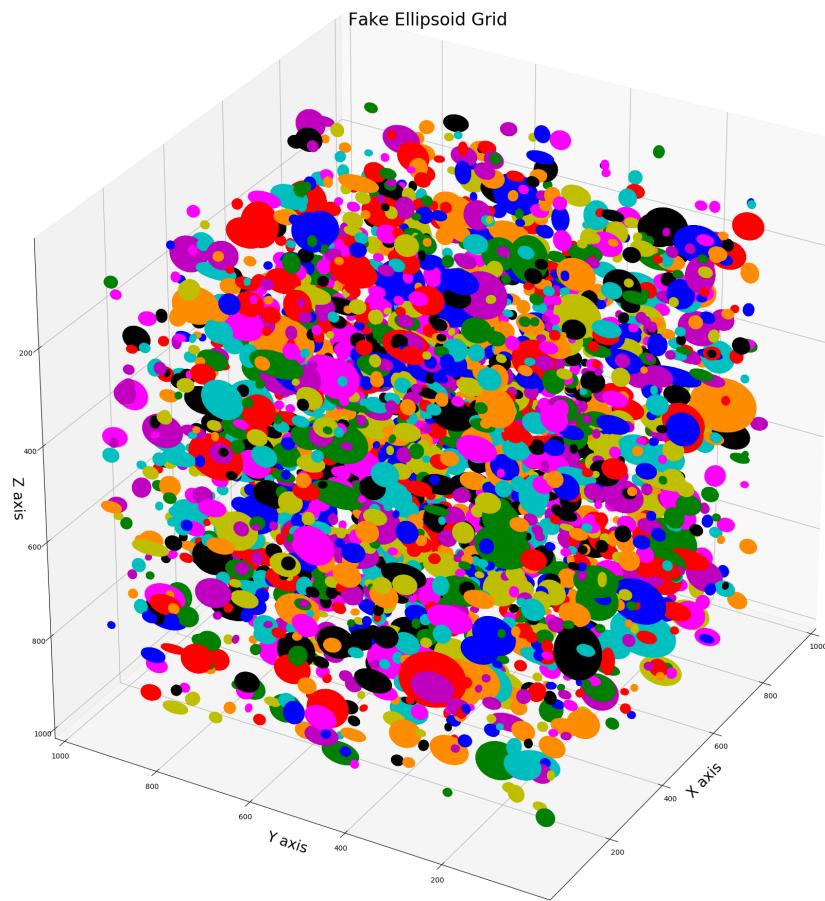


Bild 37: Dreidimensionales Gefüge mit 2589 Ellipsoiden rekonstruiert aus EBSD-Aufnahmen

Insgesamt wurden 2589 Ellipsoiden erzeugt. Somit sind einige Körner mehrfach vorhanden. Das Gefüge ist in Bild 37 zu sehen. Die Körner weisen keine Neigungen entlang einer Achse auf, daher entsprechen die Achsenkomponenten den Halbachsen.

Auch in diesem Fall soll zunächst die Volumenverteilung analysiert werden. In Tabelle 12 sind Mittelwert, Standardabweichung sowie maximales und minimales Volumen der Körner für das Originale und Rekonstruierte Gefüge zu sehen (Werte gerundet und einheitenlos). Die zugehörigen Kerndichteschätzungen sind in Bild 38 dargestellt. Für eine bessere Darstellung wurde die x-Achse auf das Intervall (0,150000)

beschränkt.

Tabelle 12: Wichtige Parameter für die Volumenverteilung des in Bild 37 zu sehenden Gefüges

	Original	Rekonstruiert
Mittelwert	13,354	13,521
Standardabweichung	32,553	22,824
Max. Vol.	540,672	153,775
Min. Vol.	126	75

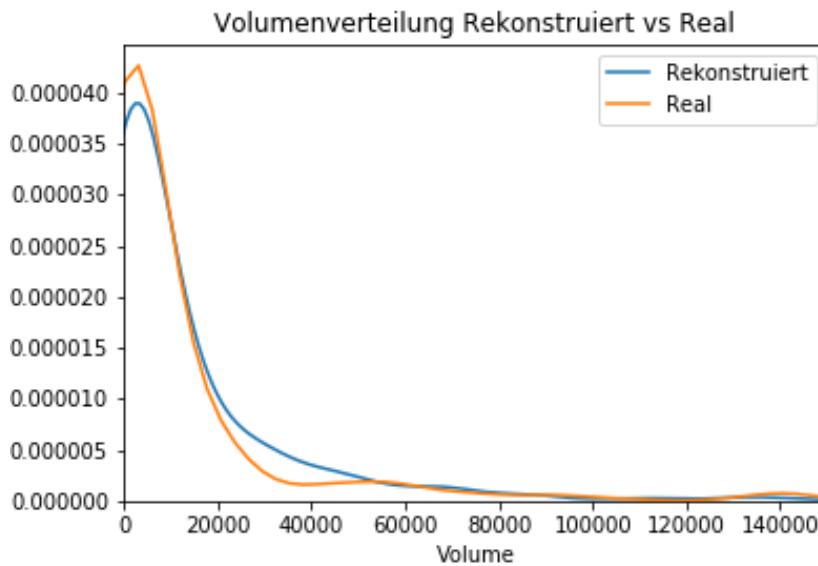


Bild 38: Volumenverteilung für das Gefüge in Bild 37

Im Gegensatz zu einem Gefüge mit gleich großen Körnern (Bild 33 und Tabelle 11) sind die Mittelwerte fast gleich, dies impliziert eine nicht vorhandene Unterschätzung der Verteilung der Kornvolumen. Dennoch ist das kleinste Volumen der rekonstruierten Mikrostruktur mit ca. 75 nur etwas mehr als halb so groß wie das kleinste originale Volumen mit 126. Da kleinere Volumen kommt dadurch zustande, dass die Ellipsoiden durch randnahe Schnitte durch die Körner unterschätzt werden. Diese Unterschätzung hat allerdings offenbar keine Auswirkung auf den Mittelwert. Bild 38 stützt diese Annahme, da die Peaks der KDE's nahezu passgenau übereinstimmen. Weiterhin lassen weder die KDE's noch die Parameter in Tabelle 12 eine falsche Verteilung der rekonstruierten Daten im Bezug auf die Verteilungsbreite erkennen. Die höhere Standardabweichung der Originalen im Vergleich zu den Rekonstruierten Daten ist voraussichtlich durch die sehr großen Ausreißer in den realen Daten begründet, welche nicht rekonstruiert werden (Das größte reale Volumen ist fast vier mal größer als das größte rekonstruierte Volumen).

Auch der Vergleich der KDE's aller drei Halbachsen zeigt eine gute Rekonstruktion

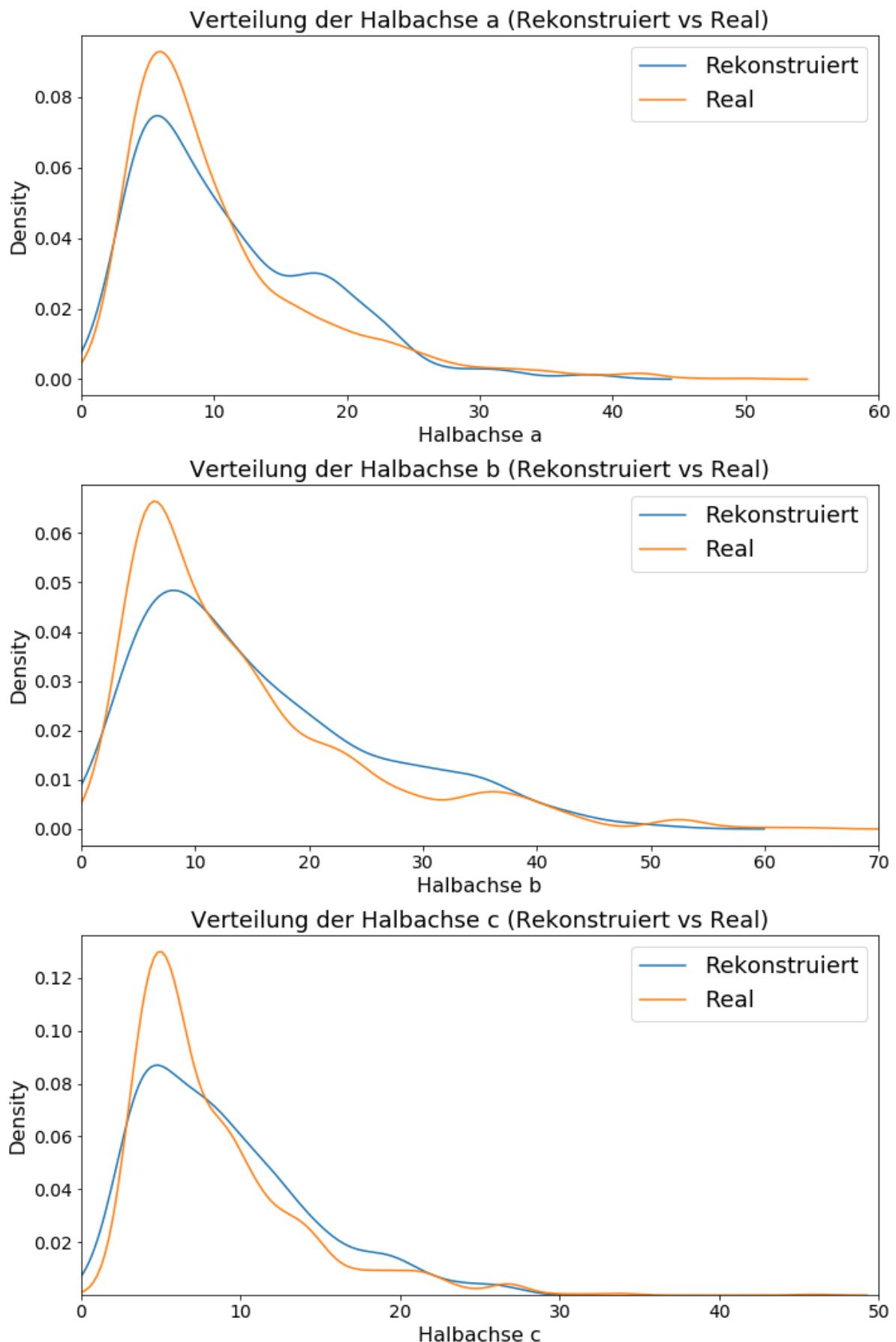


Bild 39: Halbachsen für das Gefüge in Bild 37

des Gefüges (Vgl. Bild 39) Auch hier weisen die Peaks eine gute Übereinstimmung aus, während die Breite der rekonstruierten Verteilung allerdings höher zu sein scheint, dies ist durch den niedrigeren Peak und die höheren Dichten der blauen Schätzung rechts des Peaks zu sehen. Dies deckt sich mit den Erkenntnissen aus Bild 34, allerdings ist die rekonstruierte Verteilung in diesem Fall nicht annähernd so viel breiter als die reale Verteilung. Bild 40 zeigt die Verteilung der Achsenverhältnisse (analog zu Bild 36). In diesem Fall sind die Achsenverhältnisse im realen Gefüge kein Konstanter Wert, sondern folgen auch einer Verteilung. Hier wird deutlich, dass die Verteilung der Achsenverhältnisse der rekonstruierten und echten Körner nahezu übereinstimmen. Dies zeigt, dass mit dem Algorithmus die Größenverhältnisse der Körner hervorragend rekonstruiert werden können. In Kombination mit der guten Volumen- und Halbachsenschätzung (Bilder 38 und 39) bedeutet dies, dass das Gefüge durch den Algorithmus in sehr guter Näherung abgebildet wird.

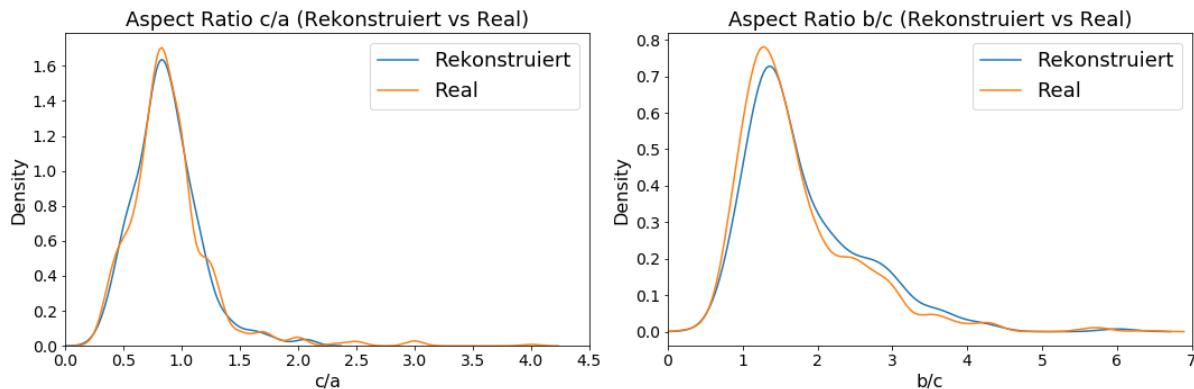


Bild 40: Achsenverhältnisse (Aspect Ratios) des rekonstruierten und realen Gefüges

Im Vergleich zu dem Gefüge mit ausschließlich gleichen Körnern zeigen rekonstruiertes und echtes Gefüge in diesem Fall eine deutlich bessere Übereinstimmung. Da die zu einer höheren Verteilungsbreite und Größenunterschätzung führenden und bereits diskutierten Effekte auch in diesem Fall vorliegen müssen (Auch hier müssen Körner für eine exakte Bestimmung des Volumens exakt in der Mitte geschnitten werden, was für alle Körnern unmöglich ist), müssen andere Effekte vorhanden sein, um die Volumenunterschätzung aufzuheben. Wahrscheinlich ist hierfür ein "Übersehen" der kleinen Körner durch den Algorithmus verantwortlich. Dies bedeutet, dass sehr kleine Ellipsoiden entweder auf dem Schnitt durch das Gefüge gar nicht zu sehen oder so klein sind, dass der Algorithmus diese Körner nicht zusammensetzen kann. Vermutlich liegt ein Zusammenspiel beider Effekte vor. Bildlich gesprochen verschieben diese beiden Effekte die blauen Kurven in Bild 38 im Vergleich zur Verteilung der Realdaten (orange) nach rechts zu größeren Werten, während die Unterschätzung durch nicht mittige Schnitte die blaue Kurve nach links zu kleineren Werten verschiebt. Somit heben sich beide Effekte auf und das rekonstruierte entspricht dem realen Gefüge.

5.2.3 Einfluss der Hyperparameter *threshold/range* auf die Rekonstruktion des Gefüges

Das in Bild 37 zu sehende Gefüge wurde mit der Hyperparameterkonfiguration $threshold = 15\%$ und $range = 15\%$ und $drop = False$ rekonstruiert, daraus ergeben sich z.B. die Volumenverteilungen in Bild 38. Diese Konfiguration wurde gewählt, da die geringe Anzahl von 2598 Körnern im Gefüge zu wenigen Ellipsen auf den Schnitten führt. Da Kapitel 5.2.1 gezeigt hat, dass verschiedene Hyperparameter zu verschiedenen Ergebnissen des Algorithmus führen, soll nun zusätzlich geprüft werden, ob diese Hyperparameter einen Einfluss auf die Rekonstruktion der Verteilungen (Volumen und Halbachsen) haben. Dazu wurden zusätzlich zwei Konfigurationen getestet: Erstens wurden *threshold* und *range* auf 5% gesetzt, hiermit soll geprüft werden, ob dies bei wenigen Körnern auf einer Aufnahme zu Problemen führt. Zweitens wurde der *drop*-Parameter auf *False* gesetzt, um den eventuellen Einfluss vieler Körner mit gleichen Achsen (diese entstehen durch das fehlende Rauslöschen von Körnern) auf die Rekonstruktion zu überprüfen. Auch hierzu wurden erneut Volumen-, Halbachsen, und Aspect Ratio Verteilung verglichen. Die Verteilung der Volumen ist in Bild 41 zu sehen.

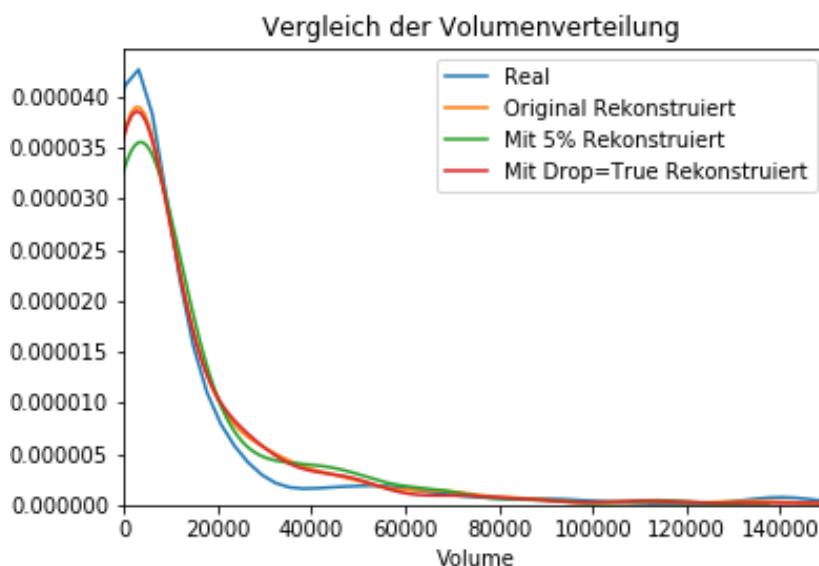


Bild 41: Volumenverteilung für das Gefüge in Bild 37 rekonstruiert mit verschiedenen Hyperparameterkonfigurationen (Original bedeutet range und threshold = 15%)

Hierbei kennzeichnet “2D_Original” die Konfiguration aus Bild 38 (Beide Parameter gleich 15% und Drop=False). Deutlich wird, dass diese Originale Konfiguration und die Änderung des drop-Parameters auf True bei dieser quasi identisch verlaufen (rote und orange Linie). Dies impliziert, dass der Einfluss des Parameters gering bis nicht vorhanden ist. Im Gegensatz hierzu ist die Verteilung bei der Wahl beider Parameter auf 5% im Peak etwas niedriger, horizontal können alle Peaks allerdings als identisch

angenommen werden. Im Bereich der Volumen von ca. 20.000 bis 60.000 liegen die Kurven der rekonstruierten Volumen über der Kurve des realen Volumens, dies liegt an der zu breit geschätzten Verteilung.

In Bild 42 sind die Verteilungen der Halbachsen zu sehen. Die Kennzeichnungen der Kurven sind analog zur Grafik in Bild 41. Hier sind ähnliche Effekte zu sehen, auch hier sind die Kurven für die Originale Konfiguration und die der Konfiguration mit aktiviertem Drop quasi identisch. Auffällig ist allerdings, dass die Kurve bei Halbachse b beim Setzen beider Hyperparameter auf 5% eine starke Abweichung zeigt. Dabei ist der Peak deutlich zu größeren Werten hin verschoben. Die Abweichung dieser Konfiguration wird auch in der Darstellung der Aspect Ratios deutlich (Vgl. Bild 43). Auch in dieser Darstellung ist die Abweichung der 5% Konfiguration deutlich. Auch hier stimmen die beiden anderen Konfigurationen und die Verteilung der realen Aspect Ratios sehr gut überein.

Es existieren keine Hinweise darauf, dass die Konfiguration mit jeweils 5% logisch schlechter sein sollte. Der Vergleich der Konfigurationen zwei und drei in Kapitel 5.2.1 zeigt im Gegensatz dazu sogar, dass die Übereinstimmung zwischen realen und rekonstruierten Daten bei der Analyse über RegPlots für 5% besser ist (Zu sehen in den Steigungsgesetzen, vgl. Bilder 30 und 31). Eine Erklärung für die schlechteren Ergebnisse liegt darin begründet, dass mit der Wahl beider Parameter gleich 5% nur ca. 100 Körner rekonstruiert werden können, im Vergleich dazu ca. 400 in der Originalen und 300 beim Setzen von drop auf True. Naheliegend ist, dass die rekonstruierten 100 Körner keine statistische Repräsentativität aufweisen und dementsprechend verzerrt sind.

Abschließend kann gesagt werden, dass die Wahl der Hyperparameter vermutlich nur indirekt über die Anzahl der Körner und die damit einhergehende statistische Reproduzierbarkeit einen Einfluss auf die Qualität der Rekonstruktion hat. Somit sollten die Parameter so gewählt werden, dass eine ausreichende Anzahl an Körnern des Gefüges reproduziert werden kann.

5.2.4 Rekonstruktion der Drehungen

Wie bereits in Kapitel 4.2.1 beschrieben wurde, arbeitet der Algorithmus mit den Achsenkomponenten (x,y,z) eines Ellipsoids und nicht direkt mit den Halbachsen (a,b,c). Die im vorherigen Kapitel durchgeföhrten Analysen müssen dieses Problem nicht berücksichtigen, da keine Drehungen vorliegen und somit das Tupel (x,y,z) dem Tupel (a,b,c) entspricht. Dies ist in der Realität allerdings meist nicht zutreffend, im Normalfall entspricht der beste Fit einer Ellipse an ein bestimmtes Korn im EBSD-Bild einer geneigten Ellipse. Somit ist es notwendig, dass durch den Algorithmus auch ein Gefüge mit geneigten Körnern abgebildet werden kann. Dazu müssen die nach Anwendung

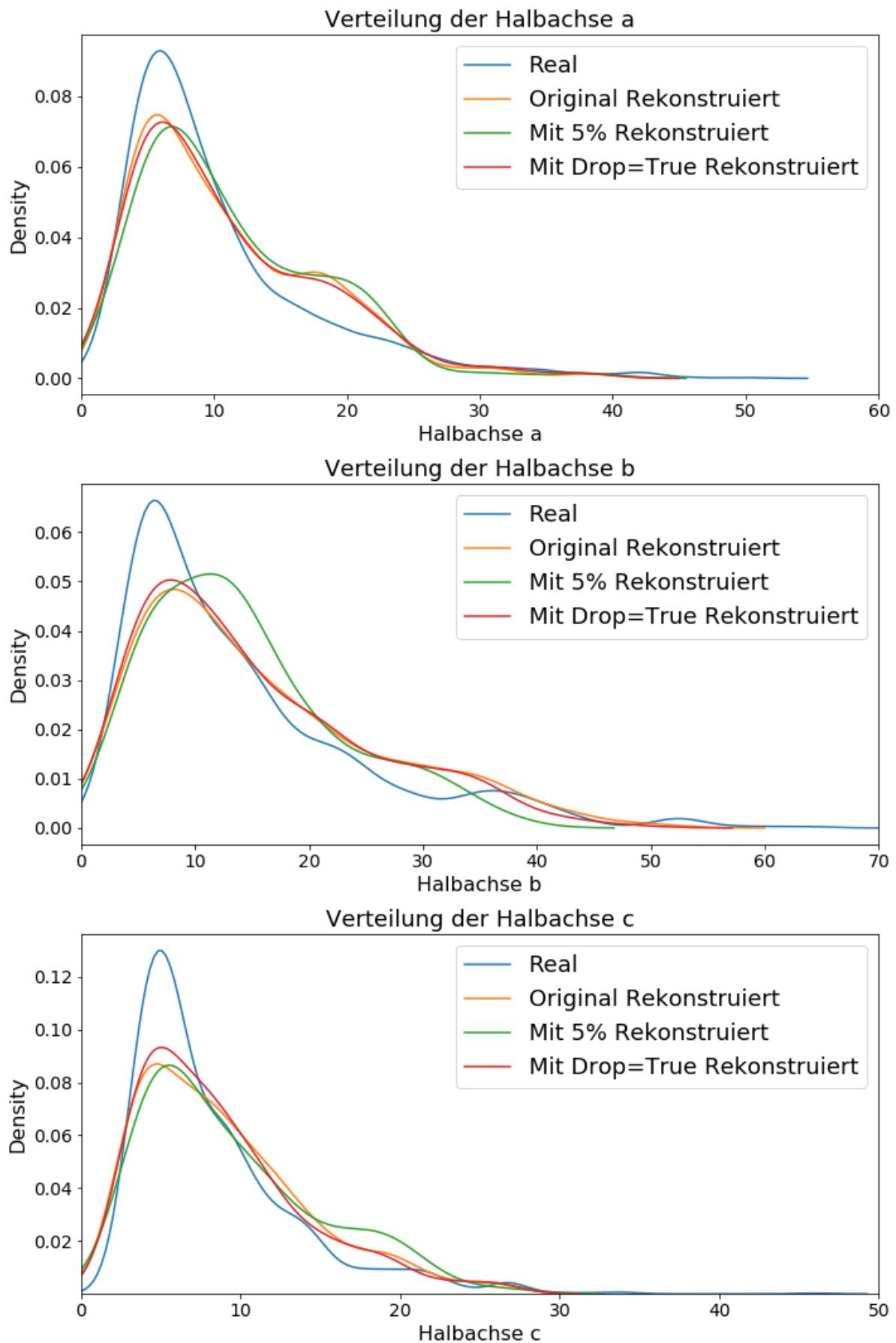


Bild 42: Halbachsen für das Gefüge in Bild 37 rekonstruiert mit verschiedenen Hyperparameterkonfigurationen (Original bedeutet range und threshold = 15%)

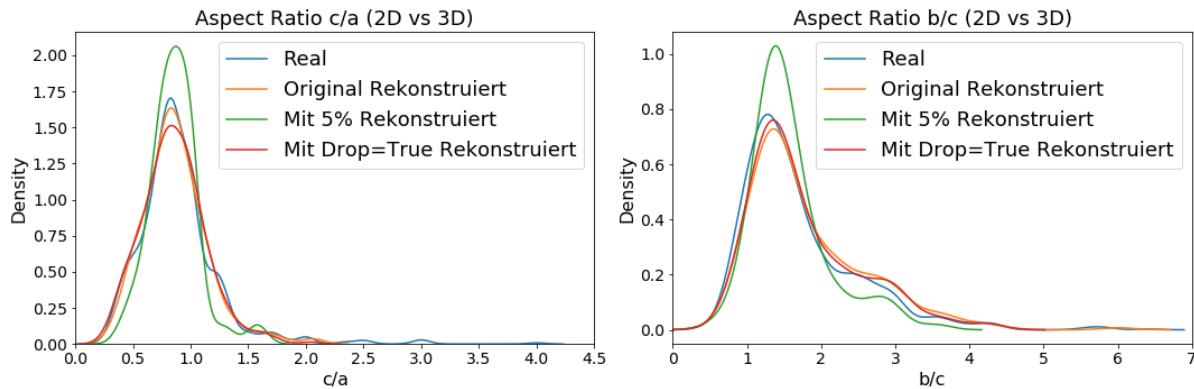


Bild 43: Achsenverhältnisse (Aspect Ratios) des rekonstruierten und realen Gefüges mit verschiedenen Hyperparameterkonfigurationen (Original bedeutet range und threshold = 15%)

des Algorithmus vorliegenden Tupel (x,y,z) in ein passendes Tupel (a,b,c) zurückgerechnet werden, hierbei werden die aus den jeweiligen Schnittbildern gewonnenen Winkel (α, β, γ) verwendet. Der Winkel α ist die Drehung aus dem Schnittbild $a \times b$, β aus $c \times b$ und γ aus $c \times a$. Zum Zurückrechnen kann das Gleichungssystem aus Kapitel 4.2.1 modifiziert werden:

$$\text{Aus } a \times b = \begin{cases} sum_x = |\cos \alpha| * a + \sin \alpha * b \\ sum_y = \sin \alpha * a + |\cos \alpha| * b \end{cases} \quad (33)$$

$$\text{Aus } c \times b = \begin{cases} sum_z = |\cos \beta| * c + \sin \beta * b \end{cases} \quad (34)$$

$$\text{Aus } c \times a = \begin{cases} sum_z = |\cos \gamma| * c + \sin \gamma * a \end{cases} \quad (35)$$

Zu erkennen ist, dass das System *überbestimmt* ist, da die fehlende Komponente sum_z durch zwei verschiedene Gleichungen berechnet werden kann. Damit stehen vier Gleichungen für drei Unbekannte zur Verfügung. Theoretisch ist auch eine Bestimmung mit drei Gleichungen mit drei Winkeln möglich:

$$sum_x = |\cos \alpha| * a + \sin \alpha * b \quad (36)$$

$$sum_y = \sin \beta * c + |\cos \beta| * b \quad (37)$$

$$sum_z = |\cos \gamma| * c + \sin \gamma * a \quad (38)$$

Insgesamt stehen mehr Möglichkeiten zur Verfügung als grundsätzlich benötigt werden. Die Gleichungssysteme können durch Verwendung eines CAS (Computer algebra system) effizient für jedes Tupel $(x,y,z,\alpha,\beta,\gamma)$ gelöst werden. In dieser Arbeit wurde die *sympy*-Bibliothek verwendet [38].

In der Praxis wird allerdings deutlich, dass die obigen Gleichungssysteme nicht vollumfänglich vernünftige Ergebnisse liefern können. Werden die obigen Gleichungssysteme auf einen durch den Algorithmus erzeugten Datensatz angewandt (siehe Kapitel 5.2.1), können die jeweiligen Halbachsen durch die Histogramme in Bild 44 visualisiert werden.

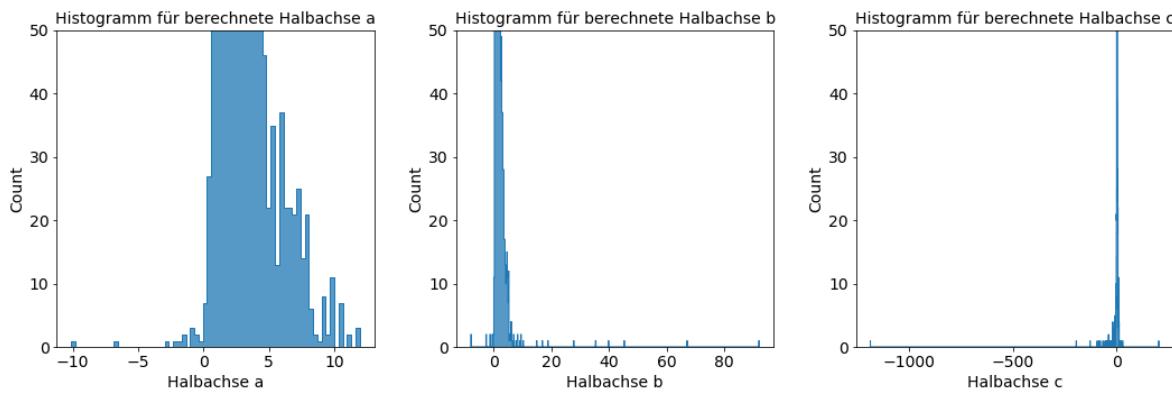


Bild 44: Histogramme für mit den Gleichungen 35 und 38 erzeugten Halbachsen (y-Achse gekürzt)

Die Darstellung als Histogramm im Vergleich zu den bislang gewählten Kerndichteschätzungen wurden gewählt, weil in den Histogrammen sofort Fehler zu sehen sind. Es werden sowohl negative Achsen als auch sehr große Achsen erzeugt, beides ist in der Realität unmöglich (v.a. negative Achsen). Somit werden durch die Gleichungen im Materialwissenschaftlichen Sinne unsinnige Ergebnisse erzeugt (auch wenn das jeweilige Gleichungssystem mathematisch korrekt lösbar ist). Ungeklärt bleibt, ob die entstandenen Körner eine gute Rekonstruktion für die echte dreidimensionale Mikrostruktur sind.

Die nicht verwertbaren Ergebnisse resultieren aus der Hinzufügung der Gleichung 35 zum ursprünglichen Gleichungssystem. Auch wenn die Punkte x, y und z durch den Algorithmus zusammenpassen, setzt dies nicht automatisch voraus, dass durch die jeweiligen Winkel (welche im Algorithmus unberücksichtigt bleiben) aus den zusammengesetzten Komponenten x, y, z valide Ellipsoiden mit (a, b, c) erzeugt werden können.

Um dieses Problem zu beheben und zu überprüfen, ob die so aus (x, y, z) berechneten Ellipsoiden eine gute Rekonstruktion für die dreidimensionale Mikrostruktur sind, wird ähnlich wie im vorherigen Kapitel vorgegangen: Zunächst wird ein künstliches Gefüge erzeugt, bei dem zu den bereits in Bild 37 verwendeten Körner die passenden Drehungen hinzugefügt werden. Diese wurden den EBSD-Aufnahmen entnommen, um möglichst realitätsnahe Neigungen zu erhalten. Dieses Gefüge ist in Bild 45 zu sehen.

Auf dieses Gefüge wird der Algorithmus wie bekannt angewandt. Nach Berechnung

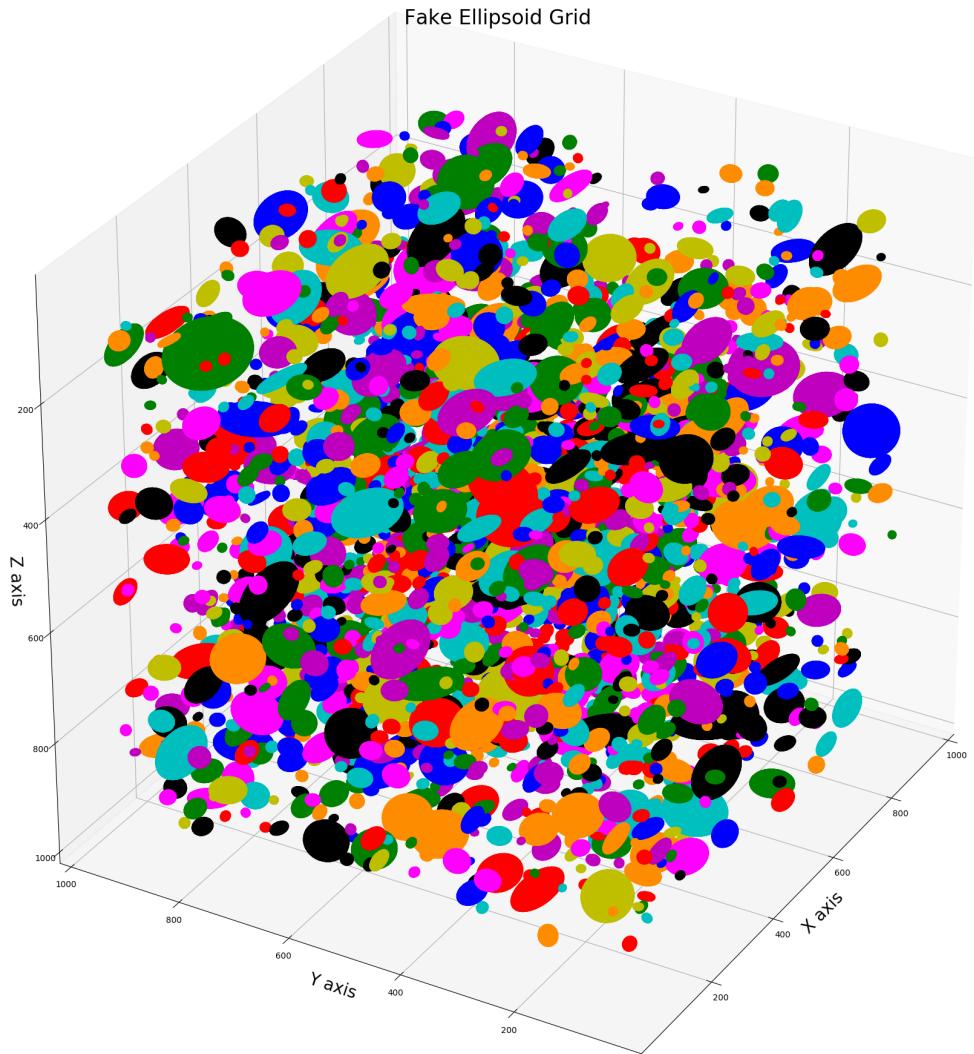


Bild 45: Synthetisches 3D Gefüge mit gedrehten Ellipsoiden

der Achsenkomponenten (x,y,z) werden die Gleichungen 35 und 38 verwendet um aus diesen Komponententupeln die Halbachsen zu berechnen. Anschließend werden alle *nicht-plausiblen* Körner entfernt. Allerdings muss klar definiert werden, welche Eigenchaften als *unplausibel* gelten. Klar ist, dass Halbachsen mit einer Länge kleiner 0 oder deutlich größer 100 nicht valide sein können, allerdings sind unrealistische Ergebnisse unter Umständen nicht so einfach zu erkennen. Daher wurde ein Ansatz entwickelt, bei dem Grenzen für die Plausibilität aus den 2D-Aufnahmen (den EBSD's) berechnet wurden. Hierzu wurden alle Ellipsen aus den zweidimensionalen Aufnahmen berechnet und die jeweiligen Aspect Ratios a/b , c/a und b/c berechnet. Zusätzlich wurden für jede Halbachse a, b und c der größte Wert berechnet. Dieser Wert multipliziert mit einem Skalierungsfaktor ergibt die obere Grenze für jede berechnete Halbachse. Die Idee hinter diesem Ansatz ist, dass die Halbachsen der dreidimensionalen Ellipsoiden

die der zweidimensional zu sehenden Ellipsen nicht deutlich übersteigen können. Ebenso können die Aspect Ratios nicht stark abweichen.

Die Auswertung erfolgt analog zu den Gefügen ohne Winkel. In Bild 46 ist der Vergleich der Volumenverteilungen zu sehen. Zugehörige Eigenschaften sind in Tabelle 13 zu finden.

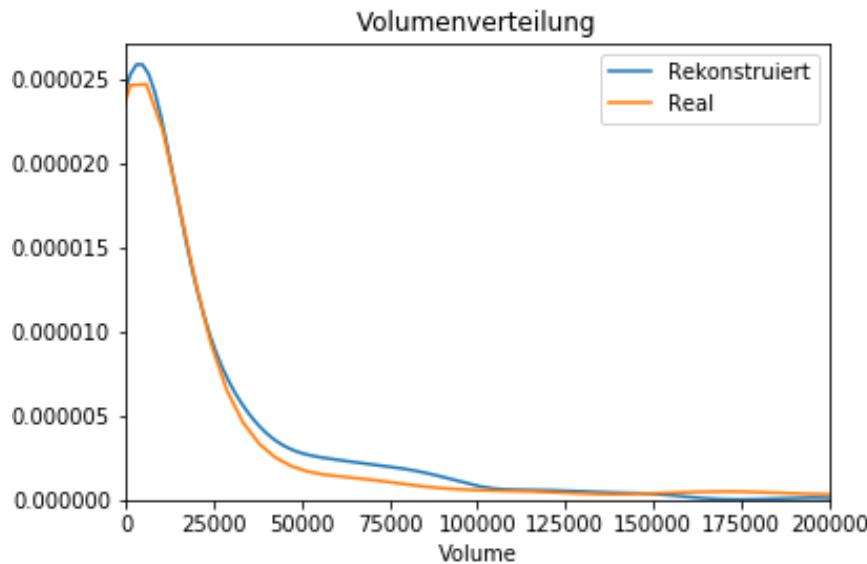


Bild 46: Vergleich Rekonstruierte und reales Volumen bei einem Gefüge mit Neigungen

Tabelle 13: Wichtige Parameter für die Volumenverteilung des in Bild 46 zu sehenden Gefüges

	Original	Rekonstruiert
Mittelwert	23161.34485165837	21230.812046873674
Standardabweichung	56252.540199749346	37686.54488326189
Max. Vol.	839370.7251861208	287262.0571369881
Min. Vol.	134.0412865531645	42.98559935384444

Die Ergebnisse sind ziemlich ähnlich zu den bei der Analysen eines Gefüges ohne Neigungen (Vgl. Kapitel 5.2.2). Die Kerndichteschätzungen der Volumen sind nahezu identisch, somit wird das Volumen sehr gut rekonstruiert. Entsprechend sind auch die Mittelwerte sehr ähnlich. Die abweichende Standardabweichung kann durch die großen Ausreißer bei den realen dreidimensionalen Daten erklärt werden, das größte dreidimensionale Volumen ist mehr als drei mal so groß wie das größte Rekonstruierte Volumen.

Die Halbachsen sind in Bild 47 zu sehen. Hier treten größere Änderungen auf im Vergleich zu Bild 39. Zum einen fällt auf, dass die Halbachse c deutlich besser rekonstruiert wurde, im Vergleich dazu wurden die anderen beiden Halbachsen a und b

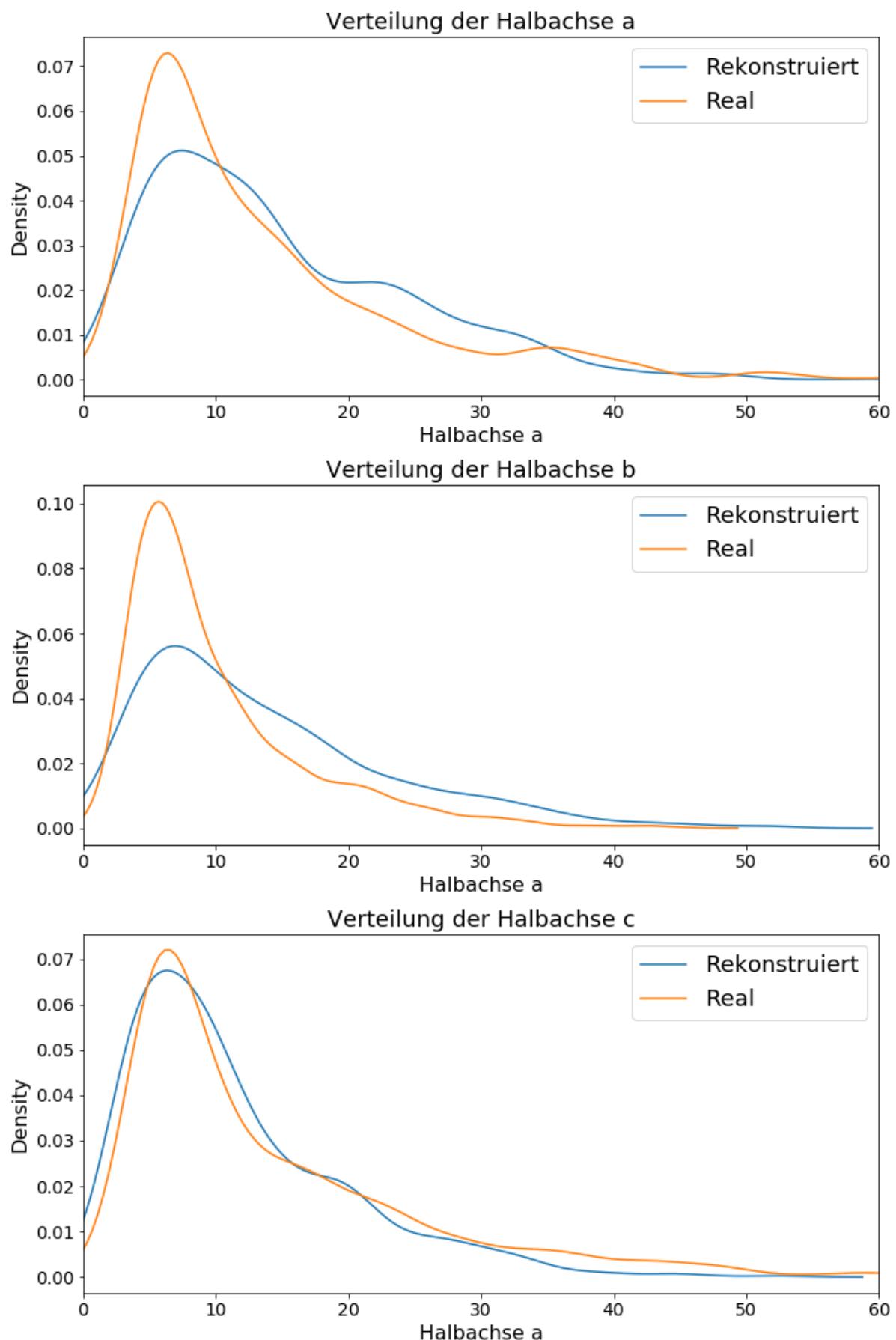


Bild 47: Vergleich Rekonstruierte und reale Halbachsen bei einem Gefüge mit Neigungen

schlechter rekonstruiert. Besonders die Verteilung für die Halbachse b weicht relativ stark von der Verteilung der echten Halbachse ab, der Peak liegt nur bei etwa der halben Dichte (0.1 vs. 0.06). Dies deutet auf eine insgesamt breitere Verteilung hin. Für die Halbachse a gilt dies ebenso, der Unterschied ist allerdings weniger stark ausgeprägt (0.07 vs. 0.05). Positiv ist allerdings, dass die Peaks bei allen drei Halbachsen den selben Wert haben und somit übereinander liegen. Auch hier ist wieder zu sehen, dass die rekonstruierte Verteilung breiter ist als die reale dreidimensionale Verteilung.

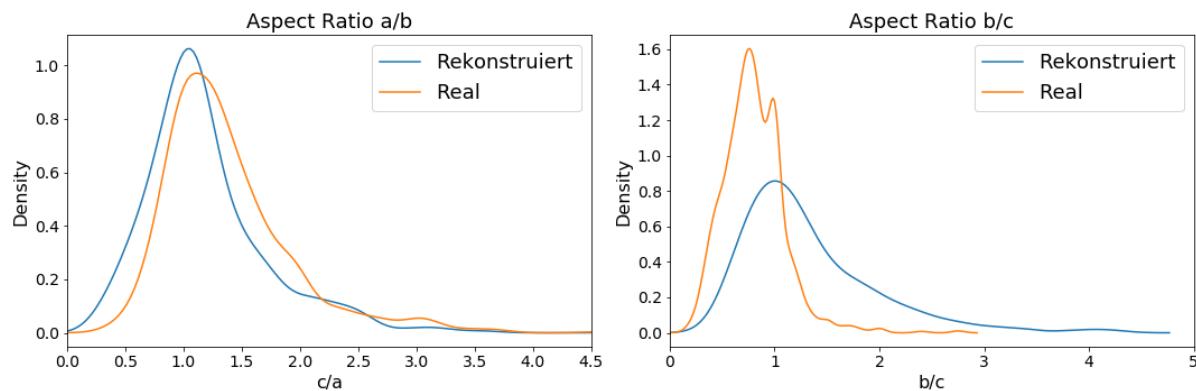


Bild 48: Vergleich Rekonstruierte und reale Aspect Ratios bei einem Gefüge mit Neigungen

Bild 48 stellt die Kerndichteschätzungen der Aspect Ratios (rekonstruiert und real) dar. Hier wird deutlich, dass im Vergleich zu den rekonstruierten Aspect Ratios bei einem Gefüge ohne Neigungen (Vgl. Bild 40) hier deutlich stärkere Abweichungen auftreten (Beim Gefüge ohne Neigungen gibt es quasi keine Abweichungen der Schätzer). Insbesondere weichen hier auch die Peaks horizontal ab, besonders beim Verhältnis b/c ist die Kurve der rekonstruierten Aspect Ratios stärker nach rechts verschoben, zudem treten deutlich größere Aspect Ratios auf, zu erkennen an der blauen Kurve, welche bis über vier hinaus reicht, während die orangefarbene Kurve nur bis drei geht. Dennoch gibt es keine gravierenden Abweichungen bei den Aspect Ratios, was auch durch das bereits erwähnte Verfahren begründet ist, dass Ellipsoiden mit zu großen Aspect Ratios verworfen werden.

Insgesamt ist zu sehen, dass die Rekonstruktion des Gefüges durch die Neigungen der Körner erschwert wird, besonders deutlich zeigt sich dies bei den Aspect Ratios. Vermutlich folgen die Drehungen keinem sonderlich strikten Muster, sodass die Rekonstruktion hierdurch erschwert wird und die Aspect Ratios nicht mehr so gut rekonstruiert werden können. Dies könnte geprüft werden, indem die Körner mit Drehungen nach einer strikteren Verteilung gedreht werden. Grundsätzlich ist das in diesem Abschnitt vorgestellte Verfahren relativ aufwendig. Zum einen muss zum Lösen von möglicherweise mehreren tausend Gleichungssystemen (3 Gleichungssysteme pro Ellipsoid, im Falle von 750 Körnern also 2250) Zeit und Rechenkapazität aufgewandt werden, zum anderen ist nicht garantiert, dass nach der Berechnung der

Körner und dem Verwerfen unpassender Körner noch ausreichend Körner zu Verfügung stehen. Dennoch liefert der Ansatz grundsätzlich eine gute Übereinstimmung von rekonstruierter und realer Mikrostruktur.

Ein zweiter Ansatz für den Umgang mit einer Mikrostruktur aus geneigten Körnern basiert auf der Komplexitätsreduzierung des Algorithmus. Wie in Kapitel 4.2.1 erläutert, kann der Algorithmus statt mit den Achsenkomponenten (x,y,z) auch direkt mit den Halbachsen (a,b,c) arbeiten. Da diese bei einer Neigung allerdings nicht eindeutig sind, müssen die Achsen je nach Winkel getauscht werden, da die Ellipsen bei einer Berechnung durch eine Bildverarbeitungssoftware (egal ob MTEX oder open-cv) wie bereits beschrieben in der Form (Major, Minor, Angle) vorliegen. Wie erläutert, ist ein reines Tauschen jedoch grundsätzlich zu vermeiden, da der Winkel hier nur begrenzt berücksichtigt wird. Der Vorteil ist allerdings, dass direkt mit Halbachsen gearbeitet werden kann und nach der Anwendung des Algorithmus statt aufwendig in (a,b,c) zu übersetzenden Tupeln (x,y,z) direkt die Halbachsen vorliegen. Für ein solchen Vorgehen wurde geprüft, ob es die reale dreidimensionale Mikrostruktur möglicherweise besser rekonstruiert als der im vorherigen Abschnitt diskutierte Ansatz. Als Intervall zum Tauschen wurden das Intervall (45,135) gewählt. Der Grund ist, dass ab einem Winkel von 45° (respektive 135°) der Anteil der jeweils anderen Achsenkomponente dominiert ($\cos(45)^2 = 0.5$).

Das zur Validierung verwendete Gefüge bleibt gleich, dies ist in Bild 45 zu sehen. Der Algorithmus wurde geringfügig modifiziert, sodass direkt die Halbachsen (a,b,c) verwendet werden. Nach Anwendung des Algorithmus ist keine zusätzliche Aufbereitung notwendig, es liegen direkt Ellipsoide der Form (a,b,c) statt Achsenkomponenten (x,y,z) vor. Die Volumenverteilung der so rekonstruierten Körner ist in Bild 49 zu sehen:

Diese Grafik liefert ähnliche Resultate wie die bisherigen Analysen, die Verteilungen weisen eine sehr gute Übereinstimmung auf. Insgesamt liegen die Peaks beider Schätzer horizontal nahezu übereinander, nur vertikal ist eine geringfügige Abweichung zu sehen. Diese Erkenntnisse werden durch die Parameter in Tabelle 14 gedeckt.

Tabelle 14: Wichtige Parameter für die Volumenverteilung des in Bild 46 zu sehenden Gefüges bei der Rekonstruktion des dreidimensionalen Gefüges mittels Tauschen der Halbachsen

	Original	Rekonstruiert
Mittelwert	23179.72412194018	22295.705483923903
Standardabweichung	56271.46905142151	38571.394173811546
Max. Vol.	839370.7251861208	253421.71838595398
Min. Vol.	134.0412865531645	53.45876762607213

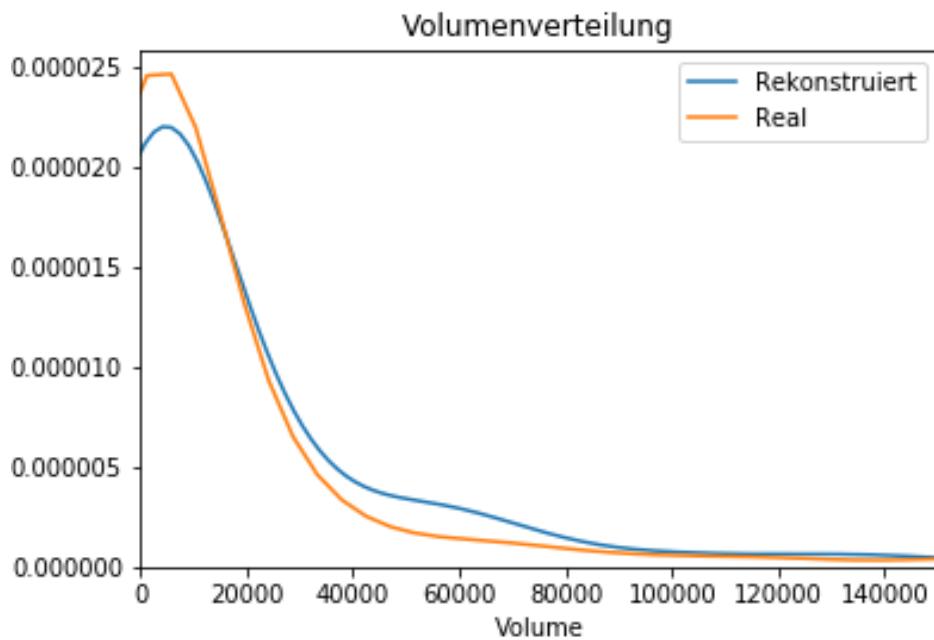


Bild 49: Vergleich der Volumenverteilung für rekonstruierte und reale Körner mittels Tauschen der Achsen

Die Parameter für die zweidimensionale Mikrostruktur stimmen im wesentlichen mit denen in Tabelle 13 überein. Es gibt keine nennenswerten Abweichungen.

Bild 50 zeigt die Halbachsenverteilungen. Auch hier werden größere Unterschiede zum Bild 47 deutlich. Auch hier sind die Abweichungen bei den Achsen a und b deutlich größer als bei der Halbachse c. Auch zeigt sich bei allen Rekonstruierten Halbachsen eine zu breit geschätzte Verteilung (zu erkennen am vertikal niedriger liegenden Peak). Positiv ist jedoch, dass die Peaks horizontal eine gute Übereinstimmung zeigen.

In Bild 51 sind die Aspect Ratios für die Halbachsen aufgetragen. Auch in diesem Fall ist die Übereinstimmung nicht so passgenau wie bei einem Gefüge ohne Neigungen. Bei Verhältnis a/b liegen die Peaks (die Mittelwerte) horizontal sehr genau übereinander, in diesem Fall ist jedoch die reale dreidimensionale Verteilung breiter (niedrigerer Peak). Im Gegensatz dazu ist beim Verhältnis b/c der Peak der rekonstruierten Aspect Ratios eher geringfügig nach Rechts verschoben. Positiv ist, dass bei keinem rekonstruierten Achsenverhältnis gehäufte Ausreißer auftreten (Zu sehen in Form von zusätzlichen Peaks).

Auch dieser Ansatz liefert insgesamt gute Ergebnisse, ähnlich zum bereits Diskutierten Ansatz der Zurückrechnung der Achsenkomponenten in Halbachsen. Auffällig ist, dass beide Ansätze vor allem bei den Halbachsen sehr ähnliche Ergebnisse liefern (c besser rekonstruiert als a und b) liefern, obwohl beide Ansätze fundamental anders

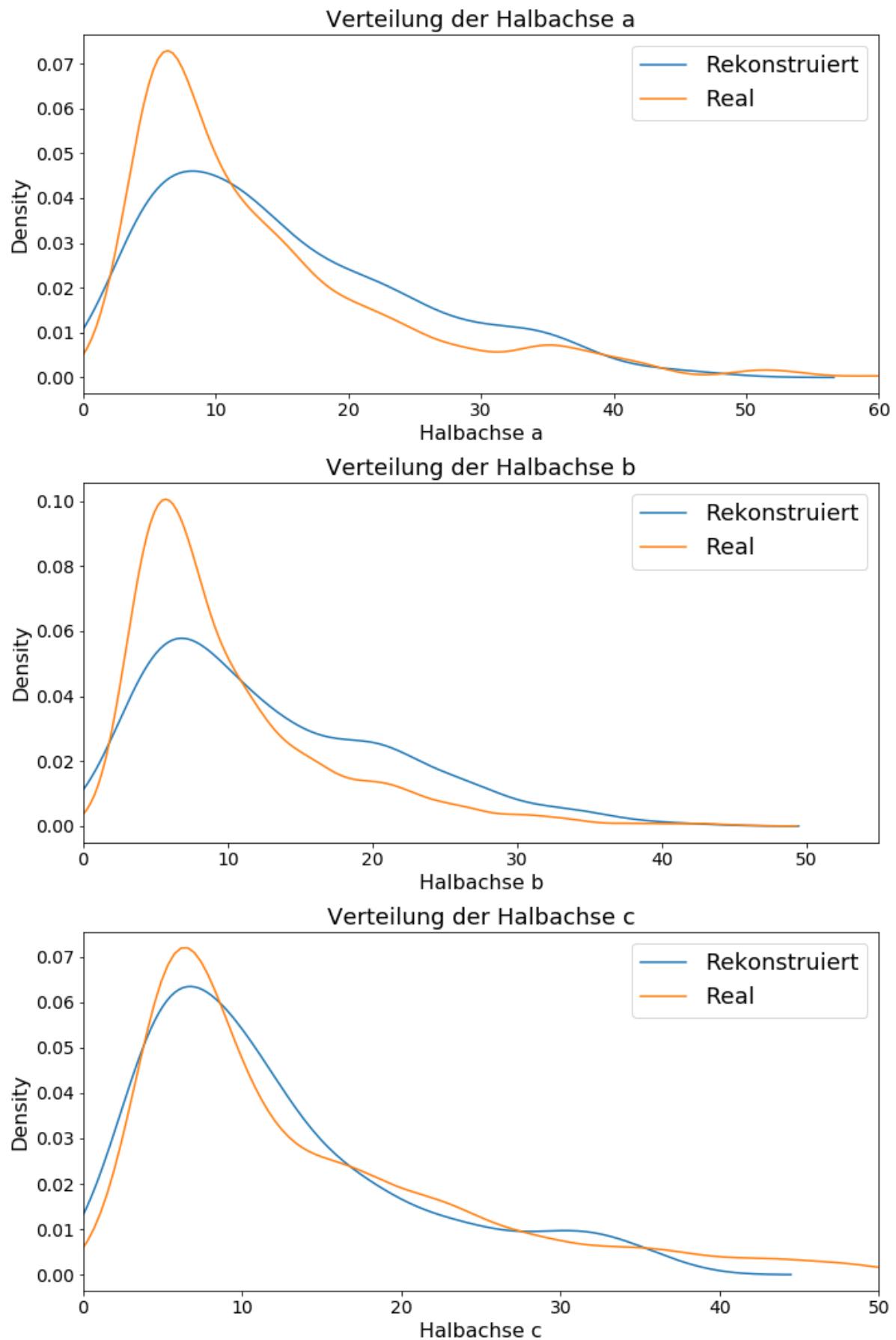


Bild 50: Vergleich Rekonstruierte und reale Halbachsen bei einem Gefüge mit Neigungen mittels Tauschen der Achsen

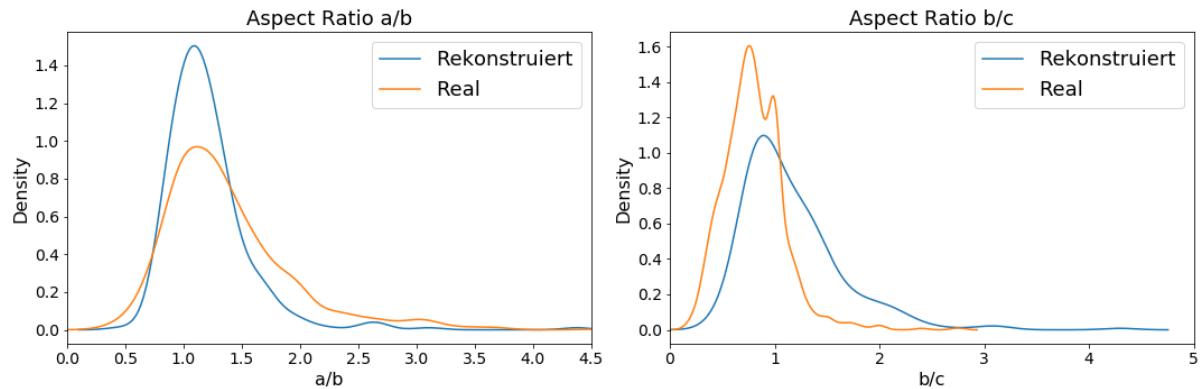


Bild 51: Vergleich Rekonstruierte und reale Aspect Ratios bei einem Gefüge mit Neigungen erzeugt mittels Tauschen der Halbachsen

funktionieren. Ansatz eins funktioniert indirekt über Umrechnung der Achsen in Komponenten, Anwendung des Algorithmus und der anschließenden Umrechnung der Achsenkomponenten zurück in Halbachsen (wie beschrieben). Ansatz zwei funktioniert direkt mit der Verwendung der Halbachsen. Weitere künstlich erzeugte Gefüge könnten dazu genutzt werden, herauszufinden ob die starke Übereinstimmung beider Ansätze in diesem Fall zufällig oder ein grundsätzliches Phänomen ist. Abschließend ist zu sagen, dass beide Ansätze das Gefüge in guter Näherung rekonstruieren können und nicht zweifelsfrei gesagt werden kann, welche der beiden Ansätze besser ist. Hierzu müssten voraussichtlich weitere Studien durchgeführt werden. In beiden Analysen verschlechtern die Neigungen des Gefüges allerdings die Rekonstruktion, besonders deutlich zu sehen bei den deutlich breiteren Verteilungen der Halbachsen und den teilweise fehlerhaft berechneten Aspect Ratios. Gegen ersteres können möglicherweise Verfahren angewandt werden, welche die Verteilung "zusammenstauchen".

Kapitel 6 Schlussfolgerung und Ausblick

Ziel dieser Arbeit war, zum einen ein bereits vorhandenes Verfahren zur Generierung zweidimensionaler Mikrostruktur weiterzuentwickeln und zusätzlich einen Ansatz zu entwickeln, welcher Eigenschaften dreidimensionaler Mikrostruktur aus zweidimensionalen Aufnahmen rekonstruieren kann. Beide Ziele können als erfüllt betrachtet werden.

Mit dem in den Kapitel 3.2.3 vorgestellten Conditional Wasserstein Generative Adversarial Network mit Gradient Penalty steht ein Algorithmus zur Verfügung, welcher zweidimensionale Mikrostruktur verschiedener Materialien mit verschiedenen Parametern abbilden kann. Gleichzeitig wurde ein Algorithmus entwickelt, welcher aus zweidimensionalen EBSD-Aufnahmen dreidimensionale Mikrostruktur in guter Näherung rekonstruieren kann. Dabei liegt der Schluss nahe, dass beide Ansätze auch kombiniert werden können. Dies kann in *zwei* Richtungen durchgeführt werden, zum einen können mittels CWGAN-GP zweidimensionale Daten aus drei Raumrichtungen generiert werden und durch den Algorithmus in rekonstruierte dreidimensionale Mikrostruktur umgewandelt werden. Auch andersherum, d.h. zunächst Anwendung des Algorithmus und direkte Erzeugung dreidimensionaler Körner durch das CWGAN ist möglich. Beide Ansätze sollten problemlos möglich sein, dies wurde aus Mangel von ausreichenden Materialdaten in dieser Arbeit allerdings nicht mehr durchgeführt, da ca. 750 Datenpunkte (Im TD x BN EBSD) zu wenig für das CWGAN sind. Es ist allerdings anzunehmen, dass der Ansatz mit mehr Datenpunkten funktionieren würde. Somit steht mit den Ergebnissen dieser Arbeit und bereits durchgeführten Studien ein integrierter Ansatz bzw. Toolchain der Form EBSD-Aufnahme -> CWGAN -> Rekonstruktionsalgorithmus -> RVE-Erstellung (Oder andersherum) zur Erstellung RVE's mit dreidimensionaler Mikrostruktur zur Verfügung.

Aus den Ergebnissen dieser Arbeit ergeben sich einige weitere möglicherweise interessante Fragestellungen. Fokus sollte zunächst auf der Validierung des Rekonstruktionsalgorithmus mittels realer dreidimensionaler Mikrostruktur liegen, welche im Rahmen dieser Arbeit nicht zur Verfügung stand. Hiermit kann der Algorithmus endgültig validiert werden. Zudem ist noch Verbesserungspotenzial bei der Rekonstruktion geneigter Körner vorhanden, vor allem sollte nach einer besseren Möglichkeit gesucht werden, die Drehungen abzubilden, da hier beiden Verfahren

noch Verbesserungsspielraum haben. Eine Möglichkeit könnte hier darin bestehen, noch zusätzliche Korneigenschaften, welche im Zuge der EBSD-Analyse bestimmt wurden, zu analysieren und ggfl. zu verwenden. Beispielsweise könnten etwa die Missorientierungen zu den Nachbarkörnern nach Mustern untersucht werden (Beispielhaft könnten die Missorientierungen in irgendeiner Form mit bereits verwendeten Parametern, z.B. der Aspect Ratio, korrelieren). Auch die Verteilungen der Halbachsen könnten verbessert werden, da sich gezeigt hat, dass die Verteilungen durch den Algorithmus zu breit geschätzt werden. Ansätze, um die Verteilung hier in irgendeiner Form zusammenzustauen oder zu transformieren, könnten hier helfen. Möglicherweise können zudem EBSD-Aufnahmen aus 45° Winkeln zusätzlich zu den bislang verwendeten 3 Aufnahmen genutzt werden, um die Qualität der Rekonstruktion mit zusätzlichen Achsenkomponenten (45° zum Ursprung) zu verbessern. Auch können handwerkliche Verbesserungen realisiert werden, z.B. kann die Zurückrechnung der Achsenkomponenten in Halbachsen direkt in den Algorithmus integriert werden. Da die Laufzeit des Algorithmus bei steigender Anzahl der Datenpunkte n mit $\mathcal{O}(n^2)$ skaliert, kann der Algorithmus aus Performance-Gründen in einer anderen Sprache geschrieben werden, welche eine höhere Ausführungsgeschwindigkeit als Python aufweist (z.B. Julia, welche eine ähnlich einfache Syntax wie Python und zusätzlich ausführungsgeschwindigkeiten von C und Fortran aufweist).

Auch das CWGAN-GP kann weiter verbessert werden. Zum einen können nicht nur Parameter des realen Gefüges, sondern zusätzlich auch Einschlusssparameter im CWGAN dargestellt werden, gegebenenfalls sogar gleichzeitig. Ebenso können noch weitere Parameter dargestellt werden. Längerfristig kann das CWGAN-GP auch mit Methoden der Bildverarbeitung/Generierung ausgerüstet werden. Da in anderen Studien Methoden der Bilderkennung auf die Martensitbänder des DP800 angewandt wurden, kann im nächsten Schritt das CWGAN-GP dazu genutzt werden, künstliche Martensitbänder für RVE's zu generieren.

Abschließend kann das CWGAN-GP theoretisch auch direkt zum Erzeugen von RVE's aus durch serial-sectioning erzeugten dreidimensionalen Daten genutzt werden. Hierbei werden sehr viele dreidimensionale Datensätze mittels Serial-sectioning erzeugt. Danach wird ein WGAN mit einer Architektur von dreidimensionalen Convolutions genutzt, um diese Mikrostruktur direkt zu erzeugen (RVEGAN).

Kapitel 7 Literaturverzeichnis

- [1] *Electron Backscatter Diffraction (EBSD)*. <https://www.czm.tu-clausthal.de/seal/analytik/electron-backscattered-diffraction-ebsd/>, – Accessed: 2021-01-29
- [2] *GAN Explained*. <https://paperswithcode.com/method/gan>, – Accessed: 2021-01-12
- [3] *Künstliche Intelligenz und Machine Learning für Ihr Unternehmen*. <https://www.weptun.de/ki-und-ml-fuer-ihr-unternehmen/>, – Accessed: 2021-01-29
- [4] ARJOVSKY, M. ; BOTTOU, L. : *Towards Principled Methods for Training Generative Adversarial Networks*. 2017
- [5] ARJOVSKY, M. ; CHINTALA, S. ; BOTTOU, L. : *Wasserstein GAN*. 2017
- [6] BACHMANN, F. ; HIELSCHER, R. ; SCHAEBEN, H. : Grain detection from 2d and 3d EBSD data—Specification of the MTEX algorithm. In: *Ultramicroscopy* 111 (2011), Nr. 12, 1720 - 1733. <http://dx.doi.org/https://doi.org/10.1016/j.ultramic.2011.08.002>. – DOI <https://doi.org/10.1016/j.ultramic.2011.08.002>. – ISSN 0304-3991
- [7] BENGIO, Y. ; LECUN, Y. : Convolutional Networks for Images, Speech, and Time-Series. (1997), 11
- [8] BERBENNI, S. ; FAVIER, V. ; BERVEILLER, M. : Impact of the grain size distribution on the yield stress of heterogeneous materials. In: *International Journal of Plasticity* 23 (2007), Nr. 1, 114 - 142. <http://dx.doi.org/https://doi.org/10.1016/j.ijplas.2006.03.004>. – DOI <https://doi.org/10.1016/j.ijplas.2006.03.004>. – ISSN 0749-6419
- [9] BERGSTRA, J. ; BARDET, R. ; BENGIO, Y. ; KÉGL, B. : Algorithms for Hyper-Parameter Optimization. In: SHAWE-TAYLOR, J. (Hrsg.) ; ZEMEL, R. (Hrsg.) ; BARTLETT, P. (Hrsg.) ; PEREIRA, F. (Hrsg.) ; WEINBERGER, K. Q. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 24, Curran Associates, Inc., 2546–2554
- [10] BERGSTRA, J. ; BENGIO, Y. : Random Search for Hyper-Parameter Optimization. 13 (2012), Febr., S. 281–305. – ISSN 1532–4435

- [11] BLECK, W. : *Werkstoffkunde Stahl für Studium und Praxis*. Bd. 6. Aachen, DE, 2016
- [12] BLECK, W. : *Spezielle Werkstoffkunde der Stähle für Studium und Praxis*. Bd. 1. Aachen, DE, 2017
- [13] BORCHARDT-OTT, W. ; SOWA, H. : *Kristallographie: Eine Einführung für Naturwissenschaftler*. Bd. 8. Göttingen : Springer Spektrum, 2013
- [14] BOTTOU, L. : Large-Scale Machine Learning with Stochastic Gradient Descent. In: LECHEVALLIER, Y. (Hrsg.) ; SAPORTA, G. (Hrsg.): *Proceedings of COMPSTAT'2010*. Heidelberg : Physica-Verlag HD, 2010. – ISBN 978–3–7908–2604–3, S. 177–186
- [15] CARTER ; BARRY, C. : *Transmission electron microscopy: a textbook for material science*. Bd. 2. New York : Springer, 2009
- [16] CRUZADO, A. ; GAN, B. ; JIMÉNEZ, M. ; BARBA, D. ; OSTOLAZA, K. ; LINAZA, A. ; MOLINA-ALDAREGUIA, J. ; LLORCA, J. ; SEGURADO, J. : Multiscale modeling of the mechanical behavior of IN718 superalloy based on micropillar compression and computational homogenization. In: *Acta Materialia* 98 (2015), 242 - 253. <http://dx.doi.org/https://doi.org/10.1016/j.actamat.2015.07.006>. – DOI <https://doi.org/10.1016/j.actamat.2015.07.006>. – ISSN 1359–6454
- [17] CUTURI, M. : Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In: BURGES, C. J. C. (Hrsg.) ; BOTTOU, L. (Hrsg.) ; WELLING, M. (Hrsg.) ; GHAHRAMANI, Z. (Hrsg.) ; WEINBERGER, K. Q. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 26, Curran Associates, Inc., 2292–2300
- [18] DANTZIG, G. : *Lineare Programmierung und Erweiterungen*. Springer Verlag, 1966
- [19] FEYDY, J. ; SÉJOURNÉ, T. ; VIALARD, F.-X. ; AMARI, S.-i. ; TROUVE, A. ; PEYRÉ, G. : Interpolating between Optimal Transport and MMD using Sinkhorn Divergences. In: *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, S. 2681–2690
- [20] FÁTIMA VAZ, M. ; FORTES, M. : Grain size distribution: The lognormal and the gamma distribution functions. In: *Scripta Metallurgica* 22 (1988), Nr. 1, 35 - 40. [http://dx.doi.org/https://doi.org/10.1016/S0036-9748\(88\)80302-8](http://dx.doi.org/https://doi.org/10.1016/S0036-9748(88)80302-8). – DOI [https://doi.org/10.1016/S0036-9748\(88\)80302-8](https://doi.org/10.1016/S0036-9748(88)80302-8). – ISSN 0036–9748
- [21] GANGBO, W. ; MCCANN, R. : The geometry of optimal transport. In: *Acta Mathematica* 177 (1996), Nr. 2, S. 113–161
- [22] GOODFELLOW, I. ; BENGIO, Y. ; COURVILLE, A. : *Deep Learning*. MIT Press, 2016. – <http://www.deeplearningbook.org>

- [23] GOODFELLOW, I. J. ; POUGET-ABADIE, J. ; MIRZA, M. ; XU, B. ; WARDE-FARLEY, D. ; OZAIR, S. ; COURVILLE, A. ; BENGIO, Y. : *Generative Adversarial Networks*. 2014
- [24] GOTTSSTEIN, G. : *Materialwissenschaften und Werkstofftechnik - Physikalische Grundlagen*. Bd. 4. Berlin/Heidelberg : Springer Verlag, 1996
- [25] GROEBER, M. ; GHOSH, S. ; MICHAEL, T. ; UCHICC, D. ; DIMIDUKC, M. ; DENNIS: A framework for automated analysis and simulation of 3D polycrystalline microstructures.: Part 1: Statistical characterization. In: *Acta Materialia* 56 (2008), Nr. 6, S. 1257–1273
- [26] GULRAJANI, I. ; AHMED, F. ; ARJOVSKY, M. ; DUMOULIN, V. ; COURVILLE, A. C.: Improved Training of Wasserstein GANs. In: *CoRR* abs/1704.00028 (2017). <http://arxiv.org/abs/1704.00028>
- [27] HANHLOSER, R. ; SARPESHKAR, R. ; MAHOWALD, M. ; DOUGLAS, R. : Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. In: *Nature* 405 (2000), S. 947
- [28] HE, K. ; ZHANG, X. ; REN, S. ; SUN, J. : Deep Residual Learning for Image Recognition. In: *CoRR* abs/1512.03385 (2015). <http://arxiv.org/abs/1512.03385>
- [29] HENRICH, M. ; PÜTZ, F. ; MÜNSTERMANN, S. : A Novel Approach to Discrete Representative Volume Element Automation and Generation-DRAGen. In: *Materials* 13 (2020), Apr, Nr. 8, 1887. <http://dx.doi.org/10.3390/ma13081887>. – DOI 10.3390/ma13081887. – ISSN 1996–1944
- [30] HOCHREITER, S. ; SCHMIDHUBER, J. : Long Short-term Memory. In: *Neural computation* 9 (1997), 12, S. 1735–80. <http://dx.doi.org/10.1162/neco.1997.9.8.1735>. – DOI 10.1162/neco.1997.9.8.1735
- [31] INDALECIO, G. ; GARCIA LOUREIRO, A. ; SEOANE, N. ; KALNA, K. : Study of Metal-Gate Work-Function Variation Using Voronoi Cells: Comparison of Rayleigh and Gamma Distributions. In: *IEEE Transactions on Electron Devices* 63 (2016), 06, S. 1–4. <http://dx.doi.org/10.1109/TED.2016.2556749>. – DOI 10.1109/TED.2016.2556749
- [32] KANTOROVICH, L. : On the transfer of masses (in Russian). In: *Doklady Akademii Nauk* 37 (1942), Nr. 2, S. 227–229
- [33] KIKUCHI, S. : Diffraction of Cathode rays by mica. In: *Proceedings opf the imperial academy* 4 (1928), S. 354–356
- [34] KINGMA, D. P. ; BA, J. : *Adam: A Method for Stochastic Optimization*. 2014

- [35] KODALI, N. ; ABERNETHY, J. D. ; HAYS, J. ; KIRA, Z. : How to Train Your DRA-GAN. In: *CoRR* abs/1705.07215 (2017). <http://arxiv.org/abs/1705.07215>
- [36] KOLOURI, S. ; PARK, S. ; THORPE, M. ; SLEPCEV, D. ; ROHDE, G. K.: Optimal Mass Transport: Signal processing and machine-learning applications. In: *IEEE Signal Process Mag* 34 (2017), Nr. 4, S. 43–59
- [37] LEWIS, A. ; BINGERT, J. ; ROWENHORST, D. ; GUPTA, A. ; GELTMACHER, A. ; SPANOS, G. : Two- and three-dimensional microstructural characterization of a super-austenitic stainless steel. In: *Materials Science and Engineering: A* 418 (2006), Nr. 1, 11-18. <http://dx.doi.org/https://doi.org/10.1016/j.msea.2005.09.088>. – DOI <https://doi.org/10.1016/j.msea.2005.09.088>. – ISSN 0921-5093
- [38] MEURER, A. ; SMITH, C. P. ; PAPROCKI, M. ; ČERTÍK, O. ; KIRPICHEV, S. B. ; ROCKLIN, M. ; KUMAR, A. ; IVANOV, S. ; MOORE, J. K. ; SINGH, S. ; RATHNAYAKE, T. ; VIG, S. ; GRANGER, B. E. ; MULLER, R. P. ; BONAZZI, F. ; GUPTA, H. ; VATS, S. ; JOHANSSON, F. ; PEDREGOSA, F. ; CURRY, M. J. ; TERREL, A. R. ; ROUČKA, v. ; SABOO, A. ; FERNANDO, I. ; KULAL, S. ; CIMRMAN, R. ; SCOPATZ, A. : SymPy: symbolic computing in Python. In: *PeerJ Computer Science* 3 (2017), Jan., e103. <http://dx.doi.org/10.7717/peerj-cs.103>. – DOI 10.7717/peerj-cs.103. – ISSN 2376-5992
- [39] MINSKY, M. ; PAPERT, S. : *Perceptrons*. Bd. 2. MIT Press, 1988. – <http://www.deeplearningbook.org>
- [40] MIRZA, M. ; OSINDERO, S. : Conditional Generative Adversarial Nets. In: *CoRR* abs/1411.1784 (2014). <http://arxiv.org/abs/1411.1784>
- [41] MONGE, G. : *Mémoire sur la théorie des déblais et des remblais*. Histoire l’Académie Royal des Sciences de Paris, 1781. – S. 666-704
- [42] MROUEH, Y. ; SERCU, T. : Fisher GAN. In: *CoRR* abs/1705.09675 (2017). <http://arxiv.org/abs/1705.09675>
- [43] MÜLLER-BOLLENHAGEN, C. : *Verformungsinduzierte Martensitbildung bei mehrstufiger Umformung und deren Nutzung zur Optimierung der HCF- und VHCF-Eigenschaften von austenitischem Edelstahlblech*, Universität Siegen, Diss., 2011. <https://dspace.ub.uni-siegen.de/handle/ubsi/581>
- [44] NAM, H. ; SHIN, C. : Study of High-k/Metal-Gate Work-Function Variation Using Rayleigh Distribution. In: *IEEE Electron Device Letters* 34 (2013), Nr. 4, S. 532–534. <http://dx.doi.org/10.1109/LED.2013.2247376>. – DOI 10.1109/LED.2013.2247376

- [45] PASZKE, A. ; GROSS, S. ; MASSA, F. ; LERER, A. ; BRADBURY, J. ; CHANAN, G. ; KILLEEN, T. ; LIN, Z. ; GIMELSHEIN, N. ; ANTIGA, L. ; DESMAISON, A. ; KÖPF, A. ; YANG, E. ; DEVITO, Z. ; RAISON, M. ; TEJANI, A. ; CHILAMKURTHY, S. ; STEINER, B. ; FANG, L. ; BAI, J. ; CHINTALA, S. : PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *CoRR* abs/1912.01703 (2019). <http://arxiv.org/abs/1912.01703>
- [46] PELE, O. ; WERMAN, M. : Fast and Robust Earth Mover's Distances, 2009, S. 460 – 467
- [47] PETZKA, H. ; FISCHER, A. ; LUKOVNICOV, D. : *On the regularization of Wasserstein GANs*. 2018
- [48] PEYRE, G. ; CUTURI, M. : Computational Optimal Transport. In: *Foundations and Trends in Machine Learning* 11 (2019), Nr. 5-6, S. 355–607
- [49] PÜTZ, F. ; HENRICH, M. ; FEHLEMANN, N. ; ROTH, A. ; MÜNSTERMANN, S. : Generating Input Data for Microstructure Modelling: A Deep Learning Approach Using Generative Adversarial Networks. In: *mdpi* 13 (2020), S. 4236
- [50] RADFORD, A. ; METZ, L. ; CHINTALA, S. : *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016
- [51] RAMACHANDRAN, P. ; ZOPH, B. ; LE, Q. V.: Searching for Activation Functions. In: *CoRR* abs/1710.05941 (2017). <http://arxiv.org/abs/1710.05941>
- [52] REDMON, J. ; DIVVALA, S. K. ; GIRSHICK, R. B. ; FARHADI, A. : You Only Look Once: Unified, Real-Time Object Detection. In: *CoRR* abs/1506.02640 (2015). <http://arxiv.org/abs/1506.02640>
- [53] ROBBINS, H. ; MONRO, S. : A Stochastic Approximation Method. In: *Ann. Math. Statist.* 22 (1951), 09, Nr. 3, 400–407. <http://dx.doi.org/10.1214/aoms/1177729586>. – DOI 10.1214/aoms/1177729586
- [54] ROSENBLATT, F. : The perceptron: a probabilistic model for information storage and organization in the brain. In: *Psychological Reviews* 65 (1958), S. 386–408
- [55] RUBINSTEIN ; REUVEN, Y. ; KROESE ; DIRK, P. : *The Cross-Entropy Method - A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer Verlag, 2004
- [56] RUMELHART, D. ; HINTON, G. ; WILLIAMS, R. : Learning representations by back-propagating errors. In: *Nature* 323 (1986), S. 533–536. <http://dx.doi.org/https://doi.org/10.1038/323533a0>. – DOI <https://doi.org/10.1038/323533a0>

- [57] SALIMANS, T. ; GOODFELLOW, I. ; ZAREMBA, W. ; CHEUNG, V. ; RADFORD, A. ; CHEN, X. ; CHEN, X. : Improved Techniques for Training GANs. In: LEE, D. (Hrsg.) ; SUGIYAMA, M. (Hrsg.) ; LUXBURG, U. (Hrsg.) ; GUYON, I. (Hrsg.) ; GARNETT, R. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 29, Curran Associates, Inc., 2234–2242
- [58] SAYLOR, D. ; FRIDY, J. ; EL-DASHER, B. ; AL. et.: Statistically representative three-dimensional microstructures based on orthogonal observation sections. In: *Mettall Mater Trans* 35 (2004), S. 1969–1979. <http://dx.doi.org/https://doi.org/10.1007/s11661-004-0146-0>. – DOI <https://doi.org/10.1007/s11661-004-0146-0>
- [59] SCHATT, W. ; WORCH, H. : *Werkstoffwissenschaft*. Stuttgart, 2014
- [60] SCHWARTZ, A. ; KUMAR, M. ; ADAMS, B. ; D.P.FIELD: *Electron Backscatter Diffraction in Material Science*. New York : Springer Science and Business Media, 2009
- [61] SHANNON, C. E.: A mathematical theory of communication. In: *The Bell System Technical Journal* 27 (1948), Nr. 3, S. 379–423. <http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x>. – DOI 10.1002/j.1538-7305.1948.tb01338.x
- [62] SHARMA, H. ; VAN BOHEMEN, S. M. ; PETROV, R. H. ; SIETSMA, J. : Three-dimensional analysis of microstructures in titanium. In: *Acta Materialia* 58 (2010), Nr. 7, 2399-2407. <http://dx.doi.org/https://doi.org/10.1016/j.actamat.2009.12.026>. – DOI <https://doi.org/10.1016/j.actamat.2009.12.026>. – ISSN 1359–6454
- [63] SINKHORN, R. ; KNOPP, P. : Concerning nonnegative matrices and doubly stochastic matrices. In: *Pacific J. Math.* 21 (1967), S. 343–348
- [64] SRIVASTAVA, A. ; VALKOV, L. ; RUSSELL, C. ; GUTMANN, M. U. ; SUTTON, C. : VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In: GUYON, I. (Hrsg.) ; LUXBURG, U. V. (Hrsg.) ; Bengio, S. (Hrsg.) ; WALLACH, H. (Hrsg.) ; FERGUS, R. (Hrsg.) ; VISHWANATHAN, S. (Hrsg.) ; GARNETT, R. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 30, Curran Associates, Inc., 3308–3318
- [65] UEJI, R. ; TSUCHIDA, N. ; TERADA, D. ; TSUJI, N. ; TANAKA, Y. ; TAKEMURA, A. ; KUNISHIGE, K. : Tensile properties and twinning behavior of high manganese austenitic steel with fine-grained structure. In: *Scripta Materialia* 59 (2008), Nr. 9, 963 - 966. <http://dx.doi.org/https://doi.org/10.1016/j.scriptamat.2008.06.050>. – DOI <https://doi.org/10.1016/j.scriptamat.2008.06.050>. – ISSN 1359–6462. – Viewpoint set no. 44 “The materials for MEMS”

- [66] VILLANI, C. : *Optimal Transport - Old and New.* Bd. 338. Heidelberg, DE : Grundlehen der Mathematischen Wissenschaften, 2009
- [67] WASKOM, M. ; TEAM the seaborn d.: *mwaskom/seaborn.* <http://dx.doi.org/10.5281/zenodo.592845>. Version: Sept. 2020
- [68] WRIGHT, S. I. ; ADAMS, B. L.: Automatic analysis of electron backscatter diffraction patterns. In: *Metallurgical Transactions A* 23 (1992), März, Nr. 3, S. 759–767. <http://dx.doi.org/10.1007/BF02675553>. – DOI 10.1007/BF02675553
- [69] ZF ZHANG, Z. W. u. J. E.: What types of grain boundaries can be passed through by persistent slip bands? In: *Journal of materials research* 18 (2003), Nr. 5, S. 1031–1034
- [70] Z.G.WANG, Z. Z.: Comparison of fatigue cracking possibility along large- and low-angle grain boundaries. In: *Materials Science and Engineering* 284 (2000), Nr. 1, S. 285–291
- [71] ZHANG, C. ; ENOMOTO, M. ; SUZAKI, A. : Characterization of three-dimensional grain structure in polycrystalline iron by serial sectioning. In: *Metall Mater Trans* 35 (2004), S. 1927–1933. <http://dx.doi.org/https://doi.org/10.1007/s11661-004-0141-5>. – DOI <https://doi.org/10.1007/s11661-004-0141-5>
- [72] ZHOU, Z. ; LIANG, J. ; SONG, Y. ; YU, L. ; WANG, H. ; ZHANG, W. ; YU, Y. ; ZHANG, Z. : Lipschitz Generative Adversarial Nets. In: *CoRR* abs/1902.05687 (2019). <http://arxiv.org/abs/1902.05687>

Kapitel 8 Anhang

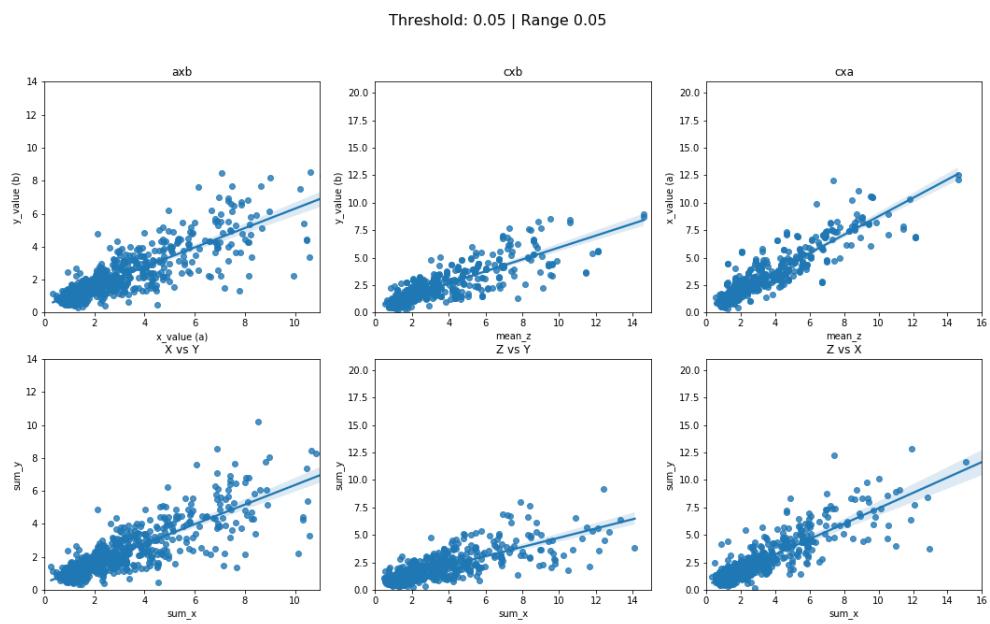


Bild 52: Vergleich der RegPlots für eine Konfiguration mit jeweils 5% und drop=False

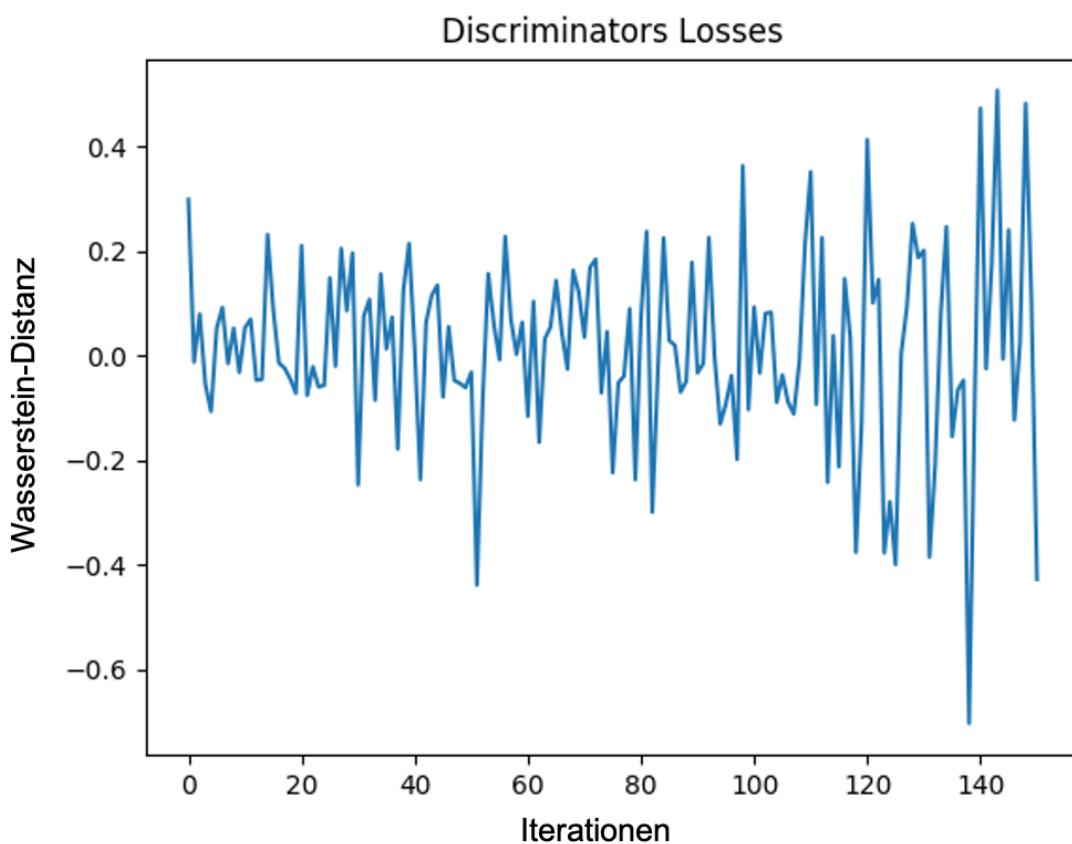


Bild 53: Wasserstein-Loss für eine BatchSize von 32, Tiefe von 2 und Breite von 128