

Hiermit versichere ich, die vorliegende Arbeit allein und nur mit den angegebenen Hilfsmitteln angefertigt zu haben. Der Veröffentlichung der Arbeit in der Bibliothek der Technischen Hochschule Aschaffenburg wird zugestimmt.

Aschaffenburg, den 11. März 2021.

Analyse multimodaler Prozessdaten aus der Additiven Fertigung unter Nutzung von Verfahren des maschinellen Lernens

Masterarbeit von
Aydin Deliktas

11. März 2021

Autor:

Aydin Deliktas

■■■■■■■■■■

■■■■■■■■■■■■■■■■■■■■

Erstprüfer:

Prof. Dr. Michael Möckel

Zweitprüfer:

Prof. Dr. Ralf Hellmann



Technische Hochschule Aschaffenburg
Fakultät Ingenieurwissenschaften
Würzburger Straße 45
D-63743 Aschaffenburg

Abstract

Selective laser melting (SLM) is a 3D printing process which is using a high power-density laser to melt metallic powders. The complex nature of the process is prone to errors and can lead to structural failures in the printed objects. Ongoing success of Convolutional Neural Network (CNN) architectures in image classification makes it possible to automatically classify images in process monitoring and quality assurance. In this work, images from the powder bed of defective and successfully printed SLM projects were labeled patch-wise in three categories: powder, component and defect. These labeled samples were used to train and evaluate a CNN. The proposed model architecture was trained from scratch and achieved an overall F_1 -Score of 85.81 % and a fault detection F_1 -Score of 84.71 %.

Inhaltsverzeichnis

1	Einführung	1
2	Grundlagen	3
2.1	Laserstrahlschmelzverfahren	3
2.2	Fehler im Druckprozess	6
2.2.1	Bauteil- und Pulverauftragfehler	7
2.2.2	Maschinendefekt	9
2.2.3	Deltafehler	10
2.3	Stand der Technik im SLM-Prozessmonitoring	11
2.4	Methodik des Maschinen Lernens	12
2.5	Datenaufbereitung	13
2.6	Over- und Underfitting	15
2.7	Vermeidung von Overfitting	16
2.8	Parameter Optimierung	17
2.9	Modellevaluierung	18
2.9.1	Hold-out Methode	18
2.9.2	Precision, Recall, F1-Score	19
2.9.3	Konfusionsmatrix	21
2.10	Neuronale Netze	22
2.10.1	Kostenfunktion und deren Minimierung	23
2.10.2	Aktivierungsfunktionen	25
2.10.3	Convolutional Neural Network	26
2.10.4	Convolutional Layer	27
2.10.5	Pooling Layer	29
2.10.6	Fully Connected Layer	30
2.10.7	Batch Normalization	31

Inhaltsverzeichnis

2.10.8 Dropout	32
2.10.9 Integrated Gradients	33
3 Praktische Umsetzung	35
3.1 Erfasste Prozessdaten	36
3.2 Aufbereitung des Datensatzes	38
3.3 Erster Trainingsversuch: Overfitting	41
3.4 Schrittweise Optimierung	45
3.4.1 Neuaufteilung des Datensatzes	45
3.4.2 Extrahierung des Pulverbetts	51
3.4.3 Entfernung der ersten Schichtbilder	55
3.4.4 Fazit des bisherigen Trainings	59
3.5 Ansatz der Bildsegmentierung	60
3.5.1 Segmentweise Datenaufbereitung	61
3.5.2 Training mit Bildsegmenten	67
3.5.3 Fazit des Trainings auf Segmenten	75
3.6 Zeitliche Betrachtung der Klassifizierungen	76
4 Fazit und Ausblick	83
Literaturverzeichnis	85
Abbildungsverzeichnis	93
Tabellenverzeichnis	97
Abkürzungsverzeichnis	99
Anhang	103
A Ergebnisse des Testdatensatzes für die einzelnen Modelle	104

1 Einführung

Die Additive Fertigung (engl. Additive Manufacturing, kurz AM) breitet sich seit Jahrzehnten in immer mehr Industriezweigen aus. Beispiele hierfür sind die Zahnmedizin, Architektur oder auch die Automobilindustrie. Solche Verfahren, welche auch unter dem Synonym 3D-Druck zusammengefasst werden, ermöglichen dabei die schnelle Herstellung von komplexen Strukturen und eignen sich deshalb auch für die Erprobung von Prototypen oder der Optimierung von Bauteilen. Beispiele sind etwa in der Automobilindustrie zu finden, in der Vorserienbauteile mittels 3D-Druck hergestellt und anschließend weitergehend optimiert werden können. Auch in der Zahnmedizin können durch die additive Fertigung genau auf den Patienten abgestimmte Zahnimplantate erstellt werden [1, 2, 3]. Das AM bedient sich dafür einem anderen Grundprinzip als konventionell abtragende Herstellungsverfahren, welche durch Entfernen von Material, Strukturen entstehen lassen. Beispiele hierfür sind etwa Fräsmaschinen oder das Hochdruckwasserstrahlschneiden. Diese abtragenden Verfahren haben meist einen hohen Materialverbrauch und benötigen viel Energie, was wiederum weiter zu Lasten der Effizienz geht.

Bei der Additiven Fertigung hingegen wird die gewünschte Geometrie aus einzelnen Schichten nacheinander aufgebaut. Durch den Schichtweisen Aufbau ist es möglich komplexere und auch sehr unterschiedliche Geometrien mit einer einzelnen Maschine herzustellen. [4] Besonders vielversprechend unter den Additiven Fertigungsverfahren ist das sogenannte selektive Laserstrahlschmelzen (engl. Selective Laser Melting, kurz SLM), wobei viele Untersuchungen gezeigt haben, dass eine Vielzahl von Werkstoffen, wie z.B. Edelstahl, Titanlegierungen oder Keramik verarbeitet werden können [5, 6]. Und auch wenn das SLM-Verfahren in den letzten Jahren verstärkt in den Fokus der Forschung gerückt ist, ist dieses nach wie vor sehr fehleranfällig, was einen produktiven Einsatz bisher noch verhindert. [5, 7]

1 Einführung

Parallel dazu haben maschinelle Lernverfahren in den letzten Jahren enorme Fortschritte gemacht, welche unseren Alltag immer mehr durchsetzen. Von Spurhalteassistentensystem moderner Autos, über Vorschläge für optimale Aufnahmen bei der Fotografie mittels eines Smartphones oder der Vorhersage des potenziell nächsten Wortes beim Schreiben von Nachrichten. Diese Entwicklung hat auch dazu geführt, dass versucht wird mit solchen maschinellen Lernalgorithmen bisher mühselig und von Hand durchgeführte Tätigkeiten zu automatisieren. Andere Verfahren versuchen wiederum in sehr umfangreichen Daten, Korrelationen zu finden, und aus diesen Zusammenhängen Schlussfolgerungen zu ziehen [8, 9]. Jedoch gibt es nach wie vor Bereiche, in denen eine solche Automatisierung nicht ohne weiteres möglich ist. Entweder weil die Ergebnisse der Machine Learning Anwendung noch nicht gut genug sind oder weil die Problemstellung, wie etwa beim Laserstrahlschmelzen, zu komplex ist. Deshalb soll im Rahmen dieser Masterarbeit mit Hilfe von Verfahren des maschinellen Lernens untersucht werden, inwieweit sich die unterschiedlichen Fehler des SLM-Verfahrens frühzeitig noch während des Drucks erkennen lassen.^{1,2}

¹vgl. Projektphasenbericht I [7]

²vgl. Projektphasenbericht II [10]

2 Grundlagen

Die erste Hälfte des nachfolgenden Kapitels beschäftigt sich mit den theoretischen Grundlagen des Laserstrahlschmelzens und des 3D-Druckprozesses. Darauf aufbauend werden die häufigsten Druckfehler sowie deren Ursachen näher betrachtet und erläutert. In der zweiten Hälfte geht es anschließend um Maschinen Lernverfahren und deren zugrunde liegende Theorien, welche im Kapitel Praktische Umsetzung eingesetzt werden.

2.1 Laserstrahlschmelzverfahren

Das SLM-Verfahren gehört zu den pulverbettbasierenden additiven Fertigungsverfahren und ist mit dem Laserstrahlschweißen vergleichbar, da bei beiden Verfahren eine genau definierte Wärmequelle das zugrunde liegende Material zum Schmelzen bringt. Die große Ähnlichkeit der beiden Methoden bei der Verarbeitung von Metallen erlauben ein Übertragen der Erkenntnisse aus dem einem Bereich in den jeweils anderen. [11, 12] Abbildung 2.1 stellt den prinzipielle Aufbau einer Maschine zum Laserstrahlschmelzen dar.

2 Grundlagen

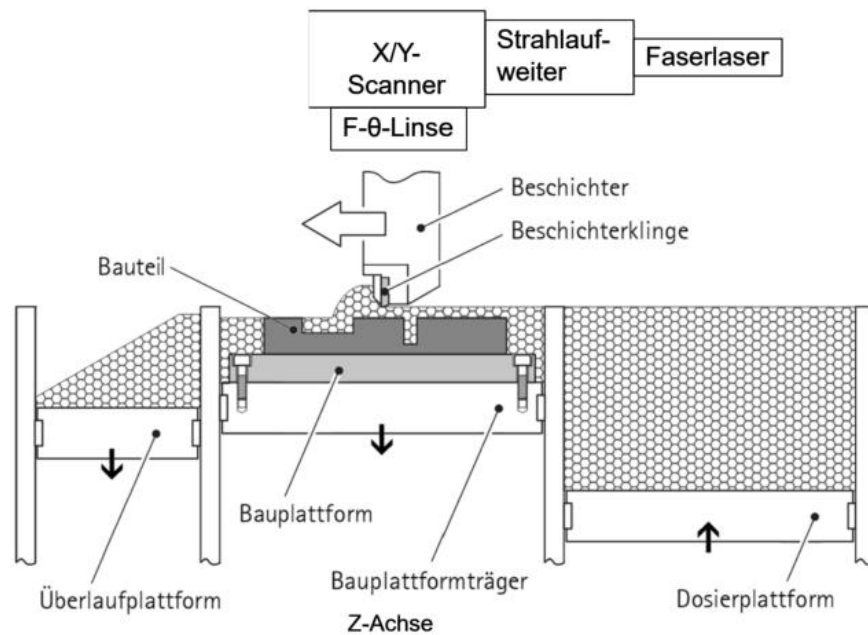


Abbildung 2.1: Prinzipieller Aufbau einer Laserstrahlschmelzmaschine (Quelle: [13])

Beim Laserstrahlschmelzverfahren wird das gewünschte Bauteil aus einzelnen Ebenen oder auch Schichten nacheinander aufgebaut. Dazu wird zunächst das digitale 3D Modell des Bauteils entlang der Z-Achse in 20-100 μm dicke Schichten (engl. Slices) aufgeteilt [14]. Zusätzlich wird das Modell weiter aufbereitet, indem etwa zur Stabilisierung notwendige Stützstrukturen berechnet und dem Modell hinzugefügt werden. Das so aufbereitete 3D-Modell wird anschließend an den Drucker übertragen um den eigentlichen Druckprozess zu starten. Zunächst wird das Rohmaterial, hier das metallene Pulver, mittels eines Beschichters auf eine horizontale und ebene Plattform aufgetragen. Ein Laserstrahl schmilzt das gleichmäßig verteilte Pulver lokal und verschweißt es mit dem darunter liegenden Substrat. Der Laser wird aus der Horizontalen mittels eines Galvanometer-Scannspiegels in die Vertikale abgelenkt. Mit Hilfe einer F-Theta-Linse und eines X/Y-Scanners kann der Laserstrahl auf die komplette Bauplattform fokussiert werden. Die horizontale Bauteilebene setzt sich aus sogenannten Schweißraupen zusammen, welche sich sowohl in der horizontalen als auch vertikalen überlappen (Abbildung 2.2). Um die Stabilität des Bauteils weiter zu verbessern, wird zum Beispiel die Laufrichtung der Schweißraupen zwischen zwei Ebenen um 90° gedreht. [7, 13, 14]¹

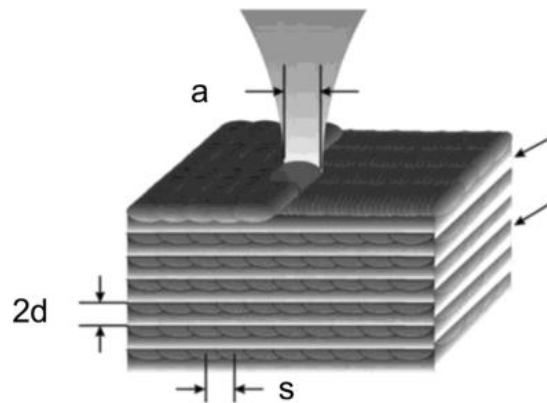


Abbildung 2.2: Schema der Schweißraupen mit a: Raupenbreite, d: Schichtstärke und s: Abstand zwischen zwei Schweißraupen [13]

Nachdem eine Schicht komplett bearbeitet wurde, wird die Bauplattform um eine Schichtstärke entlang der Z-Achse abgelassen. Anschließend wird wieder durch den Beschichter das Pulver aufgetragen und der ganze Prozess wiederholt sich so lange, bis das Bauobjekt fertiggestellt wurde. Das Bauteil wird oft mit zusätzlichen Supportstrukturen fest mit der Bauplattform verschweißt und nach Fertigstellung mit einer Säge wieder von dieser getrennt. Supportstrukturen dienen zum einen dem Abtransport der Wärme, welche durch das Schmelzen dem Bauteil zugeführt wird. Zum anderen sollen diese Strukturen das Bauteil stabilisieren, da durch das lokale Schmelzen des Pulvers Spannungen im Bauteil entstehen, welche vor allem in horizontalen oder überhängenden Abschnitten Wölbungen oder Krümmungen hervorrufen können. Eine weitere Methode um Spannungen im Bauteil zu vermeiden ist das Vorheizen des Pulverbetts und der Bauplatte auf etwa 200-500 °C.

Um äußere Störeinflüsse der Umgebung zu verhindern findet der Druckprozess innerhalb einer geschlossenen Kammer statt, welche mit einem Schutzgas gefüllt ist. Diese Schutzgasatmosphäre soll chemische Reaktionen zwischen dem erhitzten Metall und der Umgebung verhindern. Der Bauraum ist hierzu meist mit einem Edelgas wie etwa Argon oder Stickstoff gefüllt, welches das Bauobjekt umströmt. Die Strömung sorgt dafür, dass Schweißspritzer und Dämpfe abtransportiert und somit Verunreinigungen im Baujob verhindert werden. [7, 13, 14]¹

2.2 Fehler im Druckprozess

Anhand der Beschreibung des SLM Prozesses im vorherigen Kapitel wurde bereits ersichtlich, dass der Schichtenweise 3D-Druck ein komplexes Unterfangen ist. Um Fehler während des Drucks zu erkennen und diesen in Zukunft auch entgegenwirken zu können, ist eine umfangreiche Überwachung des Drucks notwendig. Der SLM Prozess und somit die Qualität des gedruckten Objekts, hängt von über 50 unterschiedlichen Parametern ab [15]. Unter diesen Parametern befinden sich auch unterschiedliche Sensoren innerhalb sowie außerhalb der Prozesskammer. Hier wird somit schon ersichtlich, dass es abzuschätzen gilt, welche Parameter im Druckprozess für eine zuverlässige Fehlerdetektion von Bedeutung sind und welche vernachlässigt werden können. Grundsätzlich lassen sich die Parameter in zwei Gruppen einteilen, die vordefinierten und die kontrollierbaren Prozessparameter. Zu den vordefinierten Parametern zählen beispielsweise die Geometrie des zu druckenden Objekts oder das verwendete Materialpulver. Kontrollierbare Parameter dagegen sind beispielsweise die Eigenschaften des verwendeten Lasers, die Schichtdicke der einzelnen Ebenen oder auch die Temperatur. Diese lassen sich, anders als die vordefinierten Parameter, während des eigentlichen Druckprozesses verändern. Aus den kontrollierbaren Parametern lassen sich außerdem noch weitere Informationen zum Druck ableiten mit deren Hilfe man auf Eigenschaften wie der Dichte im gedruckten Objekt oder auf dessen Porosität schließen kann. Diese kontrollierbaren Prozessvariablen bestimmen somit maßgeblich die Qualität des gedruckten Objekts. [16] Die soeben beschriebenen Prozessparameter lassen sich durch das periodische Speichern der vorhandenen Sensoren protokollieren und auswerten. [7]¹

Durch die Beobachtung des Pulverbetts lässt sich zudem ein Einblick in den momentanen Zustand des Druckobjekts erhalten. Auf diese Weise lassen sich etwa Fehler in der Geometrie oder Erhöhungen einzelner Abschnitte aufgrund von Spannungen erkennen. Aber auch Defekte in den Supportstrukturen oder Probleme durch ungleichmäßig verteiltes Pulver lassen sich so ausmachen [15]. Diese Arbeit beschränkt sich auf die Detektion von Fehlern, welche rein optisch erkannt werden können. Um jedoch diese Fehler überhaupt detektieren zu können, ist es nötig deren Merkmale und Ursachen zu erkennen. Die im Druckprozess auftretenden Fehler lassen sich nach ihren Merkmalen unterscheiden. [7]¹

2.2.1 Bauteil- und Pulverauftragfehler

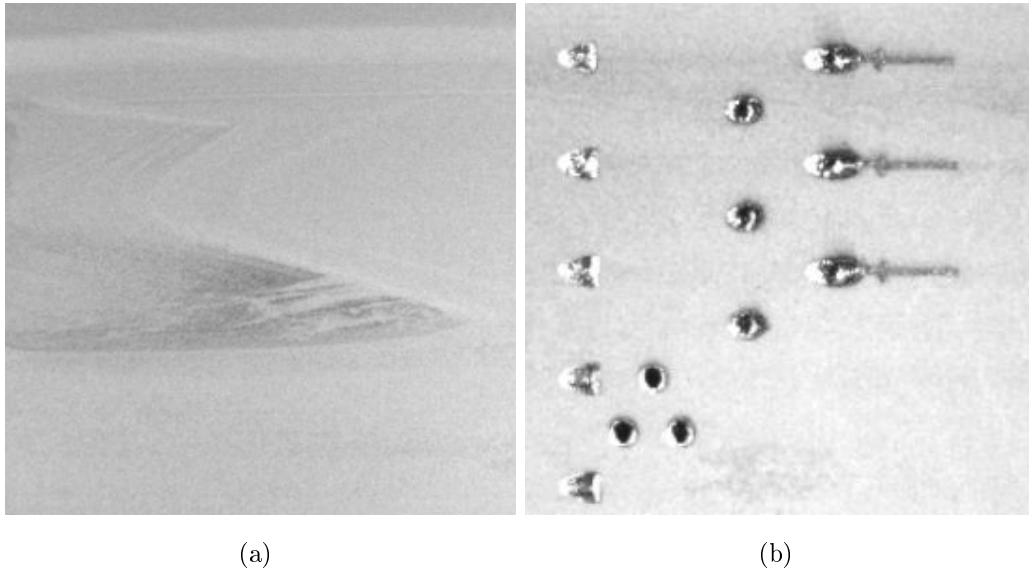
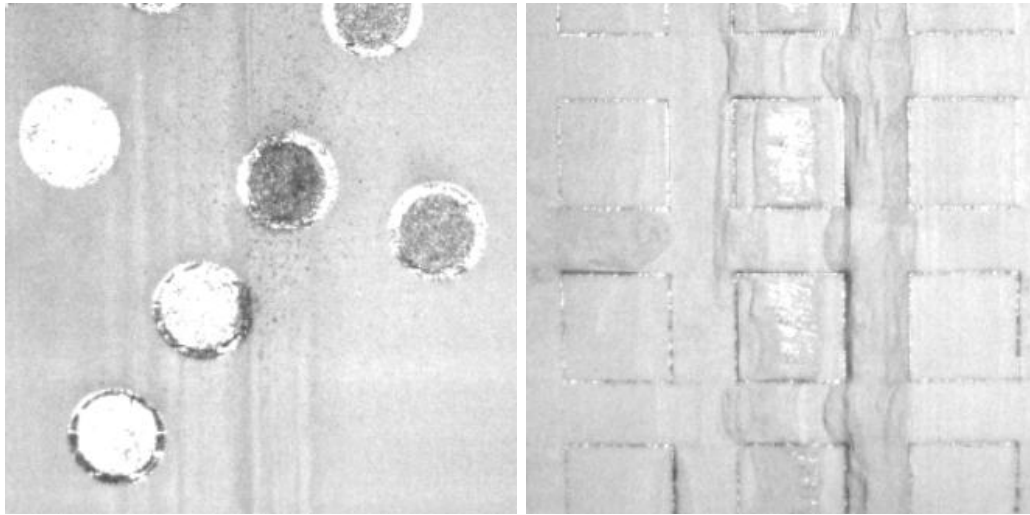


Abbildung 2.3: Das Pulver in (a) ist ungleichmäßig verteilt, es sind schlieren zu sehen. Die Bauteile in (b) wurden nicht vollständig mit Pulver bedeckt und deshalb durch den Laserstrahl verbrannt, was sich in den schwarzen Flecken auf den metallischen Bauteilen äußert.

Am Beschichter befindet sich eine Förderschnecke, mit derer das Metallpulver durch eine sogenannte Beschichterklänge verteilt wird. In einigen Fällen kann es jedoch passieren, dass das Pulver nicht die komplette Fläche des Druckraums bedeckt, etwa wenn im Pulver Reservoir nicht genug Material vorhanden ist. [17] Dies wird daran sichtbar, dass sich im Pulverbett Schlieren bilden (Abbildung 2.3(a)). Erreichen diese Schlieren nun Teile eines Bauteils so wird das Bauteil dadurch nicht vollständig mit Pulver beschichtet. Der Laser, welcher darauf folgend eigentlich das Pulver zum Schmelzen bringen soll, schmilzt dadurch lediglich die darunter liegende schon fertiggestellte Schicht an. Dadurch entstehen dunkle Flecken am Bauteil, welche Verbrennungen ähnlich sehen (Abbildung 2.3(b)). Auch wenn die anschließende Schicht wieder vollständig mit Pulver bedeckt wird, besteht nun das Problem, dass die aufeinander folgenden Schichten potentiell nicht richtig miteinander verbunden sind und es somit zu einer Schwächung der Struktur des Bauteils kommt.

2 Grundlagen

Solche Fehler, die lediglich das Pulverbett betreffen lassen sich als Pulverfehler zusammenfassen, da diese in ihrer Form und Erscheinung variieren können. Andererseits können Fehler, welche das Bauteil direkt betreffen übergeordnet als Bauteilfehler bezeichnet werden. [7]¹



(a)

(b)

Abbildung 2.4: In (a) und (b) sind im Pulverbett Wellen entstanden da die Beschichterklinge kurzzeitig an den Bauteilen hängen geblieben ist. Beim anschließenden Lösen von diesen hat die Klinge einen Sprung gemacht, der das Pulver wiederrum verwirbelt hat.

Ein nicht vollständig mit Pulver bedecktes Bauteil kann allerdings auch weitere Fehler verursachen. Durch die zugeführte Wärme des Lasers und das erneute aufschmelzen des teilweise freiliegenden Bauteils kann es passieren, dass sich dieses leicht verzieht. Wie bereits im Kapitel Laserstrahlschmelzverfahren beschrieben, wird die Bauplattform nach dem Fertigstellen einer Schicht um deren Höhe abgesenkt, sodass beim beschichten eine neue Ebene mit der gewünschten Pulverstärke entsteht. Aufgrund dessen bewegt sich die Beschichterklinge nur wenige Mikrometer über dem Bauteil hinweg. Wenn das Bauteil jedoch, z.B. weil dieses sich etwas verzogen hat, leicht aus dem Pulverbett herausragt, kann es dazu kommen, dass die Beschichterklinge am Bauteil schabt oder auch kurzzeitig hängen bleibt.

Durch anschließendes Lösen vom Bauteil macht diese einen Sprung und zerstreut somit das Pulver, wie in Abbildung 2.4(a) sichtbar wird. Solche Wellen im Pulverbett können allerdings auch durch eine ungünstige Platzierung der Bauteile entstehen. Dies ist vor allem dann der Fall, wenn die Bauteile parallel zur Beschichterklänge platziert sind und somit eine große Fläche gegenüber der Klinge bilden (Abbildung 2.4(b)). [7, 17]¹

2.2.2 Maschinendefekt

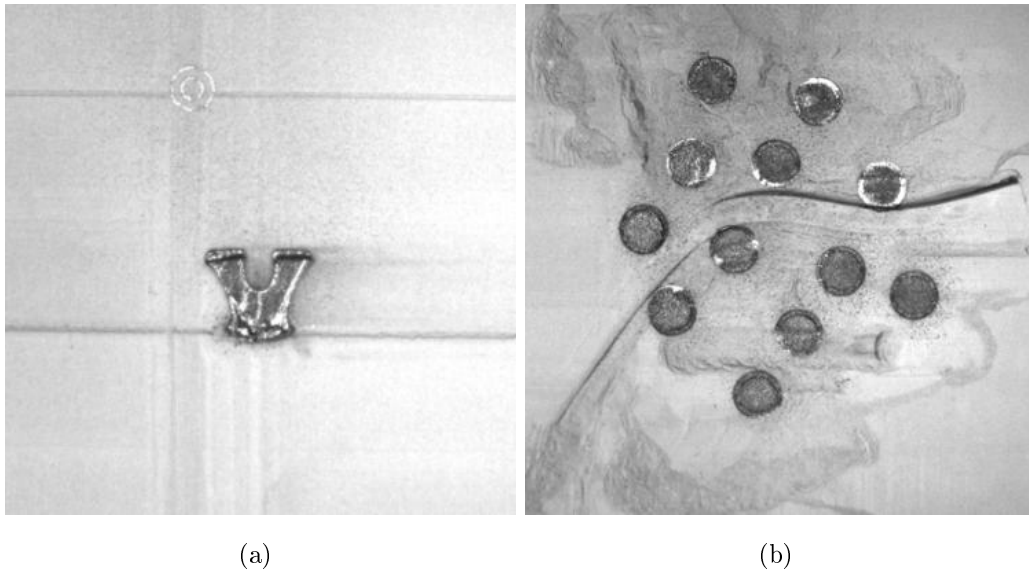


Abbildung 2.5: Horizontale Streifen im Pulverbett durch Beschädigung des Beschichters (a) und vollständig zerstörtes Pulverbett durch abgefallene Beschichterklänge (b).

Ein wiederholtes Schaben des Beschichters am Bauteil kann diesen ebenfalls beschädigen, was sich in horizontalen Streifen im Pulverbett äußert (Abbildung 2.5(a)). In einer Extremform des Maschinenfehlers kann sogar die ganze Beschichterklänge aus ihrer Halterung gerissen werden (Abbildung 2.5(b)). [7, 17]¹

2.2.3 Deltafehler

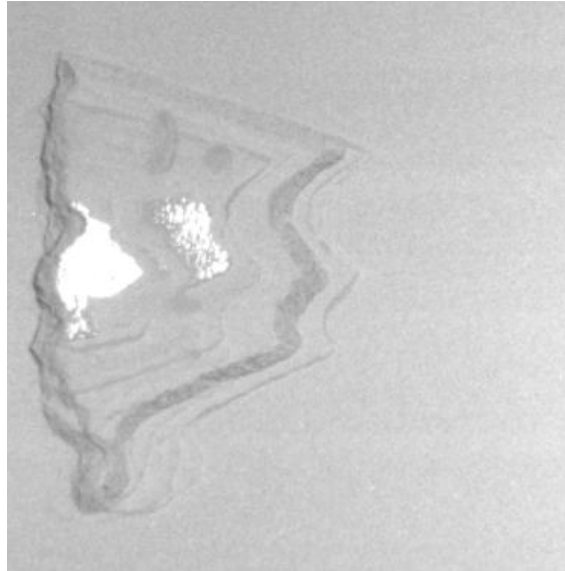


Abbildung 2.6: Typischer Aufbau eines Deltafehlers

Deltafehler können auch als Pulverfehler betrachtet werden. Jedoch haben diese sehr eindeutige Merkmale, welche sie von den restlichen Fehlern unterscheiden. Zum einen tritt der Deltafehler nur auf der linken Seite des Pulverbetts auf. Dies liegt daran, dass das Pulver durch die Förderschnecke im Beschichter ungleichmäßig über dem Pulverbett verteilt wird. Zum anderen besitzt dieser Fehler sehr scharfe Kanten (Abbildung 2.6), welche ihn vom restlichen Pulverbett stark abgrenzen. Andere Pulverfehler wie etwa Schlieren (Abbildung 2.3(a)) besitzen dagegen eher weiche Übergänge zum Pulverbett.[7]

Die hier erwähnten Fehler müssen dabei nicht exklusiv auftreten, sondern treten häufig in Kombination zueinander auf. Auch gehen diese Fehler oft fließend ineinander über oder entstehen erst langsam im Verlauf des Druckprozesses.[7]¹

2.3 Stand der Technik im SLM-Prozessmonitoring

Die Idee, den Prozess des Selektiven Laserstrahlschmelzens zur Prozesszeit zu beobachten ist dabei nicht neu. Schon 2008 haben Bayle und Doubenskaia in [18] versucht, mittels einer Infrarotkamera und eines Pyrometers Aussagen über die Qualität des Drucks zu tätigen. Hierzu wurde der Laserstrahl und das direkt geschmolzene Pulver betrachtet um so beurteilen zu können, ob es zu einer konsistenten Verbindung zwischen den einzelnen Schichten gekommen ist oder etwaige Lücken entstanden sind. Grasso et al. [19] verwendeten für das Monitoring hingegen eine Highspeed Kamera. Die Bildaufnahmen wurden hier unter der Verwendung von statistischen Verfahren wie der Hauptkomponentenanalyse analysiert. Ziel war dabei, unterschiedliches Verhalten des Schmelzpools innerhalb einer einzelnen Schicht zu erkennen. Dagegen haben Angelone et al. [20] versucht mittels Convolutional Neural Network fehlerhafte Strukturen direkt am gedruckten Bauteil zu erkennen. Gemeinsam haben diese Verfahren, dass sie direkt das zu druckende Bauteil betreffen. Um ein solches Monitoring umzusetzen, sind jedoch sehr umfangreiche Daten wie hochauflösende Bilder notwendig. Das hier entwickelte Verfahren soll sich stattdessen vermehrt auf die Umgebung des zu druckenden Objekts beziehen und Unregelmäßigkeiten im Pulverbett erkennen, welche einen Einfluss auf das Bauteil nehmen könnten. Somit kann diese Methodik eher als komplementäres Unterstützungssystem für ein Monitoring auf Bauteilebene betrachtet werden, welches dazu Verfahren des maschinellen Lernens einsetzt.

2.4 Methodik des Maschinen Lernens

Unter Machine Learning versteht man die künstliche Erzeugung von Wissen aus bereits bekannten Daten. Computer sollen Gesetzmäßigkeiten und Zusammenhänge aus einem vorliegenden Datenbestand ableiten und dieses Wissen anschließend auf neue noch unbekannte Daten anwenden können. Der Datenbestand stellt somit eine Situation dar, welche durch eine passende Modellbildung repräsentativ abgebildet werden soll.

Als Beispiel sei hier ein Filter genannt, der Spam E-Mails erkennen soll. Der klassische Programmieransatz hierfür wäre nun, für die einzelnen Arten von Spam E-Mails Regeln zu definieren. Dieser Filter würde schnell sehr komplex und müsste auch häufig aktualisiert werden, da schon eine kleine Änderung innerhalb der unerwünschten E-Mail dazu führen würde, dass die E-Mail nicht mehr als Spam erkannt wird. Ein Spam Filter, welcher auf Machine Learning Verfahren basiert, würde kontinuierlich auf neue Spam E-Mails angepasst und müsste dadurch nicht händisch geändert werden.

Gleichzeitig ist maschinelles Lernen kein Allheilmittel, da es zum einen viele unterschiedliche Methoden gibt und zum anderen keine universelle Lösung existiert, welches auf jede Problemstellung angewendet werden kann. Deshalb ist die Auswahl eines zur Aufgabenstellung passenden Verfahrens notwendig. Unterscheiden lassen sich Maschinen Lernverfahren daran, wie stark diese während der Lernphase kontrolliert bzw. unterstützt werden müssen. Grundsätzlich gibt es das überwachte Maschinen Lernen (engl. Supervised Learning) und unüberwachte Maschinen Lernen (engl. Unsupervised Learning). Gegenstand dieser Arbeit ist das Supervised Learning, weshalb an dieser Stelle das unüberwachte Maschinenlernen nur kurz angesprochen wird. [10, 21]²

Beim überwachten Machine Learning wird dem zu trainierenden Algorithmus neben den eigentlichen Daten auch das jeweils zugehörige Ergebnis mitgeteilt. Im oben genannten Beispiel von Spam E-Mails also eine Definition, ob es sich bei der jeweiligen E-Mail um Spam oder eine legitime E-Mail handelt. Diese Kategorisierung von beispielsweise E-Mails wird Klassifizierung genannt und ist ein häufiger Einsatzzweck von überwachten ML Verfahren. Ein weiteres typisches Einsatzgebiet des Supervised Machine Learnings ist die sogenannte Regression, bei der es darum geht aus einer Reihe von Datenpunkten zukünftige Datenpunkte abzuleiten. Beispiel hierfür ist etwa die Vorhersage von Aktienkursen oder die Prädiktion des nächsten Wortes in der Wortvervollständigung eines Smartphones.

Dagegen liegt beim unüberwachten Lernen oft eine unbekannte Datenstruktur vor und das Maschinen Lernverfahren soll versuchen, in dieser undurchsichtigen Struktur Überschneidungen und eventuelle Zusammenhänge zu finden. Unter anderem sei hier das sogenannte Clustering genannt, welches versucht unstrukturierte Daten in einzelne Untergruppen aufzuteilen. Somit kommt das Clustering einer Klassifizierung recht nahe. Unabhängig vom verwendeten Verfahren, ist es nötig die Daten vor der Nutzung aufzubereiten. [10, 21]²

2.5 Datenaufbereitung

Datensätze, welche für das Training verwendet werden, stammen nicht immer aus einer einzigen Quelle. Im Fall von Bilddaten stammen diese etwa von unterschiedlichen Geräten oder wurden unter verschiedenen Lichtbedingungen aufgenommen. Diese Unterschiede sind zwar einerseits erwünscht und vorteilhaft, da das Modell auf diese Weise auch erlernen kann, Merkmale aus unterschiedlichen Situationen zu extrahieren. Andererseits erhöhen diese Differenzen die Streuung des Datensatzes. Eine zu große Streuung kann dazu führen, dass der Maschine Learning Algorithmus nur langsam lernt oder bessere Ergebnisse auch nach längerem Training nicht erreicht werden. Deshalb wird der Datensatz üblicherweise standardisiert und normalisiert, d.h. der Datensatz wird zum einen zentriert und anschließend auf kleine Wertebereiche herunter skaliert.

2 Grundlagen

Dies erfolgt unter anderem durch Subtraktion des Mittelwertes und anschließender Division durch die Standardabweichung. Auf diese Weise werden die Distanzen zwischen den einzelnen Datenpunkten kleiner, während der Grundlegende Zusammenhang zwischen den einzelnen Datenpunkten beibehalten wird (Abbildung 2.7). Diese Standardisierung und Normalisierung kann also maßgeblich darüber entscheiden, wie gut das Machine Learning Modell letztendlich funktioniert. [10, 22]²

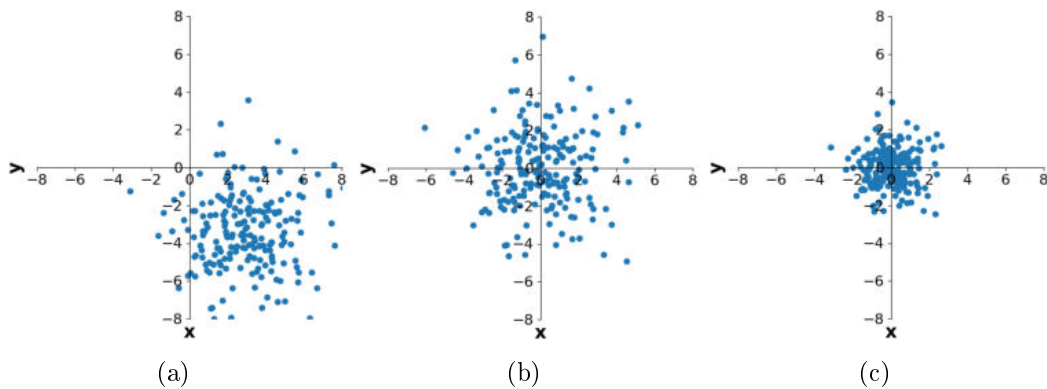


Abbildung 2.7: Die original Datenpunkte in (a) werden durch Subtraktion des Mittelwerts auf den Ursprung verschoben (b). Durch anschließende Division durch die Standardabweichung werden diese weiter zentriert, ohne die Beziehung zwischen den einzelnen Punkten zu verändert (c).

2.6 Over- und Underfitting

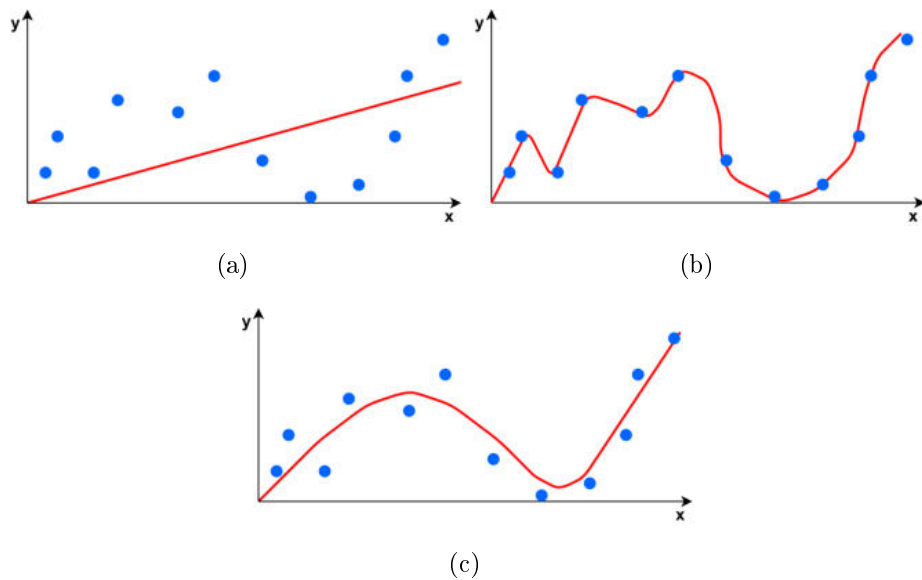


Abbildung 2.8: Bei einer zu geringen Modellkomplexität kann das Modell den Datensatz nicht komplett abbilden (a). Andererseits kann das Modell bei einer zu hohen Komplexität den Datensatz komplett erfassen und sogar auswendig lernen (b). Ein optimales Modell dagegen ist komplex genug um die für die Aufgabenstellung wichtigen Merkmale zu erlernen, ohne den Trainingsdatensatz auswendig zu können (c).

Das Ziel bei der Modellbildung liegt somit darin, möglichst genaue Vorhersagen auf ungesehenen Daten zu erhalten, gleichzeitig soll das Modell jedoch auch möglichst simpel bleiben. Wenn der CNN Algorithmus die erlernten Merkmale aus einem Datensatz, den sogenannten Trainingsdaten, auf einen anderen noch unbekannten, sogenannten Testdaten, übertragen kann, spricht man von einer Generalisierung. Dazu wird zunächst das Modell so aufgebaut, dass es auf dem Trainingsdatensatz zufriedenstellende Ergebnisse liefert. Wenn sowohl der Trainings- als auch der Testdatensatz repräsentativ für die Aufgabenstellung sind, sollte das Modell durch gute Generalisierung im Optimalfall auch gute Ergebnisse auf dem Testdatensatz liefern. Praktisch kommt es hier zunächst jedoch oft zu einem sogenannten Over- oder Underfitting.

2 Grundlagen

Wenn das Modell zu einfach gehalten ist, kann es die für die Aufgabenstellung notwendigen Merkmale nicht komplett erfassen. Verdeutlicht wird dies dadurch, dass die erreichten Ergebnisse auf dem Trainings- und Testdatensatz nur schlechte Werte erreichen. In diesem Fall spricht man von einem sogenannten Underfitting. In der Regel wird als Reaktion darauf die Architektur des CNN erweitert, wodurch dessen Komplexität ansteigt. Ein komplexeres Modell erlaubt es mehr Merkmale zu erfassen, was sich u.a. darin widerspiegelt, dass sich die Ergebnisse auf dem Trainingsdatensatz verbessern. Die Kehrseite eines komplexeren Algorithmus ist jedoch, dass dieser zu einer stärkeren Überanpassung (engl. Overfitting) neigt. Dabei beginnt der Algorithmus die Trainingsdaten auswendig zu lernen, was sich in sehr guten Ergebnissen auf den Trainingsdaten widerspiegelt. Gleichzeitig liefert das Modell bei den unbekannten Testdaten jedoch sehr viel schlechtere Ergebnisse. Ein weiterer Nachteil eines komplexen Modells liegt darin, dass dieses beginnt Muster zu erkennen, welche eigentlich so nicht existieren. [10, 23]²

2.7 Vermeidung von Overfitting

Overfitting lässt sich mit verschiedenen Verfahren vermeiden oder eindämmen. Zunächst sollte versucht werden, den Umfang des Trainingsdatensatzes zu erweitern. Dies kann beispielsweise durch Hinzufügen von realen Daten geschehen. Dies ist jedoch nicht immer möglich, weshalb man oft auf eine sogenannte Datenaugmentierung zurückgreift. Bei der Datenaugmentierung werden zusätzliche Trainingsdaten auf künstliche Weise erzeugt. Bei Bildern lassen sich zusätzliche Beispieldaten durch Spiegelungen, oder falls sinnvoll, durch Rotationen erzeugen. Zusätzlich lassen sich weitere Bilder durch hinzufügen von Rauschen oder Verzerrungen des Original Bilds erstellen. Weiterhin kann eine Überanpassung durch sogenanntes Early Stopping vermieden werden. Auch ein gut abgestimmtes Modell neigt zu einem Overfitting, wenn dieses zu lange auf den gleichen Daten trainiert wird. Um dies zu vermeiden, ist es möglich, sobald sich eine Überanpassung andeutet, das Training zu beenden. Erkennen lässt sich eine Überanpassung daran, dass die Ergebnisse auf dem Trainingsdatensatz konstant bleiben oder besser werden, gleichzeitig sich die Ergebnisse auf dem Testdatensatz jedoch nicht verändern, oder sogar verschlechtern. [10, 24]²

2.8 Parameter Optimierung

Ob ein Machine Learning Modell zu einem Over- oder Underfitting neigt, wird durch sogenannte Parameter beeinflusst. Beim CNN unterscheidet man prinzipiell zwischen zwei Arten von Parametern, den trainierbaren und nicht trainierbaren Parametern, den sogenannten Hyperparametern. Trainierbare Parameter sind Parameter, welche das Modell selbst während des Trainings erlernt. Bei diesen Parametern handelt es sich um mathematische Operationen, welche der Algorithmus innerhalb des Modells ausführt. Auf diese Operationen hat der Entwickler selbst keinen direkten Zugriff, da der Algorithmus diese durch die Trainingsdaten selbst erlernt. Beeinflussen kann der Entwickler jedoch die nicht trainierbaren Hyperparameter, welche vom Ersteller des Modells vor der eigentlichen Trainingsphase festgelegt werden. Diese Hyperparameter legen die Kapazität des Modells fest und können nicht vom Modell selbst verändert werden. Die Wahl der Hyperparameter kann durch Annahmen und Erfahrungswerten erfolgen, oder auch durch Anwenden des Trial-and-Error Prinzips. Es existieren jedoch auch automatisierte Verfahren, wie das rechenintensive Grid-Search oder Random-Search. [23] Zur Zeit wird allerdings auch an Verfahren geforscht, welche diese Parameteroptimierung durch Deep Learning Algorithmen automatisieren sollen. [10, 25]²

2.9 Modellevaluierung

Aus den bisherigen Kapiteln geht hervor, dass ein Machine Learning Algorithmus während der Trainingsphase an bekannte Daten angepasst wird und anschließend das erlernte Wissen auf unbekannte Daten überträgt. Dazu muss jedoch der trainierte Algorithmus zuverlässig und reproduzierbar überprüft werden, um somit dessen Leistungsfähigkeit einschätzen zu können.

2.9.1 Hold-out Methode

Eine Möglichkeit der Modellevaluierung ist die sogenannte Hold-out Methode. Bei dieser wird der vorliegende Datensatz in einen Trainings- und Testdatensatz aufgeteilt. Meist wird noch ein weiterer sogenannter Validierungsdatsatz erstellt. Typische Verhältnisse der Datensätze zueinander sind 80 % der Daten für den Trainingsdatensatz, 10 % für den Validierungsdatsatz und 10 % für den Testdatensatz. Das Modell erlernt die für die Aufgabenstellung wichtigen Merkmale während der Trainingsphase aus den Trainingsdaten und wird währenddessen periodisch auf dem Validierungsdatsatz überprüft. Durch Parameteroptimierung wird versucht das Modell so zu trainieren, dass auf dem Validierungsdatsatz möglichst gute Ergebnisse erreicht werden. Sind die Ergebnisse auf dem Validierungsdatsatz zufriedenstellend, wird anschließend ein finaler Test auf dem Testdatensatz durchgeführt, um unabhängig vom Validierungsdatsatz und Parametertuning überprüfen zu können, wie der Algorithmus auf völlig unbekannten Daten abschneidet. Dies ist notwendig, da durch die Parameteroptimierung die Gefahr besteht, dass der Algorithmus genau auf den vorliegenden Validierungsdatsatz angepasst wird. Auf diesem würde das Modell dann zwar auch gute Ergebnisse liefern, auf völlig unbekannten Daten würde das Modell jedoch deutlich schlechter abschneiden. Dieses Szenario soll durch den dritten Datensatz, dem finalen Testdatensatz verhindert werden. Dabei ist es wichtig sicherzustellen, dass alle drei Datensätze für die Problemstellung repräsentativ sind. [26, 24] Im Falle eines Klassifizierers bedeutet das etwa, dass alle zu erkennenden Klassen über alle drei Datensätze hinweg vorhanden sein müssen. [10]²

2.9.2 Precision, Recall, F1-Score

Ein Modell wird, wie bereits erwähnt, durch Parameteroptimierung so angepasst, dass es auf den Validierungsdaten möglichst gute Ergebnisse liefert. Es reicht jedoch nicht, diese Ergebnisse lediglich als gut oder schlecht zu bezeichnen. Stattdessen wird hierzu ein objektives Maß benötigt, die sogenannten Metriken. Eine der einfachsten Metriken zur Beurteilung eines Klassifizierers ist die sogenannte Accuracy (Formel 2.1).

$$Accuracy = \frac{\text{Anzahl korrekter Vorhersagen}}{\text{Anzahl aller Vorhersagen}} \quad (2.1)$$

Die Accuracy berechnet dabei den Anteil der korrekten Vorhersagen aus der Summe aller Vorhersagen. Ist die Aufgabenstellung eine binäre Klassifizierung, bei der beide Klassen die gleiche Anzahl an Beispielen besitzen, entspricht eine Accuracy von 50 % dem Zufall. Problematisch ist die Accuracy bei unausgewogenen Klassen. Macht eine Klasse 70 % des Datensatzes aus, und die andere Klasse lediglich 30 %, so kann ein Modell sehr einfach eine Accuracy von 70 % erreichen, indem es bei der Vorhersage einfach immer die größte Klasse annimmt. Dies bedeutet allerdings, dass das Modell für die Unterscheidung der einzelnen Klassen effektiv gesehen nichts gelernt hat. Verwendet werden kann die Accuracy jedoch als Orientierungshilfe, der sogenannten Baseline. Diese Baseline stellt somit die bestmögliche Accuracy dar, ohne dass der Algorithmus etwas erlernen musste.

Neben der Accuracy wird außerdem häufig die Precision (Formel 2.2)

$$Precision = \frac{TP \text{ (richtig positive)}}{TP \text{ (richtig positive)} + FP \text{ (falsch positive)}} \quad (2.2)$$

sowie der Recall (Formel 2.3) verwendet.

$$Recall = \frac{TP \text{ (richtig positive)}}{TP \text{ (richtig positive)} + FN \text{ (falsch negative)}} \quad (2.3)$$

2 Grundlagen

Die Precision bestimmt dabei den Anteil der richtig vorhergesagten positiven Beobachtungen in Bezug auf alle als positiv vorhergesagten Beobachtungen. Beim Rückblick auf das Beispiel der Spam E-Mails bedeutet dies, dass eine sehr hohe Precision erwünscht ist, da es besser ist, wenn eine Spam E-Mail übersehen wird, als dass eine legitime E-Mail gelöscht wird.

Der Recall dagegen bestimmt den Anteil der richtig vorhergesagten positiven Beobachtungen in Bezug auf alle tatsächlichen positiven Beobachtungen. Ein hoher Recall ist etwa bei der Krebserkennung erwünscht, da es hier besser ist eine gesunde Zelle als Krebszelle zu erkennen, als eine Krebszelle zu übersehen.

Precision und Recall sind inverse proportional zueinander, d.h. eine Verbesserung des einen Wertes geht zu lasten des anderen. Ein Kompromiss dieser beiden Metriken stellt der sogenannte F_1 -Score da (Formel 2.4). Der F_1 -Score bildet den harmonischen Mittelwert aus Precision und Recall.

$$F_1\text{-Score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (2.4)$$

Bei Datensätzen, welche ein starkes Ungleichgewicht aufweisen, kann es sinnvoll sein, zusätzliche Metriken zu betrachten. Hier gibt es zusätzlich die Macro Average Accuracy und die Weighted Average Accuracy. Die Macro Average Accuracy gewichtet alle Klassen gleich und berechnet daraus die Genauigkeit über alle Klassen. Dadurch dass alle Klassen gleich gewichtet werden, sorgt dies dafür, dass ein schlechtes Ergebnis auf einer Minderheitenklasse das Gesamtergebnis stärker verschlechtert. Dagegen gewichtet die Weighted Average Accuracy die einzelnen Klassen entsprechend ihres Auftretens entsprechend und errechnet daraus die Genauigkeit über alle Klassen. Dadurch hat ein eventuell schlechteres Ergebnis auf einer Minderheitenklasse einen geringeren Einfluss auf das Gesamtergebnis. [26, 24]

2.9.3 Konfusionsmatrix

Eine weitere sehr hilfreiche Metrik der Klassifizierung ist die sogenannte Konfusionsmatrix (Abbildung 2.9).

Tatsächliche Klasse	positiv	<div>TP (richtig positiv)</div>	<div>FN (falsch negativ)</div>
	negativ	<div>FP (falsch positiv)</div>	<div>TN (richtig negativ)</div>
		positiv	negativ
		Vorhergesagte Klasse	

Abbildung 2.9: Beispiel einer Konfusionsmatrix

Die Konfusionsmatrix zeigt dabei die richtig klassifizierten Beispiele in der Hauptdiagonalen von oben links nach unten rechts an. Die tatsächlichen Klassen finden sich in den Zeilen wieder, die durch das Modell vorhergesagten Label dagegen lassen sich aus den Spalten ablesen. Auf diese Weise erhält man unterhalb der Hauptdiagonalen die Anzahl der fälschlicherweise als positiv klassifizierten Beispiele. Oberhalb der Diagonalen dagegen finden sich die fälschlicherweise als negativ kategorisierten Beobachtungen. Diese Konstellation lässt sich für beliebig viele Klassen anwenden und hat den Vorteil, dass auf einen Blick mögliche Zusammenhänge zwischen den einzelnen Klassen erkannt werden können. Durch die Konfusionsmatrix lässt sich direkt erkennen, ob ein Klassifizierungsalgorithmus auf einer Klasse besser funktioniert als auf einer anderen oder ob zwei Klassen viele Gemeinsamkeiten besitzen, wodurch Beispiele einer Klasse oft einer anderen zugeordnet werden. [10, 24, 26, 27]²

2.10 Neuronale Netze

Künstliche Neuronale Netze (engl. Convolutional Neural Networks) sind spezielle Machine Learning Algorithmen, welche sich am Aufbau biologischer Neuronen des Gehirns orientieren. Das Gehirn besteht aus verbundenen Nervenzellen, den sogenannten Neuronen. Diese verarbeiten Signale und leiten diese wiederum an andere Neuronen weiter. Das menschliche Gehirn besteht etwa aus mehreren Milliarden Neuronen, welche miteinander verknüpft sind. [28] Von diesem Prinzip haben sich Warren McCulloch und Walter Pitts bei der Untersuchung des Gehirns inspirieren lassen, und darauf basierend das McCulloch-Pitts-Neuron entwickelt, welches eine einzelne Nervenzelle darstellt. Dieses künstliche Neuron besteht aus einem oder mehreren binären Eingängen, das durch ein oder mehrere Signale am Eingang ein zugehöriges binäres Signal am Ausgang liefert. Basierend auf dem McCulloch-Pitts-Neuron wurde einige Jahre später das Perzeptron entwickelt, welches die Eingangssignale mit unterschiedlichen Koeffizienten multipliziert und mittels einer Aktivierungsfunktion darüber entscheidet, ob der Neuronenausgang ein Signal ausgibt (Abbildung 2.10). [10, 21]²

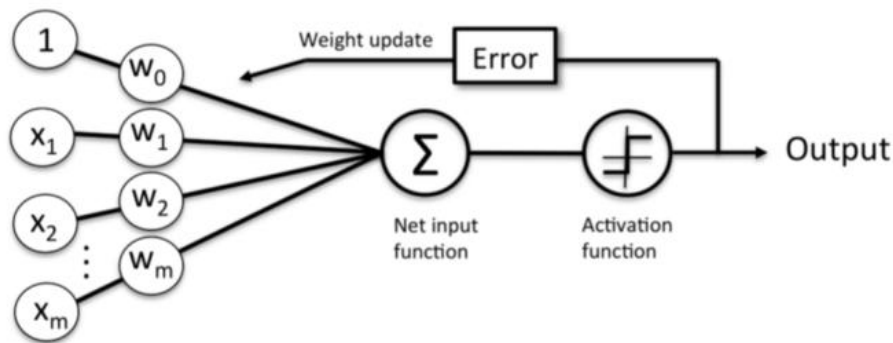


Abbildung 2.10: Aufbau eines Perzeptrons(Quelle: [29]))

Wie in Abbildung 2.10 zu sehen ist, gibt es eine zusätzliche Abzweigung am Ausgang. Dieser rückwärts wirkende Pfad wird aus der Differenz zwischen dem tatsächlichen Ist-Wert der Ausgabe, und der vom Modell vorhergesagten Ausgabe berechnet. Anschließend wird die so berechnete Differenz zu den Eingängen zurückgeführt und deren Gewichtungen entsprechend geändert, sodass die am Ausgang entstehende Differenz möglichst gering ist. Dadurch, dass Neuronen miteinander verbunden sind, d.h. der Ausgang eines Neurons ist der Eingang eines Nachfolgenden, zieht sich diese Differenzminimierung durch viele Neuronen hindurch. Dieses Verfahren nennt sich Backpropagation und ist ein wichtiger Baustein für den Erfolg von CNNs. [21]

2.10.1 Kostenfunktion und deren Minimierung

Die Backpropagation lässt dabei schon erahnen, dass Maschinen Lernen letztendlich ein Verfahren zum Lösen einer Extremwertaufgabe ist. Das Ziel liegt dabei darin, das globale Minimum einer mathematischen Funktion zu bestimmen. Dazu werden sogenannte Kostenfunktionen (engl. Loss Functions) oder auch Straffunktionen eingesetzt. Die Kostenfunktionen übertragen dafür das Problem aus der realen Welt in die Mathematik. Mit anderen Worten ist die Klassifizierungsaufgabe mathematisch betrachtet eine Wahrscheinlichkeitsverteilung, welche ausgibt, mit welcher Wahrscheinlichkeit ein Datenpunkt zu einer oder mehrerer Klassen gehört. Eine dieser Kostenfunktionen ist dabei die Kreuzentropie (engl. Cross-Entropy, Formel 2.5).

$$H(y_{true}, y_{pred}) = - \sum_{i=1}^N y_{true}(x) \cdot \log(y_{pred}(x)) \text{ für } i = 1, \dots, N \quad (2.5)$$

Wobei y_{true} die tatsächlich korrekten Label und y_{pred} die vorhergesagte Wahrscheinlichkeiten für x sind. N ist dabei die Anzahl der vorhandenen Klassen. Die Kreuzentropie berechnet die Distanz zwischen zwei Wahrscheinlichkeitsverteilungen, d.h. bei steigender Abweichung zwischen vorhergesagten und tatsächlichen Labeln steigt auch die Kreuzentropie an. [26, 30]

2 Grundlagen

Die Daten, welche in das Neuronale Netz hineingegeben werden, durchqueren diverse miteinander verbundene Neuronen. Am Ende des Netzes wird die Klassenzugehörigkeit der jeweiligen Beispiele vorhergesagt und durch die Kreuzentropie anschließend die Abweichung zwischen den tatsächlichen und den vorhergesagten Klassen berechnet. Dieser Schritt passiert im vorwärts gerichteten Teil des Neuronalen Netzes (engl. Feedforward neural network). Für die Anpassung der einzelnen Neuronen wird die bereits erwähnte Backpropagation verwendet. Dies ist der rückwärts gerichtete Teil des Netzes. Um zu bestimmen, wie stark die Neuronen angepasst werden müssen, wird die Ableitung der Kostenfunktion gebildet und anschließend durch Optimierungsverfahren minimiert. Die Schwierigkeit besteht hier darin, dass globale Minimum zu lokalisieren (Abbildung 2.11).

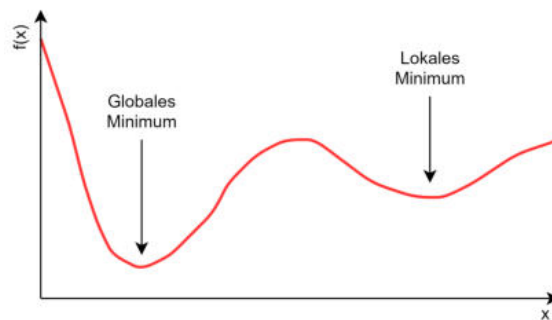


Abbildung 2.11: Im Graph ist neben dem globalen Minimum noch ein weiteres lokales Minimum zu sehen

Die Suche nach diesem Minimum wird durch unterschiedliche Lernverfahren optimiert, wobei der wichtigste Optimierer der stochastische Gradientenabstieg (engl. Stochastic Gradient Descent, kurz SGD) ist. Dieser berechnet die Ableitung der Kreuzentropie nicht auf den kompletten Trainingsdaten, sondern nur auf einer kleinen zufällig gewählten Teilmenge. Dadurch wird das Neuronale Netz mehrmals pro Epoche angepasst, anstatt das Netz nur einmal am Ende einer Trainingsepoche zu aktualisieren. Das Training wird dadurch insgesamt beschleunigt. Weitere Optimierer häufig verwendete Optimierer wie der Adam, Momentum oder etwa RMSProp Optimierer basieren ebenfalls auf dem Gradientenabstiegsverfahren. [22]

2.10.2 Aktivierungsfunktionen

Vor allem bei tiefen Neuronalen Netzen spielt die Aktivierungsfunktion am Ausgang der Neuronen eine besondere Rolle. Von tiefen Neuronalen Netzen (engl. Deep Neural Networks), wird dabei immer dann gesprochen, wenn das Netz sehr viele aufeinanderfolgende Layer besitzt. Die Aktivierungsfunktion hat die Aufgabe, den Ausgabewert eines Neurons auf einen bestimmten Wertebereich zu beschränken. Wie bereits erwähnt, ist der Ausgang eines Neurons mit den Eingängen nachfolgender Neuronen verbunden. Durch diese Verkettung kann es passieren, dass ein Signal beim durchlaufen der einzelnen Neuronen zu klein wird und irgendwann komplett verschwindet. Dieser Prozess wird als vanishing gradients problem bezeichnet. Andererseits kann aber auch das Gegenteil eintreten, welches dann als exploding gradients problem bezeichnet wird. Zusätzlich muss die Aktivierungsfunktion differenzierbar sein, da diese für die Differenzminimierung des Backpropagation Algorithmus benötigt wird. Außerdem fügt die Aktivierungsfunktion dem Neuronalen Netz eine nicht linearität hinzu, wodurch es erst ermöglicht wird, komplexe nicht lineare Probleme zu modellieren. [21, 24, 27, 31]

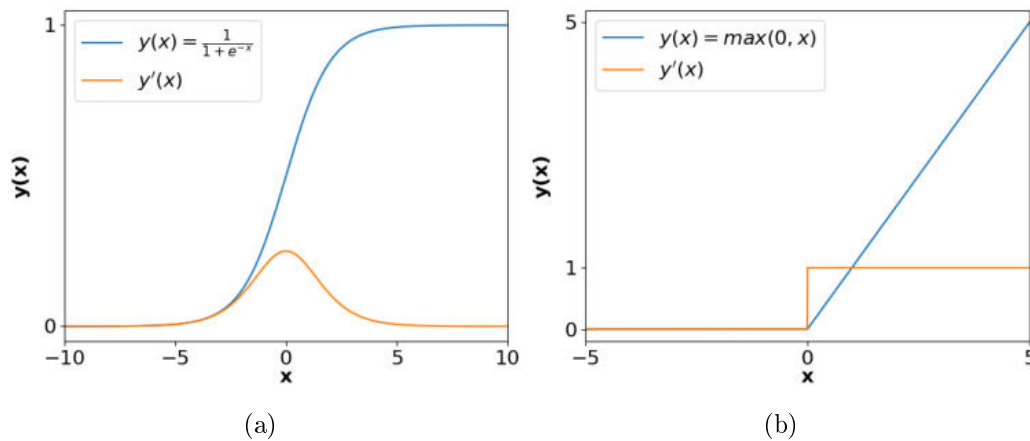


Abbildung 2.12: Sigmoid (a) und ReLU Aktivierungsfunktionen (b) in blau und ihre jeweiligen Ableitungen in orange.

2 Grundlagen

Durchgesetzt haben sich bei tiefen Neuronalen Netzen vor allem die Rectified Linear Units. Diese beschleunigen das Training und verringern zudem das Problem der verschwindenden Gradienten. In Abbildung 2.12 ist neben der ReLU Funktion auch die Sigmoid Funktion zu sehen, welche ebenfalls häufig eingesetzt wurde. Vor allem die Ableitungen der Sigmoid und ReLU Funktion in Abbildung 2.12 verdeutlichen die Unterschiede der beiden Aktivierungsfunktionen. Die Sigmoid Funktion ist zwar überall differenzierbar, jedoch hat sie die Eigenschaft, dass ihre Ableitung für große Werte von $|x|$ gegen Null läuft. Dies ist vor allem bei tiefen Neuronalen Netzen ein Problem, da bei mehreren Layern die sehr kleinen Gradienten aufgrund der Backpropagation miteinander multipliziert werden und somit noch schneller verschwinden. Dagegen kommt es bei der ReLU Funktion (Abbildung 2.12(b)) auch für große Input Werte von $|x|$ zu keiner Sättigung, da die Ableitung für $x > 0$ immer 1 ist. Dadurch wird vermieden, dass die Gradienten in tiefen Netzen verschwinden können. Dafür hat die ReLU Aktivierungsfunktion allerdings Probleme mit nicht feuernenden Neuronen bei negativen Werten von x , was wiederum die Accuracy des Netzes verschlechtern kann. [32, 33]

2.10.3 Convolutional Neural Network

Ein Convolutional Neural Network stützt sich auf die Grundannahme, dass benachbarte Pixel innerhalb eines Bildes eine engere Beziehung zueinander haben als weiter voneinander entfernte Pixel. So können lokale Merkmale aus einer kleinen Bildregion extrahiert werden, welche auch in anderen Bildbereichen nützlich sein könnten. Außerdem können solche lokalen Merkmale, welche jeweils nur einen kleinen Bildbereich umfassen durch ein Zusammenführen Rückschlüsse auf das gesamte Bild liefern. Umgesetzt werden diese Annahmen durch eine Abfolge von aufeinanderfolgenden Ebenen, sogenannten Layern. Die einfachste und auch häufigste Architektur von CNNs besteht aus einem Convolutional Layer und einem darauf folgenden Pooling Layer. Dieser Aufbau kann beliebig oft wiederholt werden und wird von einem Fully Connected Layer abgeschlossen. Auch komplexe CNN Architekturen wie die ResNet oder VGG Netzarchitektur machen sich diese Abfolge, wenn auch abgewandelt, zunutze. [10, 31, 34, 35]²

2.10.4 Convolutional Layer

Der namensgebende Convolutional Layer ist für die Merkmalsextraktion eines Bildes verantwortlich. Dazu besitzt dieser mehrere sogenannte Kernel, welche auf ein Bild mittels der Faltungsoperation angewendet werden. Bei der Faltungsoperation wird ein Signal $x(t)$ mit dem gewichteten Mittelwert eines weiteren Signals $w(t)$ überlagert. Als Faltungsoperator wird ein Asterisk verwendet (Formel 2.6). [27]

$$y(t) = \int_{-\infty}^{\infty} x(a) \cdot w(t-a) da = x(t) * w(t) \quad (2.6)$$

Da Computer jedoch mit zeitdiskreten Werten arbeiten, wird die zeitdiskrete Faltung (Formel 2.7) verwendet. [27]

$$y(t) = x(t) * w(t) = \sum_{a=-\infty}^{\infty} x(a) \cdot w(t-a) \quad (2.7)$$

Graphisch lässt sich die Faltungsoperation als Matrixmultiplikation zwischen den Kernen und dem eigentlichen Bild (Abbildung 2.13) betrachten.

2 Grundlagen

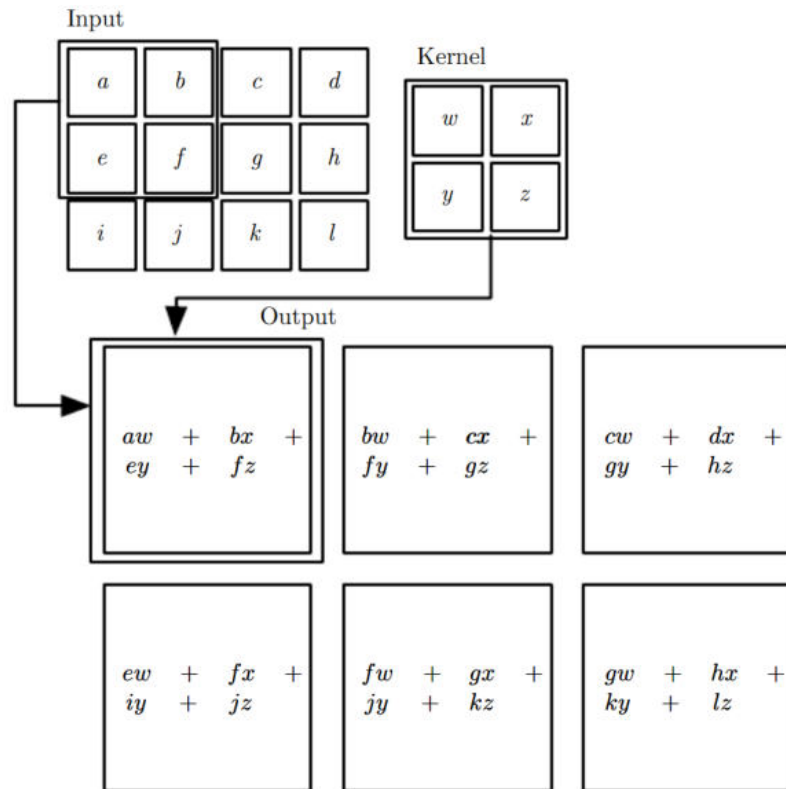


Abbildung 2.13: Beispiel einer 2D Faltung zwischen dem Bild am Eingang (Input) und dem Kernel (Quelle: [27])

Die Kernel selbst sind dabei Matrizen, welche meist eine Dimension von 3×3 , 5×5 oder 7×7 Werten besitzen. Die genauen Werte innerhalb der Kernelmatrizen werden vom CNN während der Trainingsphase erlernt. Dadurch, dass die einzelnen Kernel unterschiedliche Werte besitzen, werden verschiedene Merkmale aus dem Eingangsbild extrahiert. Auf diese Weise entsteht durch jeden Kernel eine sogenannte feature map, welche nur bestimmte Merkmale des ursprünglichen Bildes enthält. So kann beispielsweise eine feature map nur horizontale, und eine andere nur vertikale Linien enthalten. [10, 24, 26, 27, 31]²

2.10.5 Pooling Layer

In der Regel folgt auf einen Convolutional Layer ein sogenannter Pooling Layer, welcher die Aufgabe besitzt, die räumliche Dimension eines Bildes zu reduzieren. Dazu wird meist ein 2×2 Filter verwendet, d.h. es werden vier Pixel innerhalb eines 2×2 Filters betrachtet und anschließend zu einem neuen Pixel zusammengefasst (Abbildung 2.14). Durchgesetzt hat sich dabei das sogenannte MAXPooling, welches den Wert des größten Pixels übernimmt.

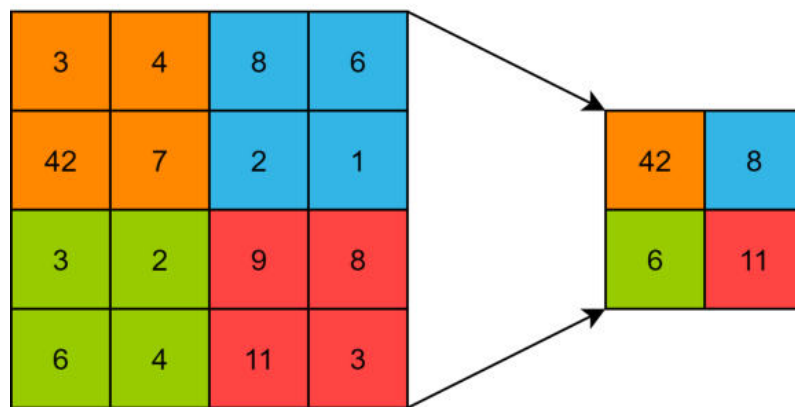


Abbildung 2.14: Beispiel einer MAXPooling Operation

Durch diese Zusammenfassung von Pixeln gehen zwar einige Informationen verloren, jedoch werden dadurch die Anzahl der zu trainierenden Parameter reduziert, was wiederum einem Overfitting entgegen wirkt. Außerdem fügt der Pooling Layer dem Neuronalen Netz eine translatorische Invarianz hinzu. So ist es beispielsweise bei der Gesichtserkennung nicht notwendig, die Pixelgenaue Position der Augen zu erkennen, sondern lediglich, dass zwei Augen vorhanden sind. Die Dimensionsreduktion in Kombination mit anschließendem erneuten Verwenden von Convolutional Layern macht es möglich, nachfolgende Kernelmatrizen auf neue Merkmale zu trainieren, welche erst sichtbar werden, wenn ein größerer Bildbereich betrachtet wird. [10, 24, 27]²

2.10.6 Fully Connected Layer

Der Fully Connected Layer ist mit allen Neuronen der vorherigen Schicht verbunden. Durch das Training erlernt dieser, welche Neuronen zu bestimmten Zeitpunkten feuern und wie diese in Zusammenhang mit einer Klasse zu interpretieren sind. Dieser ist somit für die eigentliche Klassifizierung verantwortlich. Je nachdem, wie komplex die Aufgabe ist, kann es auch sein dass ein oder mehrere Fully Connected Layer verwendet werden. [10, 24]²

Beim letzten Fully Connected Layer entspricht die Anzahl der Neuronen der Anzahl der zu erkennenden Klassen, wobei jedem Neuron eine Klasse zugewiesen wird. Als Aktivierungsfunktion wird hier in der Regel bei einer binären Klassifizierung, also der Unterscheidung zwischen zwei potentiell möglichen Klassen, die Sigmoid Funktion verwendet. Wenn mehr als zwei Klassen erkannt werden müssen, wird auf eine Softmax Funktion zurück gegriffen. Diese normalisiert die Ausgabewerte der Neuronen im letzten FC Layer auf einen Wertebereich zwischen 0 und 1 (Formel 2.8).

$$y(x)_j = \frac{e^{x_j}}{\sum_{i=1}^N e^{x_i}} \text{ für } j = 1, \dots, N \quad (2.8)$$

Wobei N für die Anzahl der möglichen Klassen steht. Ausgegeben wird mit $y(x)_j$ ein Vektor, welcher die Wahrscheinlichkeitsverteilung angibt, mit der eine Beobachtung x_j zu den jeweiligen Klassen $y_j \in 1, \dots, N$ gehört. [31, 33]

2.10.7 Batch Normalization

Vor allem bei großen Datensätzen wie Bildern ist es nicht immer möglich, alle Daten gleichzeitig für das Training zu verwenden. Deshalb wird das Neuronale Netz in sogenannten Batches trainiert und auch evaluiert. Ein Batch ist dabei eine zufällige Zusammenstellung von einigen Beispielen aus dem verwendeten Datensatz. Das zu trainierende Modell wird gleichzeitig auf einem Batch von Beispielsweise 32 Bildern trainiert und anschließend auf dieses Batch angepasst. Dieser Vorgang wird mehrmals wiederholt. Wenn der komplette Trainingsdatensatz auf diese Weise das Neuronale Netz durchlaufen hat, spricht man von einer sogenannten Epoche. Vor der Verwendung wird der zu verwendende Datensatz, wie bereits erwähnt, aufbereitet und normalisiert. Jedoch kann es während dem Training dennoch dazu kommen, dass die Ausgabewerte der Neuronen verschwinden, weil sie zu klein werden oder explodieren und ins unendliche ansteigen. Um diesem Problem entgegen zu wirken, wurde die sogenannte Batch Normalization entwickelt. Diese soll die Ausgabe eines Layers bzw. dessen Neuronen normalisieren. Dadurch soll sichergestellt werden, dass die Ausgabe der Neuronen im Mittel um 0, und die Standardabweichung etwa bei 1 liegt. Dazu wird die Normalisierung während des Trainings Batchweise berechnet (Formel 2.9).

$$\bar{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}} \quad (2.9)$$

Während des Trainings wird der Mittelwert μ_β (Formel 2.10) und die Varianz σ_β^2 (Formel 2.11) für jede Batch einzeln berechnet.

$$\mu_\beta = \frac{1}{m} \sum_{i=1}^m x_i \quad (2.10)$$

$$\sigma_\beta^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_\beta)^2 \quad (2.11)$$

2 Grundlagen

Während der Testphase hingegen wird μ_β und σ_β durch ihren jeweiligen laufenden Mittelwert ersetzt, welcher während der Trainingsphase berechnet wurde. Dadurch soll vermieden werden, dass das Modell durch die letzte Mini-Batch einen Bias erhält. Des Weiteren beschleunigt und stabilisiert die Batch Normalisierung das Training, da die Schwankungen der Neuronen Ausgaben verringert werden. [22, 36]

2.10.8 Dropout

Eine weitere Methodik um Overfitting zu vermeiden ist das zufällige deaktivieren von Neuronen während der Lernphase. Dies wird mittels eines sogenannten Dropout Layers umgesetzt. Durch das zufällige deaktivieren von Neuronen soll das Neuronale Netz robuster und Abhängigkeiten zwischen einzelnen Neuronen minimiert werden, da diese sich nicht mehr darauf verlassen können, dass unter bestimmten Bedingungen immer die gleichen Neuronen aktiv sind. Üblicherweise wird dabei das Dropout auf den Fully Connected Layer angewendet, da dieser die meisten Parameter hat und deshalb am ehesten zum Overfitting neigt. Auch wenn dagegen Convolutional Layer relativ wenige Parameter besitzen und eher selten von Overfitting betroffen sind, kann es bei einigen CNN Architekturen hilfreich sein, das Dropout bei diesen Layern einzusetzen. Durch das Entfernen von Neuronen im Convolutional Layer wird ein zusätzliches Rauschen in den Input gebracht, was für einige Problemstellungen hilfreich sein kann. Das Rauschen kommt dabei dadurch zustande, dass zufällige Elemente der Matrizen entfernt werden. [37, 38, 39]

2.10.9 Integrated Gradients

Ein häufiges Problem des maschinellen Lernens besteht darin, dass für den Menschen der angewendete Algorithmus einer Black Box ähnelt. Es ist zwar bekannt welche Informationen in das Modell hineingegeben werden und welche heraus kommen, doch wie genau der ML-Algorithmus zu diesem Ergebnis kommt ist nicht immer nachvollziehbar. Um diese Entscheidungsfindung transparenter und nachvollziehbarer zu machen wurden die sogenannten Integrated Gradients entwickelt, welche zur Methodik der Explainable AI gehören.

Um zu erkennen welche Bereiche innerhalb eines Bildes für die Klassifizierung wichtig sind und welche nicht, werden die Gradientenverläufe der Backpropagation für die Eingangsdaten ausgewertet. Da Backpropagation die Differenz zwischen tatsächlicher und vorhergesagter Klasse widerspiegelt, bedeutet dies, dass sich für eine korrekte Klassifizierung die Kostenfunktion gegen Null bewegt. Dadurch bewegt sich jedoch auch die Backpropagation gegen Null. In diesem Fall würde die Backpropagation nichts interpretierbares aussagen. Um dem jedoch entgegenzuwirken, wird das zu klassifizierende Bild künstlich verschlechtert indem die Farbintensität zunächst soweit heruntergeregelt wird, bis nur noch ein schwarzes Bild zu sehen ist. Dieses schwarze Bild dient als Startpunkt, die sogenannte Baseline. Nun wird die Bildintensität leicht erhöht wodurch sich im schwarzen Bild leichte Konturen zu erkennen geben. Dieses Bild wird nun vom Modell klassifiziert und anschließend werden der Gradient der Backpropagation ausgewertet. Dieses Verfahren wird solange wiederholt, bis der Gradient der Backpropagation anfängt zu sättigen und das Modell die Klassifizierung richtig vornimmt. Mit jedem dieser Schritte wird das Bild deutlicher und für das Netz leichter zu erkennen. Sobald das Netz beginnt das Bild richtig zu klassifizieren wird an der Bildintensität nichts mehr geändert und für diese Aufnahme ausgewertet, welche der Bildregionen den größten Einfluss auf die Klassifizierung haben. [40, 41]

3 Praktische Umsetzung

Um im Rahmen dieser Masterarbeit ein Convolutional Neuroal Network umzusetzen, wurde die Programmiersprache Python in Version 3.7 gewählt. Diese erlaubt durch eine unkomplizierte Syntax den schnellen Einstieg in die Entwicklung von verschiedenen Maschine Learning Prototypen, ohne sich zusätzliche Gedanken wie etwa einer Speicherallokierung machen zu müssen. Als Framework wurde die Open Source Bibliothek Tensorflow in Version 2.3.0 verwendet. Diese wurde für den Einsatz und Aufbau von Machine Learning Verfahren konzipiert und ermöglicht es, durch vorgefertigte Schnittstellen mathematische Funktionen auf einer dedizierten Grafikkarte zu berechnen. Grafikkarten besitzen eine grundlegend andere Struktur als CPUs und haben ihre Stärke vor allem im Bereich der parallelen Datenverarbeitung, was dem Machine Learning zugutekommt. Außerdem besitzen moderne Grafikkarten oft speziell für den Einsatz von Machine Learning Anwendungen entwickelte Hard- und Software Funktionen, welches das rechenintensive Training zusätzlich beschleunigen. Durchgeführt wurden die Berechnungen auf einem Rechnersystem mit einer NVIDIA GTX 1070, welche 8GB Grafikspeicher zur Verfügung stellt.

3.1 Erfasste Prozessdaten

Während des 3D-Drucks fallen unterschiedliche Prozessdaten an, welche protokolliert wurden und für diese Arbeit als Datengrundlage dienen. Zum einen erfassen mehrere Sensoren in periodischen Intervallen verschiedene Informationen zum Bauprozess selbst und zur unmittelbaren Umgebung innerhalb der Prozesskammer. Zum anderen wird nach jeder abgeschlossenen Schicht mittels einer Kamera ein Bild der Ebene erstellt. Zusätzlich liegt das 3D-Modell des Bauobjekts vor, welches für jede einzelne Schicht den Pfad des Laserstrahls sowie eventuell benötigte Supportstrukturen definiert. Zuletzt liegt als weitere Information vor, ob der Druckprozess erfolgreich oder nicht erfolgreich war, und im letzteren Falle, ein zusätzlicher Kommentar, der die Ursache für den fehlerhaften Druckprozess beschreibt. Wobei diese Ursachenabschätzung lediglich den subjektiven Eindruck und die Erfahrung des Maschinenoperators darstellt. Nachdem eine Ebene fertiggestellt wurde, wird von einer Kamera aus der Vogelperspektive eine Aufnahme erstellt (Abbildung 3.1). Dies wird für jede einzelne Schicht eines Baujobs wiederholt. Zur Unterscheidung zwischen den einzelnen Ebenen, werden die Aufnahmen nach der jeweiligen Schicht bzw. Höhenlage benannt.

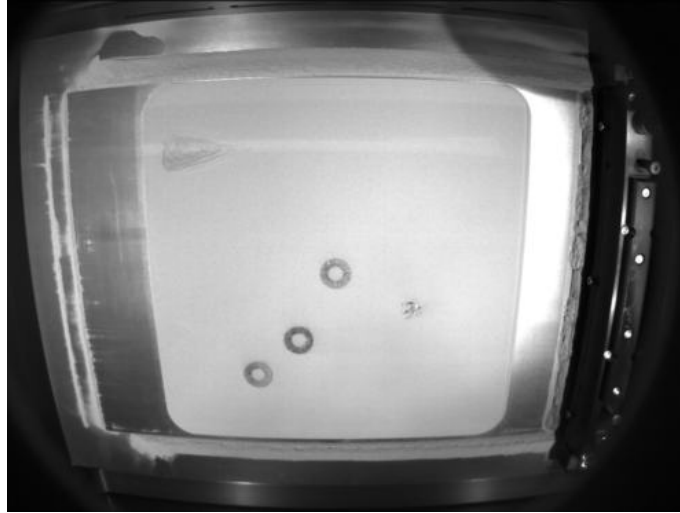


Abbildung 3.1: Typische Schichtaufnahme eines Baujobs während des Druckprozesses

Sensoren fragen periodisch, im Fall der vorliegenden Sensordaten etwa alle zwei Sekunden, verschiedene Messwerte ab und protokollieren diese anschließend. Somit lassen sich einer einzelnen Ebene, je nach Komplexität, mehrere Sensorwerte zuweisen. Zu den gemessenen Werten zählen unter anderem die Zeit, Sauerstoffgehalt innerhalb der Kammer, Gas Durchflussrate, die Temperatur innerhalb der Kammer und die abgegebene Leistung des Lasers. Zusätzlich liegt außerdem das 3D-Modell des zu druckenden Objekts vor, wobei dieses in einem proprietärem und verschlüsselten Format abgespeichert wird.

Insgesamt besteht der vorliegende Datensatz aus 56 Baujobs der Jahre 2018 und 2019. Für den 3D-Druck wurde eine LASERTEC 30 der Firma DMG MORI verwendet [42]. Gedruckt wurden bei den Baujobs unterschiedliche Objekte, welche neben Zugproben auch Würfel oder Zylinder umfassen. Die Höhe der Baujobs, und somit auch deren Dauer, variiert dabei von einigen hundert bis einigen tausend Schichten, welche über mehrere Tage hinweg gedruckt wurden.

3.2 Aufbereitung des Datensatzes

Da es sich bei der hier angewendeten Methodik um ein Supervised Machine Learning handelt, werden gelabelte Daten benötigt. Als verwendete Datengrundlage dienen hierzu die abgespeicherten Protokolle von 56 unterschiedlichen Druckaufträgen. Zu jedem Druckauftrag liegt die Beschreibung des Maschinenoperators vor, welche einen abgeschlossenen Druck als fehlerhaft oder fehlerfrei bezeichnet, und im Falle eines fehlerhaften Drucks den Grund des Fehlers. Auf dieser Information und der in Kapitel Fehler im Druckprozess beschriebenen Fehlertypen wurden einzelne Klassen definiert (Tabelle 3.1). Anhand dieser Kriterien wurden anschließend alle Schichtbilder eines Baujobs betrachtet und gelabelt. Zusätzlich wurde ein binäres Label vergeben, welches lediglich zwischen fehlerfreien und fehlerhaften Pulverbettaufnahmen unterscheidet. Dieser Vorgang wurde für alle 56 Druckaufträge wiederholt.

Tabelle 3.1: Festgelegte Label und ihre Definitionen

Label Nummer	Label Klasse	Definition
0	Fehlerfrei	Fehlerfreie Schichtaufnahme
1	Pulverauftragfehler	Dunkle Stellen am Bauteil
2	Bauteilfehler	<ul style="list-style-type: none"> • Ablösen oder Umkippen des Bauteils • Schaben der Beschichter- klinge am Bauteil
3	Deltafehler	<ul style="list-style-type: none"> • Pulver ungleichmäßig verteilt • Mulden im Pulverbett
4	Maschinendefekt	Beschichter- klinge herausgerissen

3.2 Aufbereitung des Datensatzes

Der auf diese Weise erhaltene gelabelte Datensatz wurde nun quantitativ betrachtet. Insgesamt lagen nun 25501 Schichtaufnahmen mit ihren jeweils dazugehörigen Label vor. Von den 25501 Schichtaufnahmen wurden 16011 als fehlerfrei und nur 9490 als fehlerhaft klassifiziert. Noch stärker wurde dieser Kontrast wenn die einzelnen Fehlertypen betrachtet wurden. Hier hatten von den 9490 Schichtaufnahmen 949 einen Pulverfehler, 909 einen Bauteilfehler, 7526 einen Deltafehler und nur 106 Aufnahmen einen Maschinenfehler. In Abbildung 3.2 ist die Verteilung der Schichtaufnahmen entsprechend ihrer zugehörigen Klasse zu sehen.

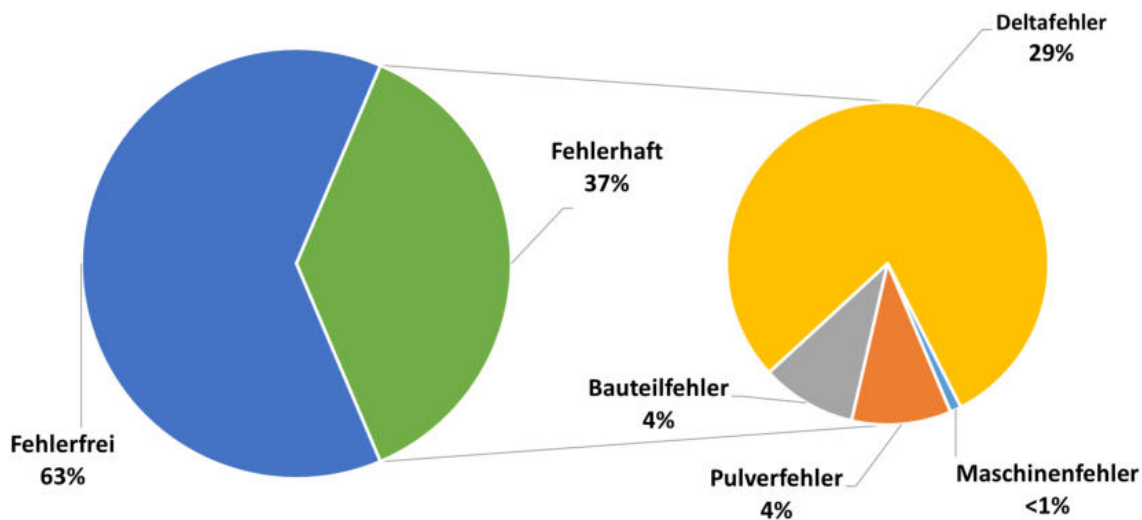


Abbildung 3.2: Nur etwa 37 % aller Schichtaufnahmen weisen überhaupt einen Fehler auf (links). Dabei setzen sich diese 37 % zu einem Großteil aus Deltafehlern zusammen (rechts).

Da der vorliegende Datensatz nur einen begrenzten Umfang und eine sehr starkes Ungleichgewicht besitzt wurde die Problemstellung zunächst vereinfacht. Anstatt direkt ein CNN zu entwickeln, dass den Schichtbildern einzelne Klassen zuordnet, sollte zunächst ein Algorithmus entwickelt werden, der die Aufnahmen in fehlerhafte und fehlerfreie Schichtbilder unterteilt. Hier lag das Verhältnis fehlerfreier zu fehlerhafter Aufnahmen bei etwa 1,7:1.

3 Praktische Umsetzung

Um den Umfang des Datensatzes zu erweitern wurden Augmentierungsverfahren unter Berücksichtigung der auftretenden Fehler angewendet. Es wurde auf eine Rotation verzichtet und lediglich eine Spiegelung der Daten entlang der horizontalen und vertikalen Achsen durchgeführt. Der Grund für dieses Vorgehen liegt darin, dass die Fehler nur in bestimmten Orientierungen entstehen können. Der Beschichter des 3D-Druckers, an welchem sich die Gummilippe befindet, bewegt sich lediglich von rechts nach links und wieder zurück. Dabei wird das Metallpulver auf der Arbeitsfläche verteilt. Die Fehler im Pulverbett entstehen meist daraus, dass die Gummilippe am Bauteil hängen bleibt, das Pulver nicht gleichmäßig verteilt wird, oder das Bauteil sich verzogen hat. Dieses Verhalten der Maschine sorgt dafür, dass die Fehler ebenfalls eine feste Orientierung besitzen, wodurch es wenig Sinn machen würde, die Pulverbettaufnahmen um beispielsweise 90° zu drehen, da die Fehler nicht aus dieser Richtung auftreten können. Durch eine Spiegelung entlang der beiden Achsen wird eine einzelne Aufnahme vervierfacht, wodurch der Trainingsdatensatz um einen Faktor von vier vergrößert wird (siehe Beispiel, Abbildung 3.3).

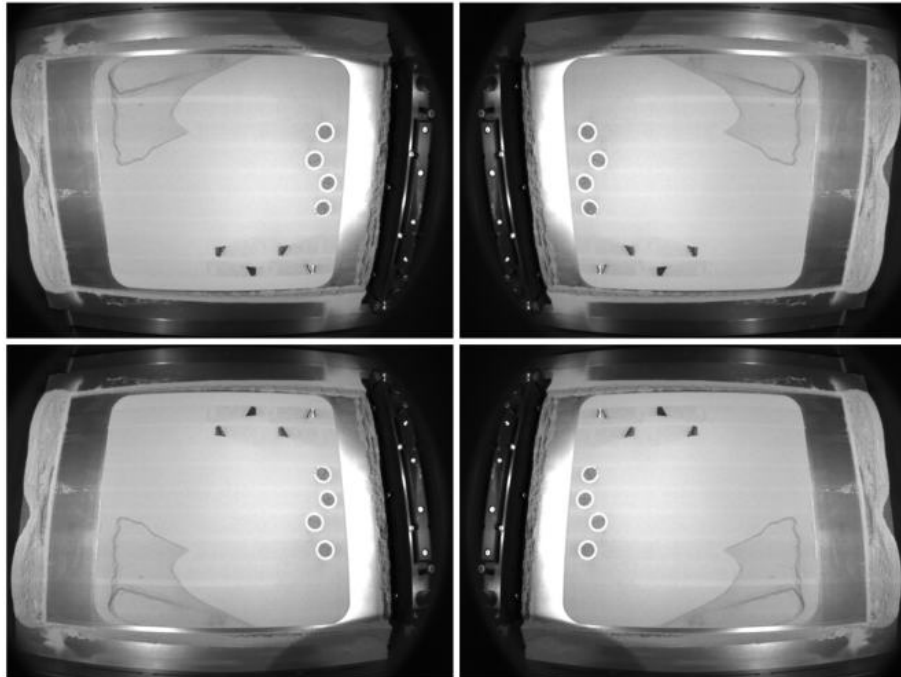


Abbildung 3.3: Durch Spiegelung augmentiertes Schichtbild

3.3 Erster Trainingsversuch: Overfitting

Für die ersten Trainingsversuche wurde der Datensatz in einen Validierungs-, Test- und Trainingsdatensatz aufgeteilt. Als Verhältnis zwischen den einzelnen Datensätzen wurde eine Verteilung von 20 % für den Testdatensatz, 20 % für den Validierungsdatensatz und 60 % für den Trainingsdatensatz gewählt. Des Weiteren wurde darauf Wert gelegt, dass die jeweiligen Datensätze repräsentativ sind, indem diese alle Fehlertypen enthalten. Da die Schichtaufnahmen relativ groß sind und der GPU Speicher der limitierende Faktor ist, wurden die einzelnen Schichtaufnahmen herunterskaliert. Dabei wurde versucht einen Kompromiss zwischen dem Speicherbedarf und einem möglichst geringen Informationsverlust zu finden. Anschließend wurden verschiedene Modelle trainiert, wobei der grundsätzliche Aufbau der Architektur beibehalten wurde. Hier zeigten sich Entgegen der Erwartungen sehr schnell sehr gute Ergebnisse (Abbildung 3.4).

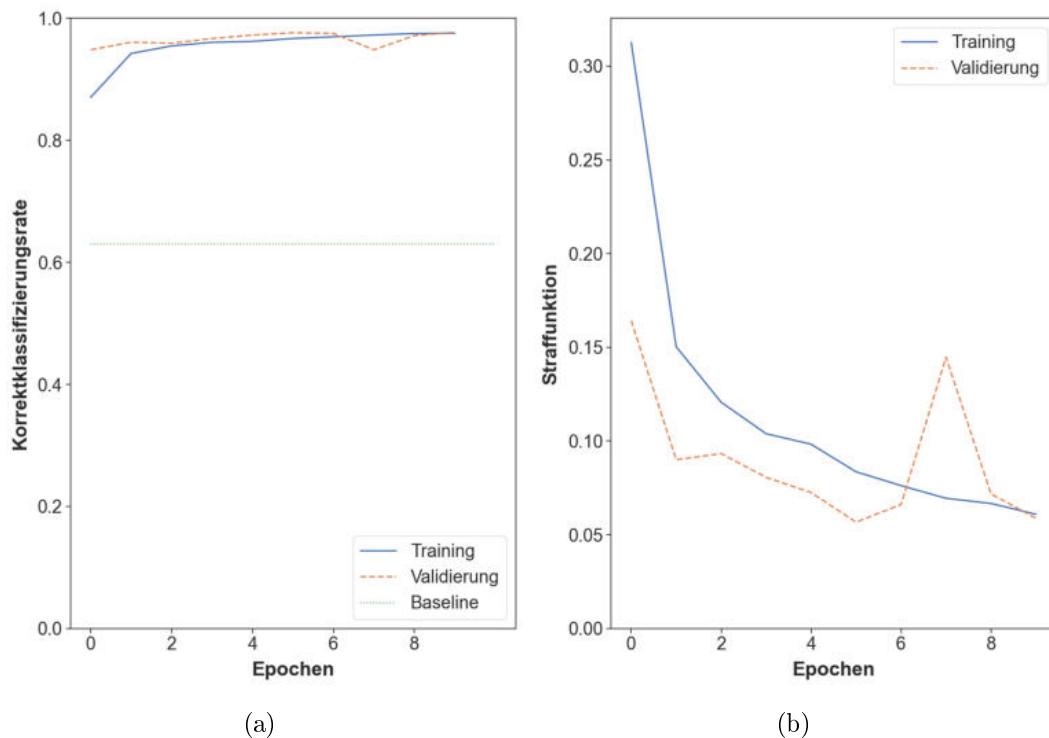


Abbildung 3.4: Korrektklassifizierungsrate (a) sowie Straffunktion (b) des ersten Modells

3 Praktische Umsetzung

Die Korrektklassifizierungsrate des Trainings-, als auch des Validierungsdatensatzes (Abbildung 3.4(a)) erreichen sehr früh nach wenigen Epochen Klassifizierungsraten von über 90 %. Auch die Konfusionsmatrix zeigt ein ähnliches Bild (siehe Abbildung 3.5).

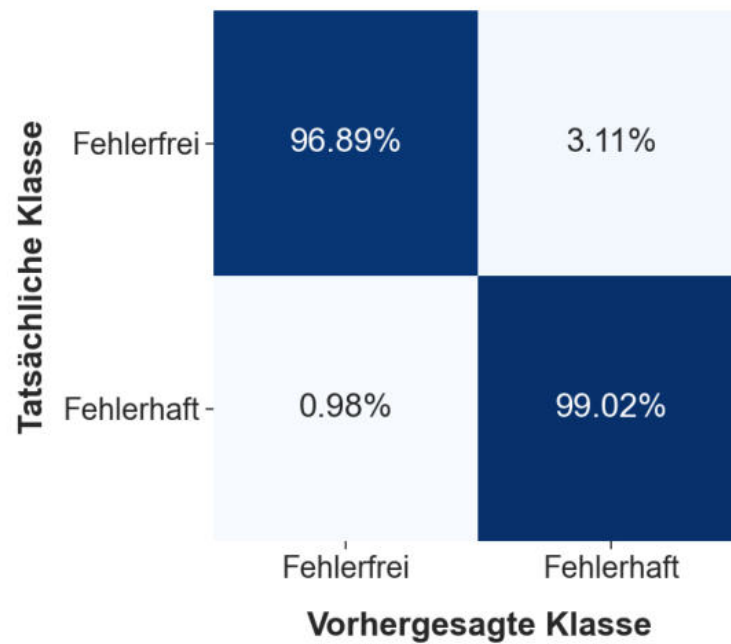


Abbildung 3.5: Konfusionsmatrix des ersten trainierten Modells

Diese sehr guten Ergebnisse, zusammen mit dem sehr kurzen Training bedürfen einer genaueren Betrachtung, insbesondere vor dem Hintergrund, dass das händische Labeling der Baujobs sich als schwierig erwiesen hat. Zum einen hatten die im Prozess auftretenden Fehler oft einen fließenden Übergang, d.h. diese haben sich langsam entwickelt und wurden erst im Verlauf des Drucks größer und deutlich erkennbar. Dadurch war es oft nötig aufeinanderfolgende Bilder mehrmals zu betrachten um festzulegen, ab wann genau ein Fehler begonnen hat zu entstehen. Aus einer einzelnen Aufnahme war die Anomalieerkennung mit bloßem Auge somit, vor allem zu Beginn der Fehlerentstehung, oft nicht eindeutig.

3.3 Erster Trainingsversuch: Overfitting

Erschwert wurde das Labeling außerdem durch die Tatsache, dass die Bildaufnahmen aus dem Inneren der Prozesskammer nur eine relativ geringe Auflösung von 1280×920 Pixel besitzen. All diese Erkenntnisse sind eher gegenläufig zu den hier erhaltenen Ergebnissen. Um zu verstehen wie das trainierte Modell zu diesen Ergebnissen gekommen ist, wurden einzelne Aufnahmen des Validierungsdatensatzes mit Hilfe des Verfahrens der Integrated Gradients (siehe Kapitel Integrated Gradients) untersucht.

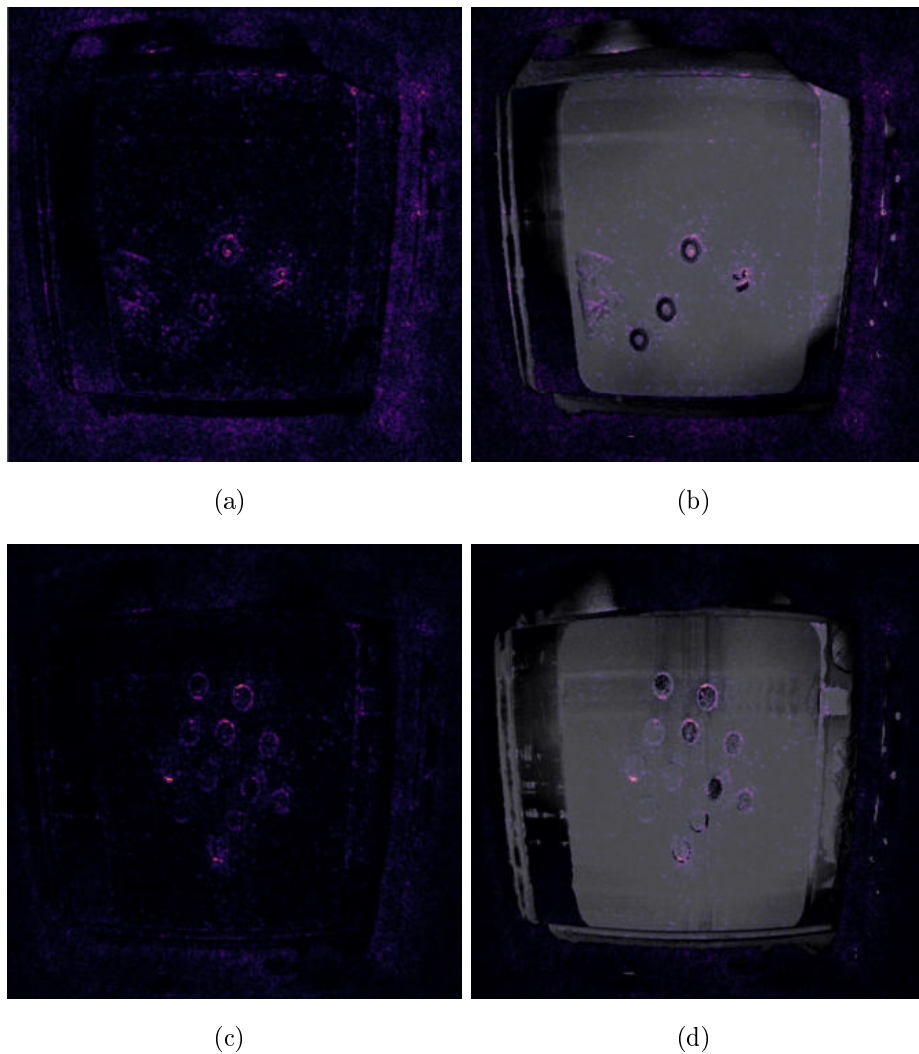


Abbildung 3.6: Durch Gradienten erstellte Masken (a) und (c), sowie mit der Maske überlagerte original Bilder (b) und (d)

3 Praktische Umsetzung

Anhand der Gradienten Masken und den überlagerten original Schichtbildern wird deutlich, dass das trainierte Modell vor allem die Pixel im Bereich der gedruckten Bauteile als wichtig erwachtet hat (Abbildung 3.6(b) und (d)). Auch der Bereich um das Pulverbett interpretiert das Modell als wichtig, wobei dieser eigentlich für die Klassifizierung überhaupt nicht relevant sein sollte. Der Deltafehler in Abbildung 3.6(b) sowie der Pulverfehler in Abbildung 3.6(d) dagegen wurden kaum beachtet. Die sehr hohe Korrektklassifizierungsrate muss somit einen anderen Grund haben.

Da die Objekte Schichtweise entlang der Z-Achse aufgebaut werden, und zwischen den einzelnen Schichten lediglich 50 μm liegen, ähneln sich aufeinanderfolgende Schichtaufnahmen sehr stark. Auf den ersten Blick ist ein direkter Unterschied zwischen zwei aufeinanderfolgenden Schichtbildern oft nicht sichtbar. Diese Eigenheit des Datensatzes begünstigt dadurch stark die Überanpassung des Neuronalen Netzes auf den Datensatz und somit dem reinen auswendig lernen. Beim näheren untersuchen des Validierungsdatensatzes wurde außerdem deutlich, dass sich Bilder stark mit denen aus dem Trainingsdatensatz ähneln. Dies erklärt, wieso das trainierte Modell den Randbereich, sowie die Bauteile sehr stark gewichtet hat, da diese beiden Bereiche die wohl größten Gemeinsamkeiten zwischen den Trainings- und dem Validierungsdatensatz liefern.

Der hier verwendete Ansatz alle Schichtaufnahmen zusammenzufassen und nach dem Zufallsprinzip in einen Trainings-, Validierungs- und Testdatensatz aufzuteilen funktioniert somit nicht. Die einzelnen Baujobs müssen somit also als zusammenhängende Sequenzen betrachtet werden.

3.4 Schrittweise Optimierung

Der erste Trainingsversuch war nicht zielführend. Deshalb wurde nun ein schrittweises Vorgehen für die Schichtbilder gewählt. Dazu wurden zunächst neue Kriterien definiert, anhand derer der Datensatz aufgeteilt wurde. Anschließend wurden die Schichtbilder selbst bearbeitet, indem diese zugeschnitten und entzerrt wurden.

3.4.1 Neuaufteilung des Datensatzes

Aufgrund der vorliegenden Erkenntnisse des ersten Trainingversuchs wurde der Datenbestand nun nach gedruckten Objekten aufgeteilt, wobei hier einige Einschränkungen gemacht wurden. Anstatt den Datensatz zufällig nach Baujobs aufzuteilen, wurde die Aufteilung händisch durchgeführt. Der Grund dafür liegt darin, dass der Datensatz nach Druckaufträgen nur einen sehr begrenzten Umfang von 56 Baujobs besitzt. Dieses Vorgehen birgt zwar die Gefahr, einen stärkeren menschlichen Bias in das Neuronale Netz mit hineinzubringen, jedoch kann so sichergestellt werden, dass jede Art von Fehler in jedem der drei Datensätze auch vorkommt. Außerdem kann auf diese Weise sichergestellt werden, dass es zwischen je zwei der drei Datensätze möglichst wenige Überschneidungen bezüglich der gedruckten Objekte gibt. Dadurch soll vermieden werden, dass das Neuronale Netz nur die Geometrien der Objekte erlernt und anhand dieser die Klassifizierung vornimmt, anstatt die eigentlich wichtigen Anomalien zu erlernen. Mit Hilfe dieser Kriterien wurde anschließend der Datensatz in drei Teile mit einem Verhältnis von etwa 60 % für den Trainingsdatensatz, sowie je 20 % für den Validierungs- und Testdatensatz aufgeteilt. Auch hier wurde erneut ein Neuronales Netz trainiert und per Parameter Tuning zu optimieren. Anschließend wurden wieder die Korrektklassifizierungsrate und Straffunktion (Abbildungen 3.7(a) und (b)), sowie die Konfusionsmatrix (Abbildung 3.8) ausgewertet.

3 Praktische Umsetzung

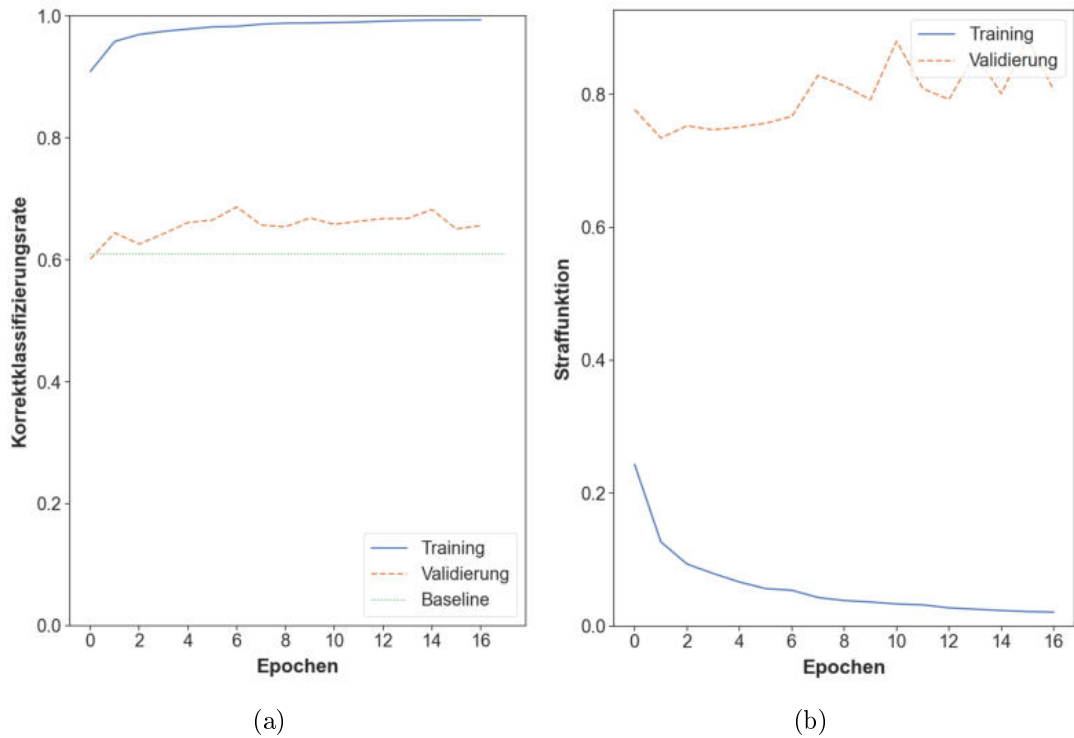


Abbildung 3.7: Korrektklassifizierungsrate (a) sowie Straffunktion (b) des zweiten Modells

Die Korrektklassifizierungsrate (Abbildungen 3.7(a)) erreicht auch bei diesem Training nach etwa sechs Epochen ihren Maximalwert auf dem Validierungsdatensatz. Auch der Verlauf der Trainingskurve deutet dabei schon auf eine sehr frühe Überanpassung, da nach etwa drei bis vier Epochen ca. 95 % der Samples im Trainingsdatensatz korrekt klassifiziert werden können. Die Korrektklassifizierungsrate des Validierungsdatensatzes liegt dabei nur leicht über der Baseline, was ebenfalls auf eine eher schlechte Generalisierungsfähigkeit des Modells deutet. Dieser Eindruck wird durch die Straffunktion in Abbildung 3.7(b) ebenfalls bestätigt, da die Differenz zwischen der Trainings- und Validierungskurve schon nach zwei bis drei Epochen deutlich größer wird.

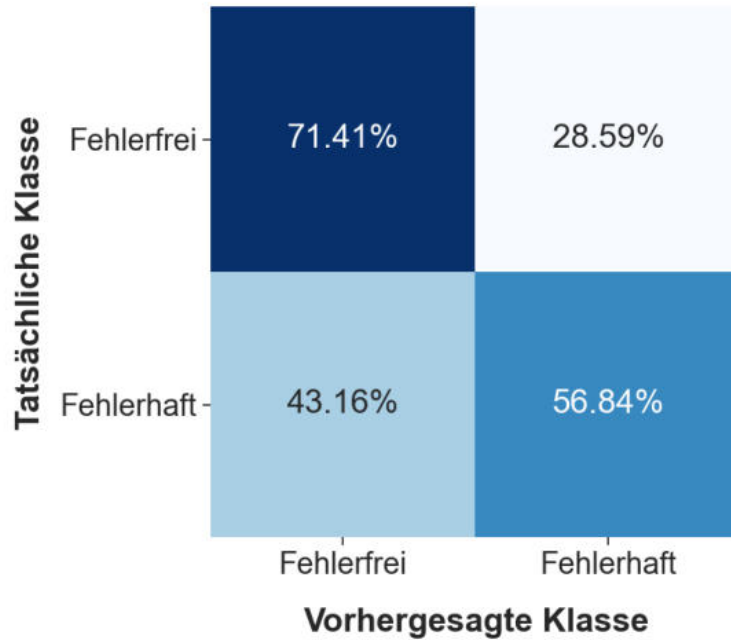


Abbildung 3.8: Konfusionsmatrix des zweiten trainierten Modells

Die in Abbildung 3.8 dargestellte Konfusionsmatrix zeigt außerdem, dass das trainierte Modell besser auf den fehlerfreien als auf den fehlerhaften Aufnahmen funktioniert. So sind etwas mehr als ein viertel aller tatsächlich fehlerfreien Aufnahmen als Fehlerhaft klassifiziert worden. Dagegen wurde annähernd die Hälfte der tatsächlich fehlerhaften Aufnahmen als fehlerfrei erkannt. Dies deutet darauf hin, dass das Neuronale Netz nicht weiß, welche Merkmale für die Fehlererkennung wichtig sind.

Untersucht wurden hier ebenfalls mittels der Gradienten Masken einige Aufnahmen des Validierungsdatensatzes (Abbildung 3.9).

3 Praktische Umsetzung

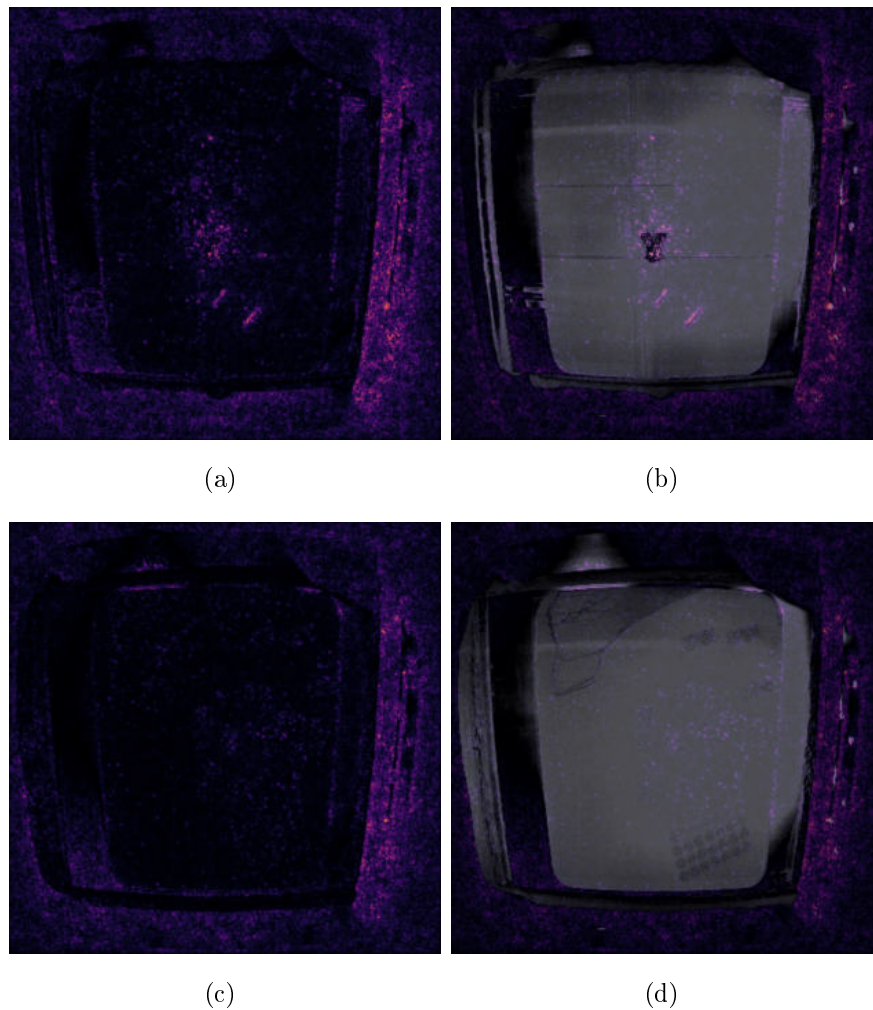


Abbildung 3.9: Durch Gradienten erstellte Masken (a) und (c), sowie mit der Maske überlagerte original Bilder (b) und (d)

Vor allem am Beispiel der Abbildung 3.9(b) wird deutlich, dass das Modell anscheinend Bauteile im Pulver relativ gut erkennen kann. Gleichzeitig werden jedoch sehr deutliche Fehler im Pulverbett, wie etwa die Streifen in Abbildung 3.9(b) oder der Deltafehler links in Abbildung 3.9(d), ignoriert. Andererseits wird auch wieder vor allem dem unwichtigen Randbereich rechts relativ viel Beachtung geschenkt (Abbildung 3.9(a)). Vermutlich liegt dies daran, dass dieser Randbereich um das Pulverbett herum in allen Schichtaufnahmen vorzufinden ist und das Neuronale Netz somit diese Bildbereiche immer versucht zu interpretieren.

Trotz dem Aufteilen des Datenbestands nach gedruckten Bauteilen und Fehlern hat das zu trainierende Neuronale Netz Probleme damit zu erkennen, welche Informationen in den vorliegenden Bildern als wichtig zu erachten sind. Sehr deutliche Fehler werden kaum beachtet und stattdessen wird eine große Gewichtung auf unwichtige Randbereiche oder Bauteile selbst gelegt. Um ein Overfitting auf den Randbereich zu vermeiden und somit auch den Informationsgehalt der Schichtaufnahmen auf das Nötigste zu verringern wurde der Datensatz weiter bearbeitet, indem allen Schichtbildern der Randbereich entfernt wurde und nur das eigentliche Pulverbett auf den Aufnahmen zu sehen ist. Problematisch ist dabei, dass die Kamera innerhalb der Prozesskammer nicht zentral über dem Pulverbett positioniert ist und außerdem leicht schief auf das Pulverbett gerichtet ist. Dadurch wurde aus dem eigentlich quadratischen Pulverbett auf den Bildern eher ein Trapez. Damit das Pulverbett dennoch extrahiert werden kann, wurden aus den Pulverbettaufnahmen mit Hilfe des Canny-Algorithmus[43] alle Kanten extrahiert. Aufgrund des Pulvers sind jedoch nicht immer die Kanten und abgerundeten Ecken der sich absenkenden Bauplattform sichtbar. Deshalb wurden zunächst die ersten 10 Schichtaufnahmen in Bilder umgewandelt, welche lediglich die Kanten der Aufnahmen zeigen. Anschließend wurden diese 10 Schichtaufnahmen überlagert und dadurch die Kanten der Bauplattform besser sichtbar. Ein selbstgeschriebener Kernel-Filter, welcher auf die Form der abgerundeten Ecken der Plattform angepasst wurde suchte anschließend die Koordinaten der vier Ecken. Damit das Neuronale Netz nur das Pulverbett zu sehen bekommt, musste aus dem trapezförmigen Pulverbett ein Rechteck oder Quadrat werden, d.h. eine Entzerrung wurde an dieser Stelle benötigt. Hierzu wurde eine fertige Funktion der OpenCV Library verwendet [44]. Diese berechnet anhand vier gegebener Ursprungskoordinaten und vier gegebener Zielkoordinaten eine Transformationsmatrix, welche anschließend auf das zu entzerrende Bild, in diesem Fall das reine Pulverbett, angewendet wird (Abbildung 3.10).

3 Praktische Umsetzung

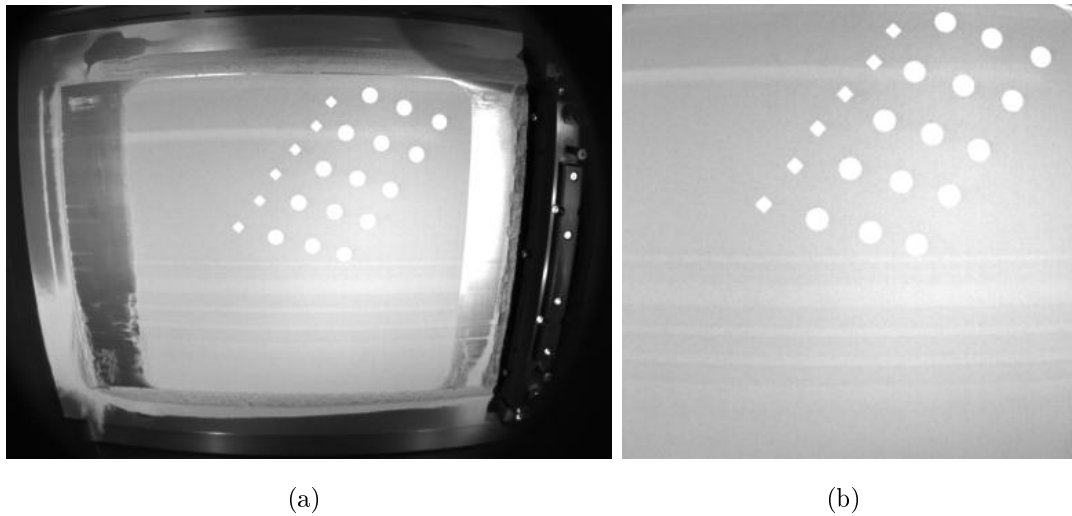


Abbildung 3.10: Original Schichtaufnahme (a) sowie die zugeschnittene und entzerrte Schichtaufnahme (b)

Bei diesem Verarbeitungsschritt wurde versucht möglichst wenig an den eigentlichen Bilddaten zu verändern, da diese mit einer Auflösung von 1280×960 Pixeln einen relativ geringen Detailgrad besitzen. Deshalb wurde beim entzerren als Zielgröße für das reine Pulverbett eine Auflösung von 768×768 Pixel gewählt. Diese Größe entspricht fast der nativen Dimension des reinen Pulverbetts im ursprünglichen Schichtbild. Angewendet wurde diese Bildverarbeitung auf allen Pulverbettaufnahmen.

3.4.2 Extrahierung des Pulverbetts

Anschließend wurde auch hier wieder ein Neuronales Netz trainiert und dessen Parameter auf eine möglichst hohe Korrektklassifizierungsrate optimiert. Die bisher festgelegte Aufteilung der Baujobs in Test, Training und Validierungsdaten wurde zur Vergleichbarkeit der einzelnen Experimente beibehalten. Ferner wurde hier die Dimension der Pulverbettaufnahmen beim einlesen durch das Modell auf eine Auflösung von 384×384 Pixel halbiert. Eine noch größere Verringerung der Bildauflösung hätte zur Folge gehabt, dass zu viele Details verloren gingen und einige Fehler im Grundrauschen des Pulverbetts untergegangen wären.

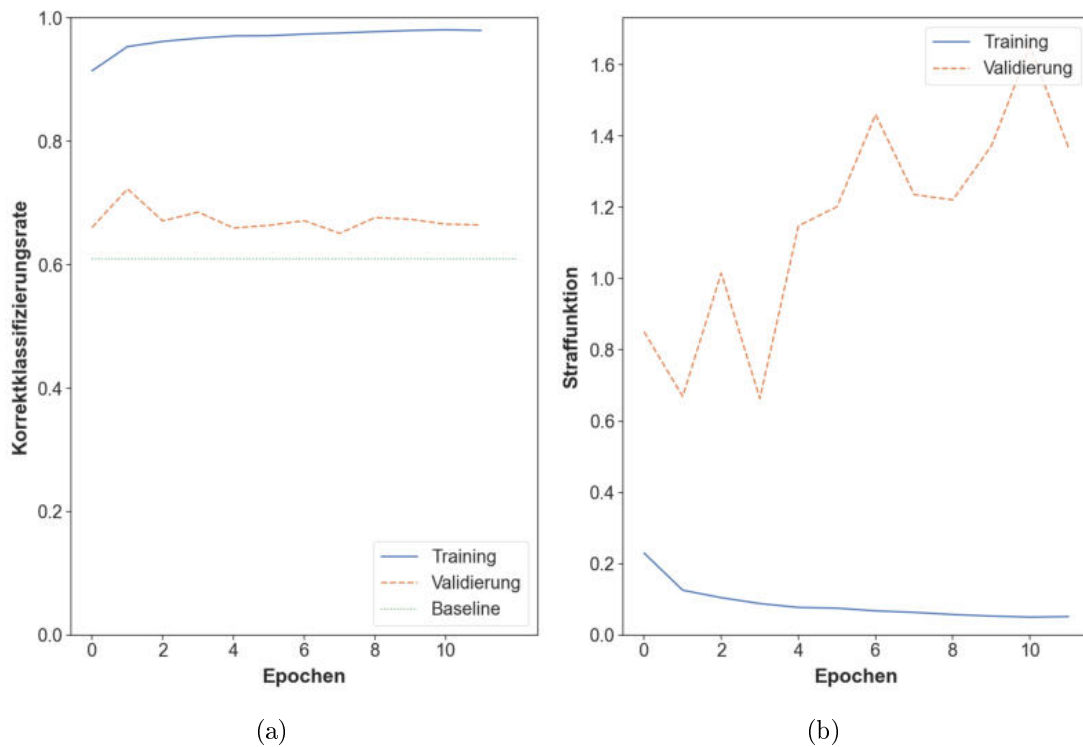


Abbildung 3.11: Korrektklassifizierungsrate (a) sowie Straffunktion (b) des dritten Modells

3 Praktische Umsetzung

Entgegen der Vermutungen, haben sich die Ergebnisse des Modells auf den extrahierten Pulverbetten kaum verbessert. Zudem zeigt sich hier wieder eine sehr früh eintretende Überanpassung des Modells, was wiederum daran sichtbar wird, dass das Modell die Trainingsdaten auswendig lernt (Abbildung 3.11(a)) und dabei kaum bessere Ergebnisse als die Baseline des Validierungsdatensatzes liefert.

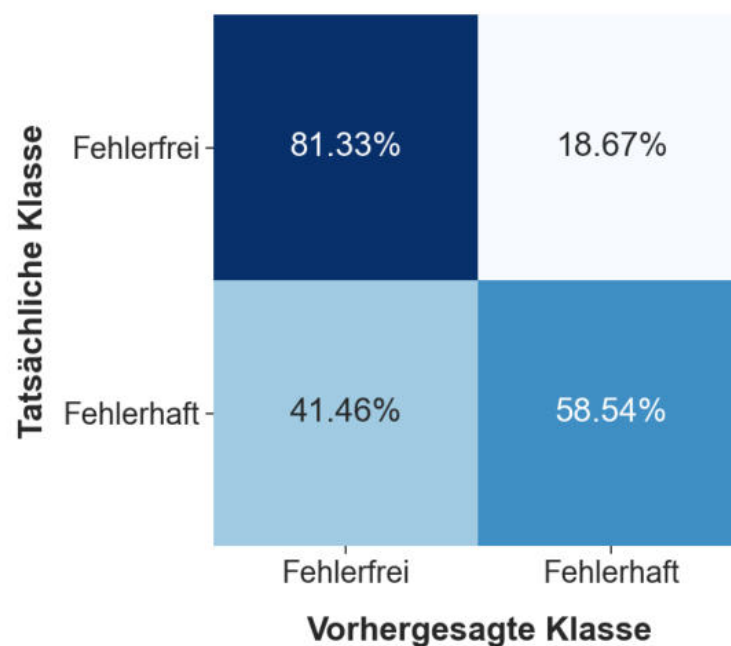


Abbildung 3.12: Konfusionsmatrix des dritten trainierten Modells

Die Konfusionsmatrix in Abbildung 3.12 zeigt ein ähnliches Bild. Die Erkennungsrate der fehlerfreien Bilder hat sich zwar gegenüber dem vorherigen Trainingsversuch um etwa 10 % verbessert, jedoch ist die Erkennungsrate auf den fehlerhaften Schichtbildern gleich geblieben. Wie zuvor deutet sich an, dass das Neuronale Netz nicht erkennt, welche Merkmale zur Identifizierung von Fehlern wichtig sind.

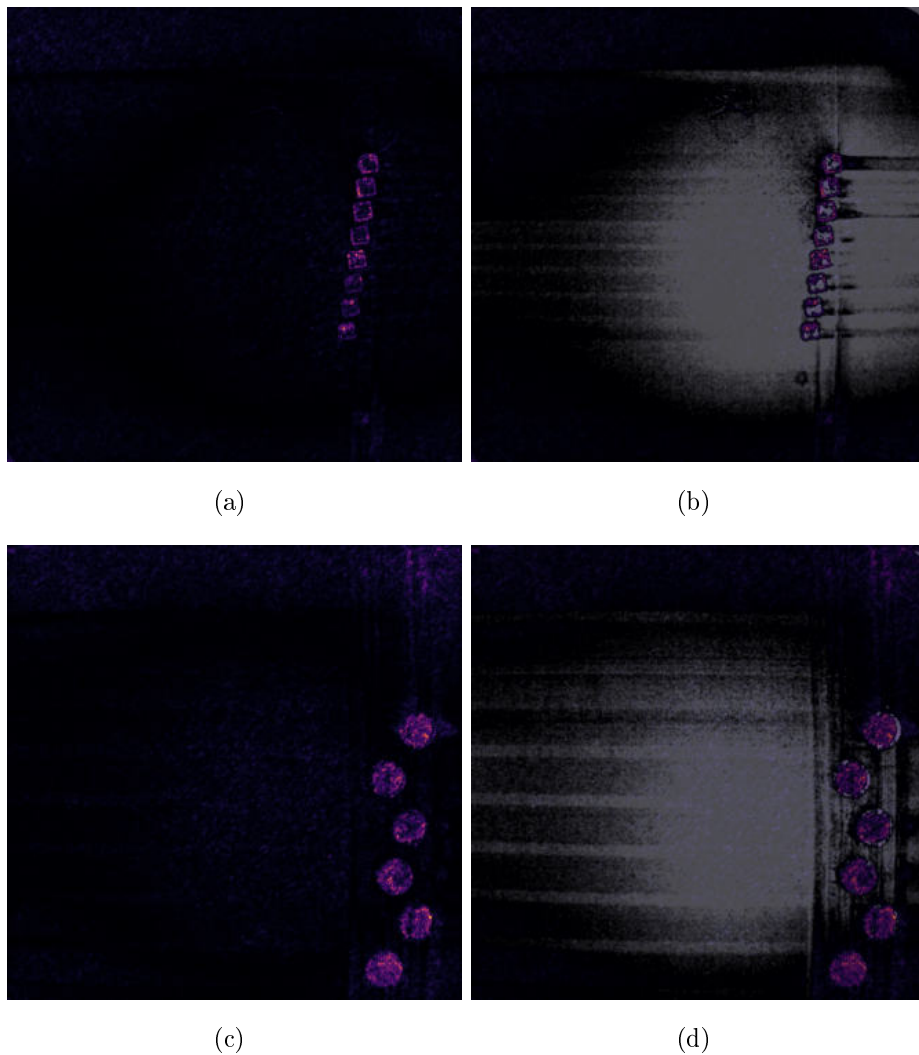


Abbildung 3.13: Durch Gradienten erstellte Masken (a) und (c), sowie mit der Maske überlagerte original Bilder (b) und (d)

Die Gradienten Masken in Abbildung 3.13 bestätigen diese Vermutung ebenfalls. So werden in beiden Fällen die Streifen im Pulverbett komplett ignoriert. Wobei auch hier interessant ist, dass das Modell die Bauteile an sich ziemlich gut erkennen kann. Vermutlich liegt dies daran, dass die Bauteile relativ gut vom restlichen Pulverbett abgegrenzt werden können. Dagegen haben die auftretenden Fehler oft sehr fließende Übergänge (vgl. Kapitel 2.2).

3 Praktische Umsetzung

Für weitere Erkenntnisse wurden die falsch klassifizierte Schichtbilder des Validierungsdatensatzes betrachtet. Hier hat sich gezeigt, dass das trainierte Modell auch sehr offensichtliche Fehler übersehen hat und es sich keineswegs um Grenzfälle handelt, die auch für den Menschen schwer zu unterscheiden sind. Einige Beispiele zu falsch klassifizierten Aufnahmen sind in Abbildung 3.14 zu sehen.

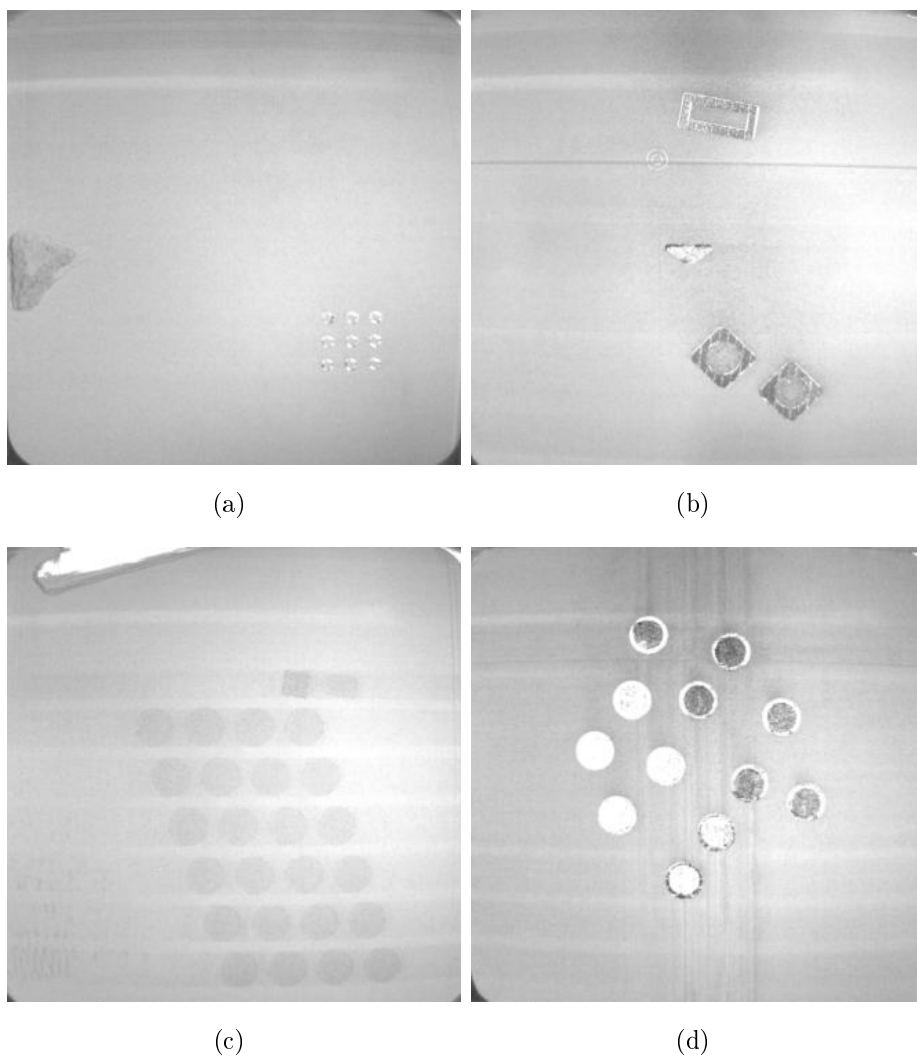


Abbildung 3.14: Beispiele für falsch klassifizierte Schichtaufnahmen des trainierten Modells

3.4.3 Entfernung der ersten Schichtbilder

Für einen weiteren Versuch wurde der Datensatz erneut bearbeitet. Am Anfang eines jeden Baujobs ist die Grundplatte aus Metall für etwa 10 bis 20 Schichten zu sehen, da in dieser Zeit das Pulverbett noch aufgebaut wird (Abbildung 3.15).

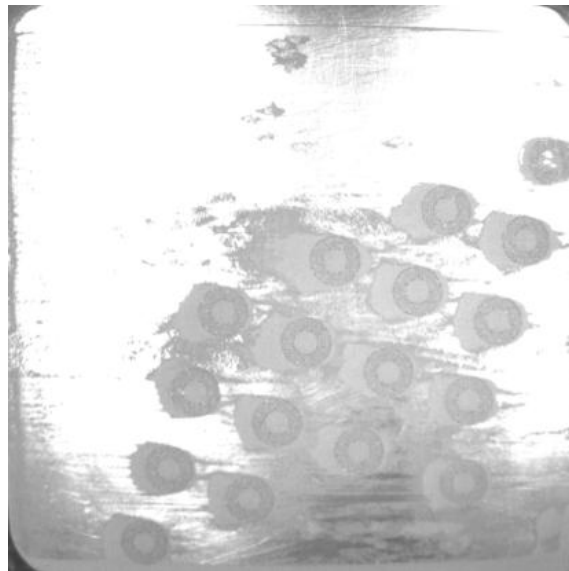


Abbildung 3.15: Im Aufbau befindliches Pulverbett am Anfang eines 3D-Drucks

Während dem Labeln der Schichtaufnahmen wurden diese Bilder der Klasse der fehlerfreien Aufnahmen zugewiesen, da es sich hierbei nicht um einen Fehler handelt. Problematisch könnte hier jedoch sein, dass das CNN die Schichtaufnahmen einzeln und unabhängig voneinander klassifiziert. Ohne diesen Kontext, d.h. zu wissen, dass das Pulverbett am Anfang sich noch im Aufbau befindet, könnte die Klassifizierung für das Modell zusätzlich erschwert werden. Deshalb wurde versuchsweise das Training ohne diese Schichtaufnahmen durchgeführt.

3 Praktische Umsetzung

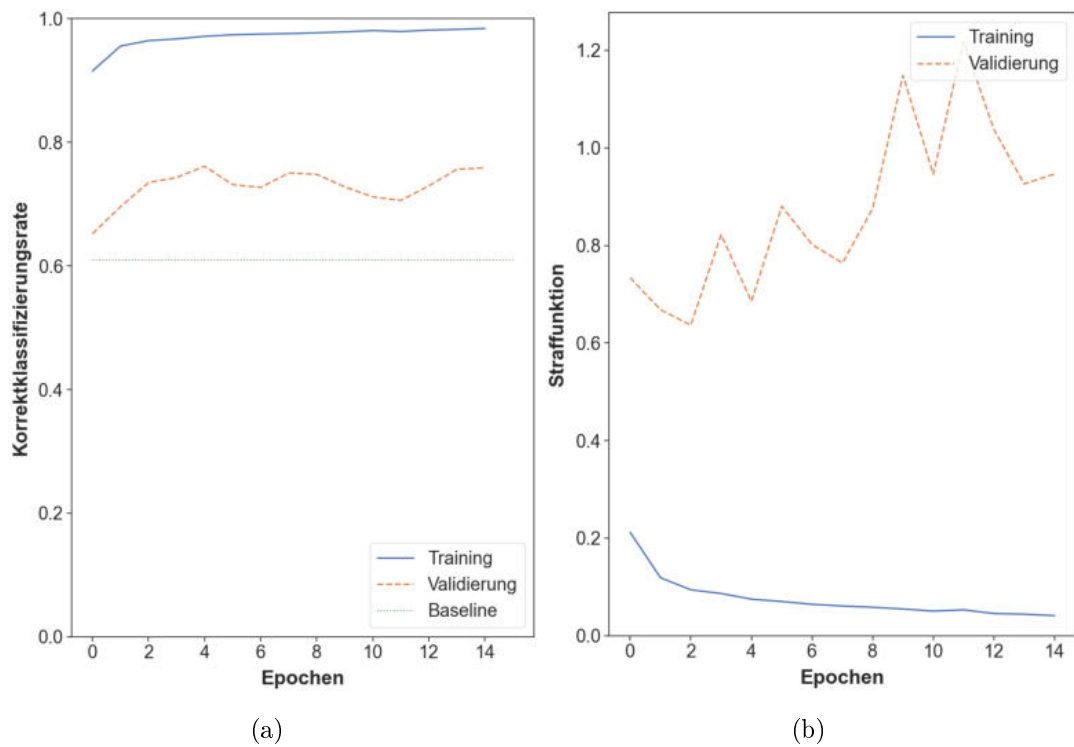


Abbildung 3.16: Korrektklassifizierungsrate (a) sowie Straffunktion (b) des vierten Modells

Die Korrektklassifizierungsrate in Abbildung 3.16(a) weist eine leichte Verbesserung gegenüber des vorherigen Trainings (Abbildung 3.11(a)) auf. Dennoch kommt es hier relativ schnell zu einer Überanpassung.

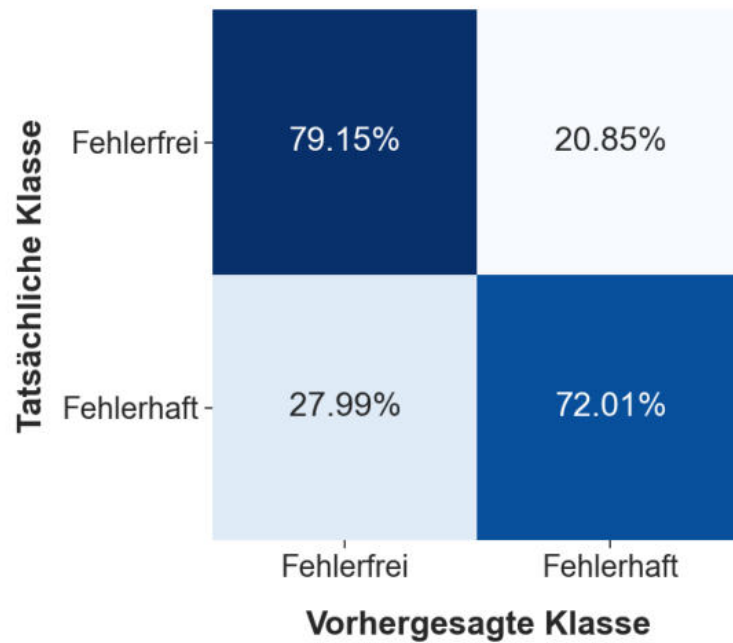


Abbildung 3.17: Konfusionsmatrix des vierten trainierten Modells

Ein etwas anderes Bild liefert hier die Konfusionsmatrix (Abbildung 3.17). Laut dieser blieb die Rate der korrekt klassifizierten fehlerfreien Aufnahmen gegenüber dem vorherigen Versuch annähernd konstant, allerdings hat sich gleichzeitig die Rate der korrekt klassifizierten fehlerhaften Schichtbilder verbessert. Dennoch ist die Rate der falsch klassifizierten fehlerhaften Bilder mit fast einem Drittel immer noch relativ hoch.

3 Praktische Umsetzung

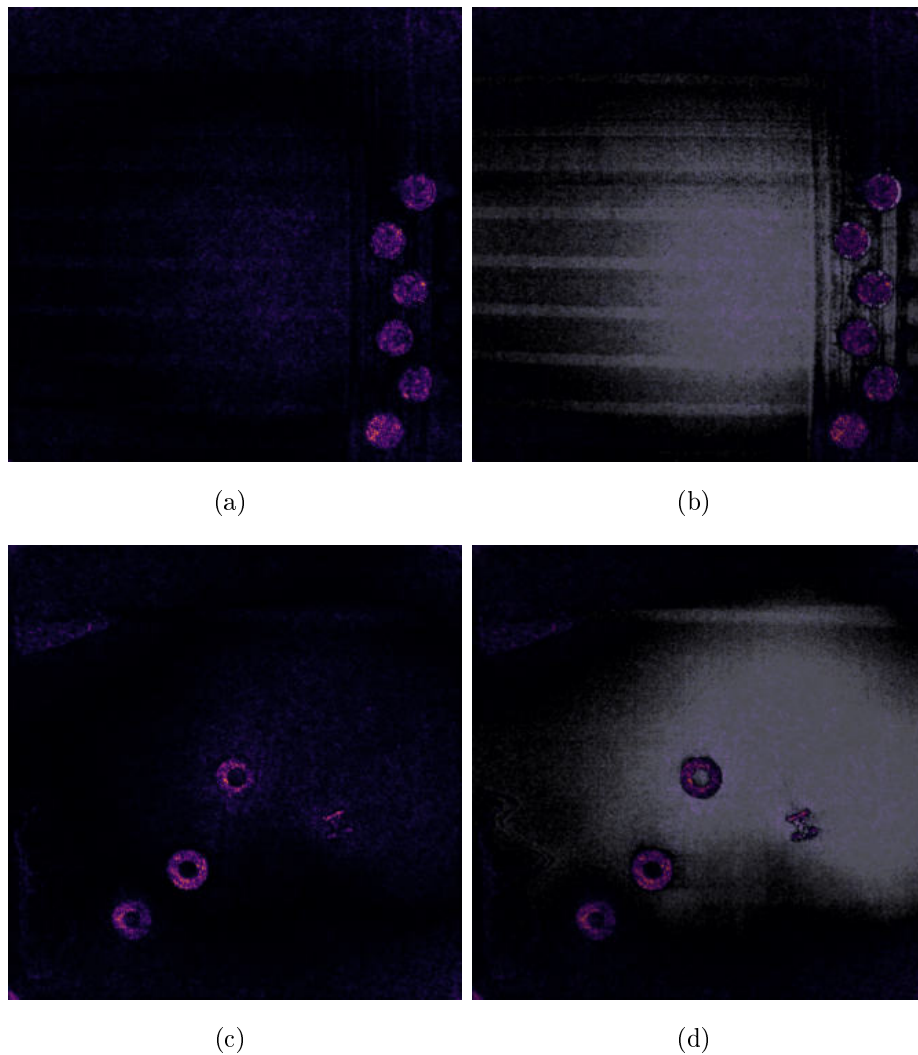


Abbildung 3.18: Durch Gradienten erstellte Masken (a) und (c), sowie mit der Maske überlagerte original Bilder (b) und (d)

Zudem zeigen die Gradienten Masken wieder, dass das Neuronale Netz vor allem auf Bauteile geachtet hat und die eigentlichen Fehler im Pulverbett selbst weitestgehend ignoriert wurden (Abbildung 3.18). Es gab somit zwar eine Verbesserung des Modells, jedoch führte auch diese nicht zu dem gewünschten Ergebnis.

3.4.4 Fazit des bisherigen Trainings

Zusammengefasst lässt sich somit sagen, dass der bisherige Ansatz, das Pulverbett als ganzes zu betrachten nicht zielführend war. Trotz intensiver Datenaufbereitung und Augmentierung kann das Neuronale Netz Fehler im Pulverbett nicht erkennen. Dies gilt sowohl für sehr offensichtliche Fehler als auch für solche, die eher subtiler Natur sind. Stattdessen konzentriert sich das Neuronale Netz lediglich auf die Bauteile selbst und deren Geometrien. Dies liegt vermutlich daran, dass das Modell hier die höchsten Kontraste auffindet und somit am einfachsten Muster erkennen kann. Das andere Problem des Trainings ist das durchgehend auftretende sehr starke Overfitting. Dies hat vermutlich zwei Hauptgründe. Zum einen besitzt der Datensatz nur einen relativ geringen Umfang mit 56 unterschiedlichen, teilweise jedoch auch doppelt vorliegenden, Druckaufträgen. Zum anderen kommt hinzu, dass zwischen zwei aufeinanderfolgenden Ebenen lediglich 50 μm liegen, was zur Folge hat, dass sich viele Schichtbilder stark ähneln. Dies gilt umso mehr, wenn Geometrien wie Zylinder oder Würfel gedruckt werden. Zuletzt kommt hinzu, dass die abgespeicherten Schichtaufnahmen eine relativ geringe Auflösung besitzen, wodurch den Schichtbildern ein gewisser Detailgrad fehlt. Gleichzeitig verlieren die Aufnahmen durch herunterskalieren weitere Details die für das Training wichtig sind, wodurch subtile Fehler im Grundrauschen des Pulverbetts untergehen. Um diese Einschränkungen zu umgehen, bedarf es somit einer anderen Methodik.

3.5 Ansatz der Bildsegmentierung

Das Hauptproblem der bisherigen Versuche lag darin, dass das CNN nicht auf die für Fehler Detektion notwendigen Merkmale achtet. Dies liegt vermutlich auch an dem geringen Umfang des Datensatzes, da zusätzlich unterschiedliche Geometrien gedruckt wurden, gleichzeitig jedoch auch die Fehler an sich eine eigene Varianz besitzen und nicht immer gleich aussehen. Vergleichbar mit dieser Problematik ist die Erkennung von Rissen in Beton. Hier wurde das Verfahren der Bildsegmentierung, wie in [45, 46, 47, 48] beschrieben, bereits erfolgreich angewendet. Bei diesem Verfahren wird das Bild in einzelne Segmente unterteilt und anschließend darauf ein CNN trainiert. Dadurch kann das CNN gezielter trainiert werden, da dem Modell jeweils nur ein bestimmter Ausschnitt mit ganz bestimmten Merkmalen gezeigt wird. Zu den anderen Vorteilen dieser Methodik gehören außerdem die Vergrößerung des Datensatzes sowie das Wegfallen des herunterskalierens der Bilder, da die einzelnen Segmente zwangsweise kleiner sind, als das ganze Pulverbett. Das Prinzip der Segmentierung wird in Abbildung 3.19 verdeutlicht.

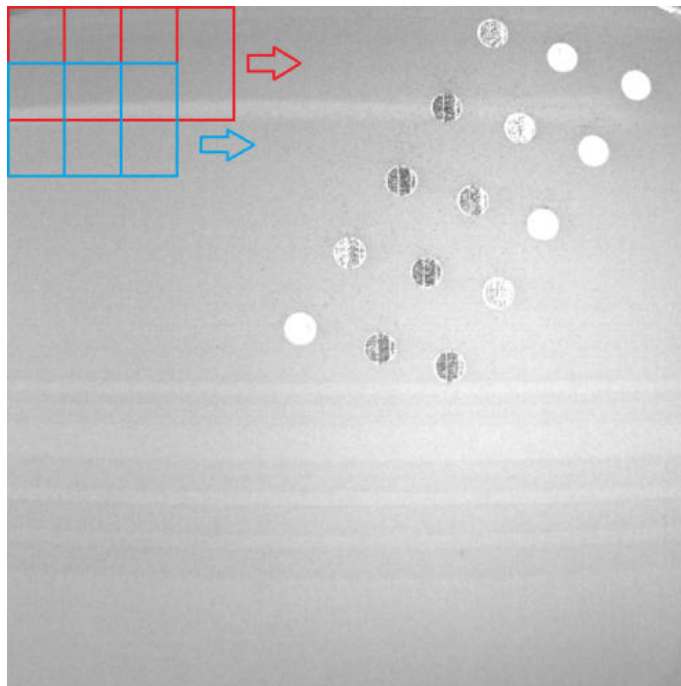


Abbildung 3.19: Prinzipielle Funktionsweise der Segmentierung anhand eines Pulverbetts

3.5.1 Segmentweise Datenaufbereitung

Um die Segmentierung umzusetzen, wurde ein Tool benötigt, welches die Schichtbilder einliest, zerschneidet, labelt und anschließend abspeichert. Hierzu wurde ein Python Tool geschrieben, welches zunächst eine Schichtaufnahme einliest und darstellt. In dieser Darstellung kann eine oder mehrere Regionen ausgewählt, und anschließend mit einem Button gelabelt werden (Abbildung 3.20). Die so ausgewählten Felder wurden daraufhin für ein erneutes Labeling vergrößert dargestellt um falsche Label zu vermeiden. Diese falschen Label können insbesondere an den Randbereichen entstehen. Wie in Abbildung 3.19 dargestellt wird, rastert das Tool die einzelnen Felder über das Schichtbild. Wenn sich nun ein Feld mit einem ausgewählten Bereich überschneidet, erhält dieses Feld das vom Benutzer vergebene Label. Alle restlichen Felder, welche keinerlei Überschneidungen aufweisen, erhalten das Label Fehlerfrei. Insbesondere in den Randbereichen kann es jedoch passieren, dass das Feld sich zwar mit einer markierten Region überschneidet, jedoch in dem so entstehenden Feld nichts von dem markierten Fehler zu sehen ist. Deshalb ist ein nachträgliches Betrachten und Prüfen der Felder erforderlich.

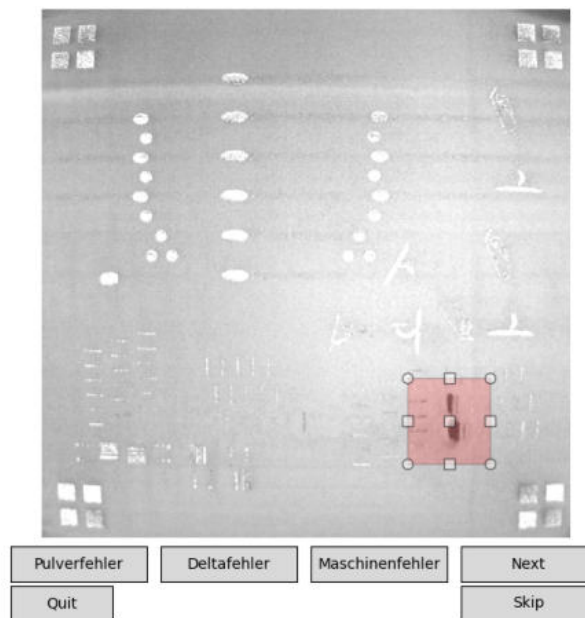
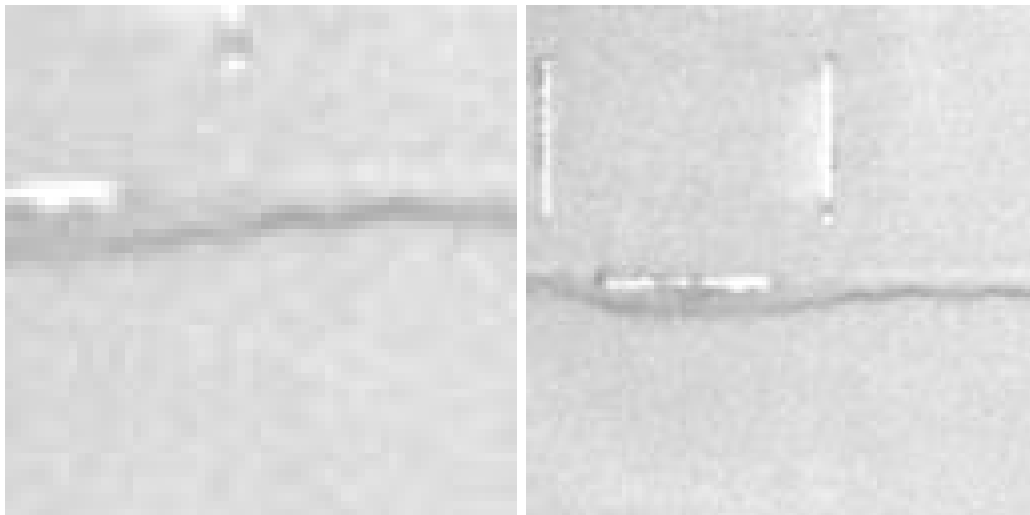


Abbildung 3.20: Das entworfene Python Tool in Version 1 zum Labeling der Bilddaten

3 Praktische Umsetzung

Als Parameter für die Segmentierung wurde eine Feldgröße von 128×128 Pixel gewählt. Kleinere Felder hätten einen deutlichen Mehraufwand für das Labeling bedeutet. Der Hauptgrund für diese gewählte Feldgröße liegt jedoch in der geringen Auflösung der Pulverbettaufnahmen. Bei kleineren Feldern sind fast nur die einzelnen Pixel sichtbar und Objekte oder Muster im Pulverbett kaum noch von einem rauschen unterscheidbar (Abbildung 3.21(a)). Als Augmentierung wurde für den Trainingsdatensatz eine Überlappung zwischen zwei aufeinanderfolgenden Segmenten von 50 % gewählt. Dadurch kann ein bestimmtes Merkmal in einem Segment mittig auftauchen und in den anderen eher am Rand oder auch teilweise zerschnitten. So kann der Umfang des Datensatzes zusätzlich erweitert werden. Für den Validierungs- und Testdatensatz wurde keine Überlappung verwendet, damit ein Modell den gleichen Fehler nicht mehrfach klassifizieren kann. Dies könnte sonst das Ergebnis der Klassifizierung verfälschen.



(a)

(b)

Abbildung 3.21: Vergleich der Segmentgröße von 64×64 Pixel (a) und einem Ausschnitt von 128×128 Pixel an der gleichen Position (b)

3.5 *Ansatz der Bildsegmentierung*

Die bisher verwendete Aufteilung des Datensatzes nach gedruckten Objekten und Art von Fehlern wurde beibehalten um eine Vergleichbarkeit mit den vorherigen Ergebnissen zu gewährleisten. Jedoch kommt hier eine weitere Einschränkung aufgrund des stark auftretenden Overfittings hinzu (siehe: Fazit des bisherigen Trainings). Um das Overfitting möglichst weit einzugrenzen, wurde der bisherige aufgeteilte Datensatz in seiner Masse ausgedünnt. Dazu wurden aus den einzelnen Baujobs jeweils nur so viele Bilder entnommen, dass ein deutlicher Unterschied zwischen den einzelnen Schichten sichtbar ist. Effektiv bedeutet dies, dass die vorliegenden Bauaufträge jeweils nur noch aus etwa 5 bis 40 Schichtbildern bestehen. Um einen möglichst klaren Datensatz zu bekommen wurden außerdem die Schichtbilder zu Beginn eines jeden Baujobs nicht verwendet, da sich in diesen das Pulverbett noch im Aufbau befindet. Des Weiteren wurden Segmente, welche nicht eindeutig einer Klasse zuweisbar waren, nicht berücksichtigt. Sollte etwa während des Labelings auf einem extrahierten Segment ein nicht eindeutiger Fehler zu sehen sein, wurde dieses Feld verworfen. Bei anschließend folgenden Labeling fiel dabei schon nach einigen Schichtaufnahmen auf, dass es hier wieder zu einer sehr starken Unausgewogenheit kam, da die auftretenden Fehler meist lokal begrenzt waren und die restlichen Ausschnitte des Bildes dagegen fehlerfreie Situationen zeigten. Dies erforderte eine weitere Anpassung des entwickelten Labeling Programms sowie einer Neubewertung der ursprünglich festgelegten Kategorien da schon diese sehr unausgewogen waren. Hinzu kommt, dass durch die Segmentierung nun nicht mehr Fehler als ganzes betrachtet werden, sondern nur noch deren Ausschnitte.

3 Praktische Umsetzung

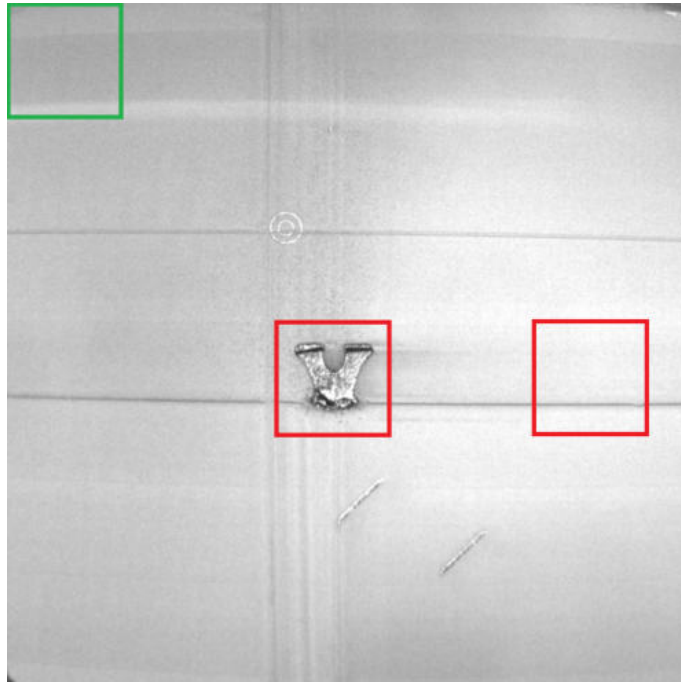


Abbildung 3.22: Das hier gezeigte Schichtbild als ganzes stellt einen Bauteilfehler dar. Jedoch zeigen die einzelnen Felder teilweise fehlerfreie Ausschnitte (grün, links oben), einen Bauteilfehler (rot, mitte) oder einen Pulverfehler (rot, rechts mitte).

Diese Problematik wird an Abbildung 3.22 verdeutlicht. Als Ganzes genommen stellt die Schicht einen Bauteilfehler dar. Werden jedoch einzelne Segmente betrachtet, wird deutlich, dass der Bauteilfehler zwar der Ursprung des Fehler ist, jedoch andere Ausschnitte reine Pulverfehler aufzeigen, aus denen nicht ersichtlich ist, dass diese aus einem Bauteilfehler hervorgehen. Somit führt die Unterteilung der einzelnen Fehlertypen zu einem deutlich größerem Mehraufwand des Labelings, bei gleichzeitig erhöhtem Schwierigkeitsgrad. Um diese Problematik zu vereinfachen, wurden die einzelnen Fehlerklassen zusammengefasst.

Aus den bisherigen Experimenten ging außerdem hervor, dass die Neuronalen Netze besonders Bauteile vom Pulverbett gut unterscheiden konnte. Deshalb wurde eine eigene Klasse Bauteile definiert, welche lediglich Bauteile darstellt. Zuletzt wurde eine Klasse Pulver festgelegt, welche weder Bauteile, noch Fehler aufwies und lediglich Pulver aufzeigte, dass gut verteilt wurde. Die so festgelegten Kriterien sind nochmals in Tabelle 3.2 dargestellt.

Tabelle 3.2: Festgelegte Label und ihre Definitionen für das Labeling Tool V2

Label Nummer	Label Klasse	Definition
0	Pulver (Fehlerfrei)	Fehlerfreies Pulver
1	Bauteil	Fehlerfreies Bauteil oder Teile dessen
2	Fehler	Alle bisher definierten Fehler

Auf Grundlage dieser Kriterien wurde das bereits entwickelte Labeling Programm angepasst, wie in Abbildung 3.23 zu sehen ist.

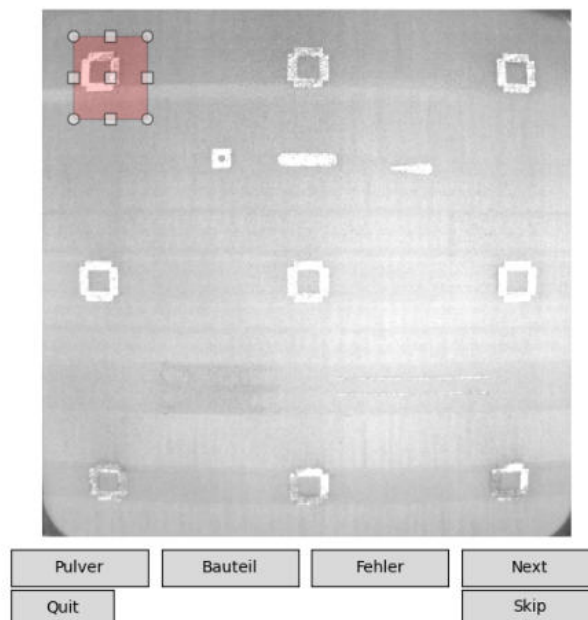


Abbildung 3.23: Das entworfene Python Tool in Version 2 zum Labeling der Bilddaten

3 Praktische Umsetzung

Als Datengrundlage wurde hier wieder der gleiche ausgedünnte und nach Bauteilen bzw. Fehlern aufgeteilte Datensatz verwendet. Um nur klare Samples für das Training zu erhalten, wurden ausschließlich Felder abgespeichert, welche sich mit markierten Regionen überlappen. Alle anderen Segmente wurden verworfen. Im Falle, dass Bauteile (siehe Abbildung 3.23), mehrfach gedruckt wurden, wurden nicht alle markiert. Außerdem wurden auch hier Segmente, welche nicht eindeutige Situationen darstellten verworfen. Dazu zählen auch solche Felder, welche ein Bauteil und einen Fehler darstellen oder nur einen sehr leichten Fehler aufweisen. Das Neuronale Netz soll während des Trainings nicht durch solche Grenzfälle falsch antrainiert werden. Für die Trainingsdaten wurde eine Überlappung zwischen zwei Segmenten von 50 % gewählt und bei dem Test- und Validierungsdatensatz wurde stattdessen keine Überlappung angewendet. Der so entstandene Trainingsdatensatz besteht aus 6858 Bildern, wovon 1884 das Label Pulver, 2475 das Label Bauteil und 2499 das Label Fehler besitzen. Ein ähnliches Bild zeigen die Validierungs- und Testdaten auf. Auch diese haben ein ähnliches Verhältnis über alle Klassen hinweg. Als Augmentierungsverfahren wurden die Trainingsdaten zufällig entlang der horizontalen und vertikalen Achsen gespiegelt. Auf den Validierungs- und Testdaten wurde keine Augmentierung angewendet.

3.5.2 Training mit Bildsegmenten

Mit den so erstellten Datensätzen wurden erneut Modelle trainiert und auf eine hohe Korrektklassifizierungsrate optimiert (Tabelle 3.3).

Tabelle 3.3: Architektur des finalen Modells (empirisch ermittelt nach gängiger grundlegender Struktur)

Layer		Anzahl Filter	Größe Filter	Stride	Padding	Größe am Ausgang
Bildeingang		-	-	-	-	128 x 128 x 3
Convolutional Layer	Convolution	32	3 x 3	1 x 1	valid	128 x 128 x 32
	ReLU	-	-	-	-	
BatchNormalization		-	-	-	-	128 x 128 x 32
Pooling Layer	MaxPooling	1	2 x 2	2 x 2	valid	64 x 64 x 32
Convolutional Layer	Convolution	64	3 x 3	1 x 1	valid	64 x 64 x 64
	ReLU	-	-	-	-	
BatchNormalization		-	-	-	-	64 x 64 x 64
Pooling Layer	MaxPooling	1	2 x 2	2 x 2	valid	32 x 32 x 64
Dropout Layer	Dropout p=0.10	-	-	-	-	32 x 32 x 64
Convolutional Layer	Convolution	128	3 x 3	1 x 1	valid	32 x 32 x 128
	ReLU	-	-	-	-	
BatchNormalization		-	-	-	-	32 x 32 x 128
Pooling Layer	MaxPooling	1	2 x 2	2 x 2	valid	16 x 16 x 128
Dropout Layer	Dropout p=0.20	-	-	-	-	16 x 16 x 128
Convolutional Layer	Convolution	256	3 x 3	1 x 1	valid	16 x 16 x 256
	ReLU	-	-	-	-	
BatchNormalization		-	-	-	-	16 x 16 x 256
Pooling Layer	MaxPooling	1	2 x 2	2 x 2	valid	8 x 8 x 256
Dropout Layer	Dropout p=0.30	-	-	-	-	8 x 8 x 256
Convolutional Layer	Convolution	256	3 x 3	1 x 1	valid	8 x 8 x 256
	ReLU	-	-	-	-	
BatchNormalization		-	-	-	-	8 x 8 x 256
Pooling Layer	MaxPooling	1	2 x 2	2 x 2	valid	4 x 4 x 256
Dropout Layer	Dropout p=0.30	-	-	-	-	4 x 4 x 256
Flatten		-	-	-	-	1 x 4096
Fully Connected Layer	512 Units	-	-	-	-	1x 512
	ReLU	-	-	-	-	
Dropout Layer	Dropout p=0.30	-	-	-	-	1 x 512
Softmax Layer	3 Units	-	-	-	-	1 x 3
	Softmax	-	-	-	-	

3 Praktische Umsetzung

Auch hier wurden wieder die Kurvenverläufe der Korrektklassifizierungsrate (Abbildung 3.24(a)) sowie der Straffunktion (Abbildung 3.24(b)) grafisch dargestellt.

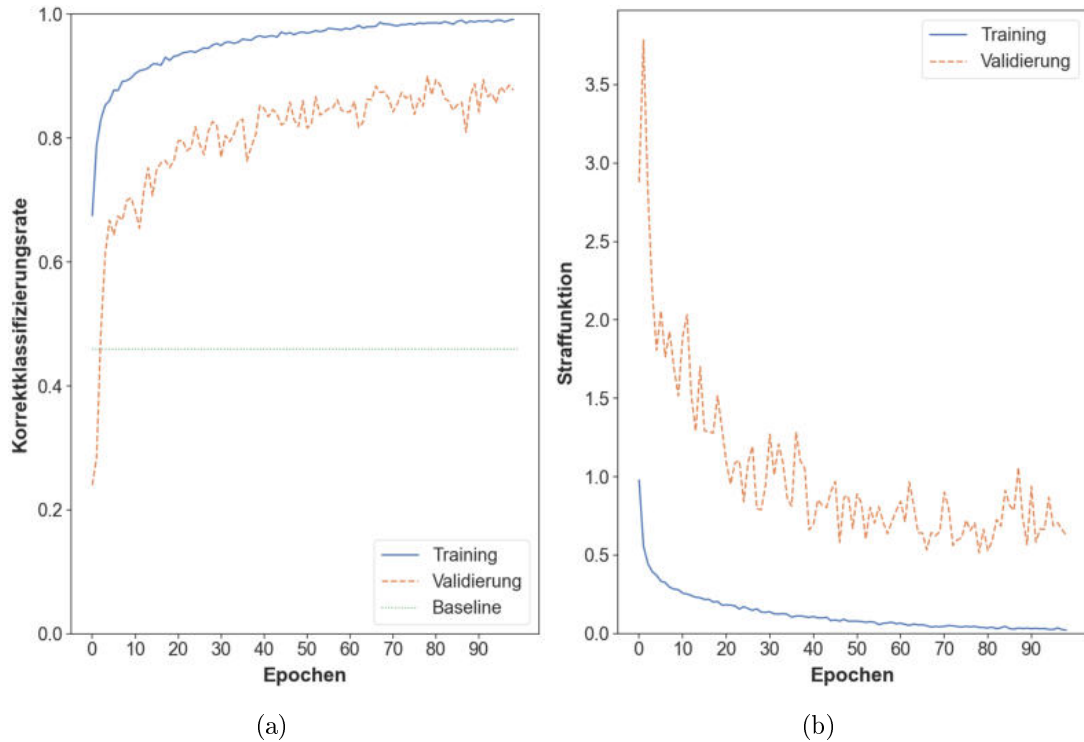


Abbildung 3.24: Korrektklassifizierungsrate (a) sowie Straffunktion (b) des segmentierten Modells auf den Validierungsdaten

Auffällig ist beim Betrachten der Graphen in Abbildung 3.24, dass die Verläufe des Validierungsdatensatzes bei diesem Modell sehr viel stärker oszillieren und gleichzeitig das Training deutlich länger anhält, ohne dass es zu einer Überanpassung kommt. So hat dieses Modell erst nach etwa 80 Epochen die höchste Korrektklassifizierungsrate erreicht. Die starke Oszillation auf den Validierungsdaten kommt möglicherweise durch den verwendeten Dropout zu Stande. Dieser wird, wie in Tabelle 3.3 ersichtlich, sowohl nach dem Fully Connected Layer, als auch nach den MaxPooling Layern zwischen den Convolutional Layern eingesetzt. Durch den Einsatz von Dropout nach MaxPooling Layern werden einige der Variablen beim Training zufällig entfernt. Dies kann im Grunde als Hinzufügen von Rauschen betrachtet werden [38, 39].

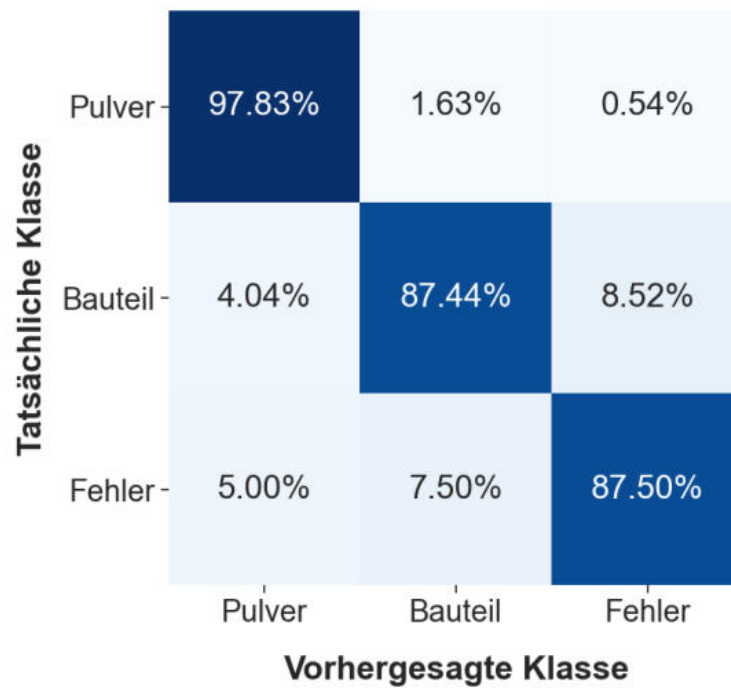


Abbildung 3.25: Konfusionsmatrix für den Validierungsdatensatz des auf Segmenten trainierten Modells

Die Konfusionsmatrix in Abbildung 3.25 zeigt auf den Validierungsdaten sehr gute Ergebnisse für die Klasse der Pulver. Die Erkennungsrate auf den Bauteilen und Fehlern sind dagegen etwas schlechter aber dennoch auf relativ hohem Niveau.

3 Praktische Umsetzung

Anschließend wurden hier wieder einige der falsch klassifizierten Segmente der Validierungsdaten betrachtet. Dabei ist zum Einen aufgefallen, dass das Modell nach wie vor Probleme damit hat, sehr feine Strukturen im Pulver zu erkennen (Abbildung 3.26).

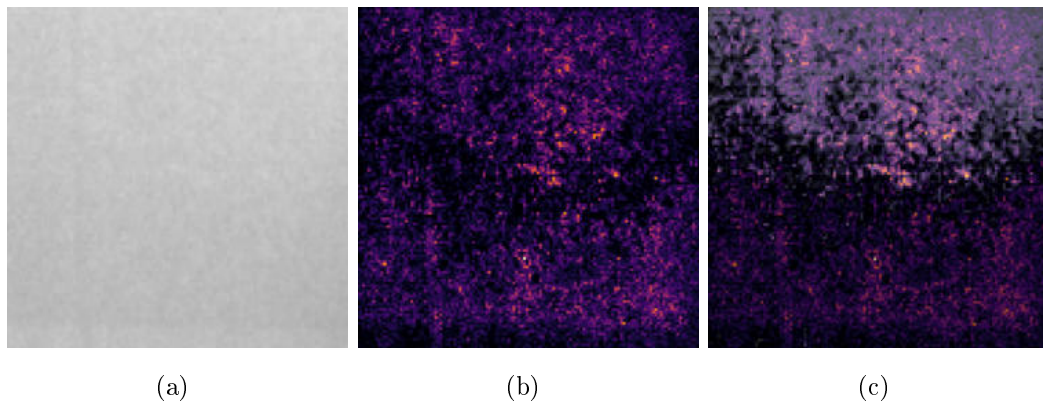


Abbildung 3.26: Original Segment (a) mit einem Fehler links im Bild, Gradienten Maske (b) sowie mit der Maske überlagertes original Bild (c). Klassifizierung erfolgte als Pulver.

Andererseits scheint das Modell auch einige der Bauteile nicht zu erkennen (Abbildung 3.27 und 3.28). In beiden Fällen erfolgte eine Klassifizierung der Bilder als Fehler.

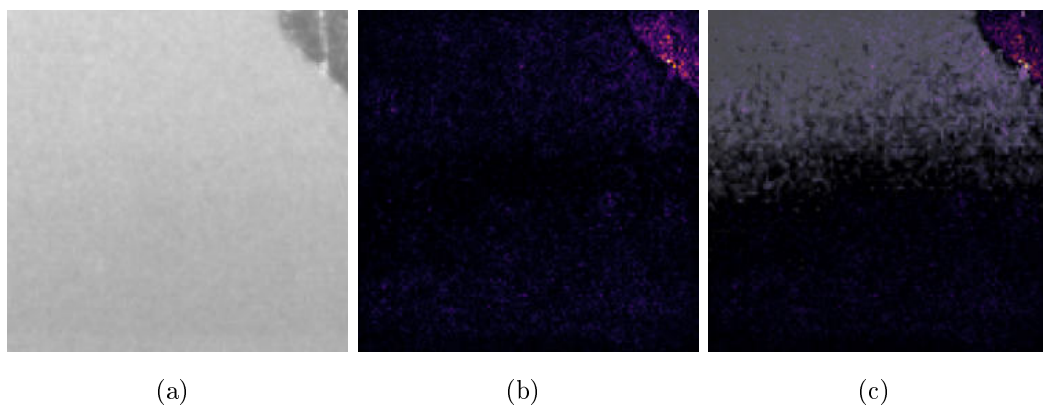


Abbildung 3.27: Original Segment (a) mit Bauteil oben rechts im Bild, Gradienten Maske (b) sowie mit der Maske überlagertes original Bild (c). Klassifizierung erfolgte als Fehler.

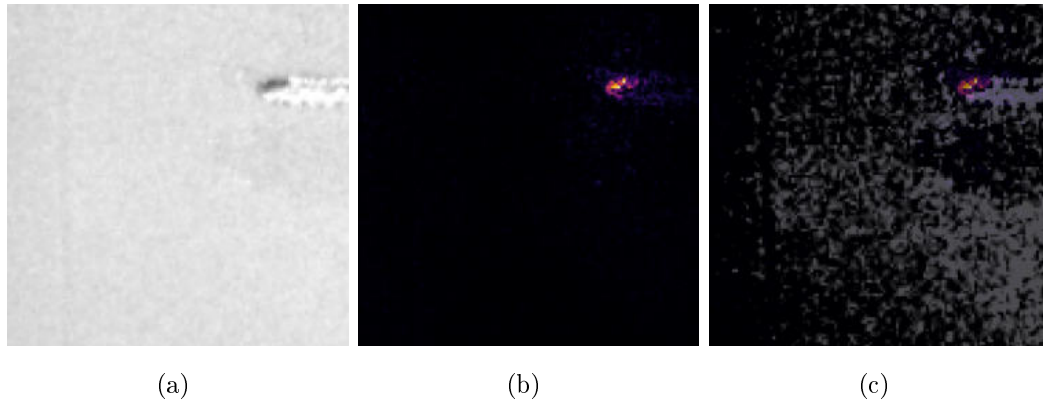


Abbildung 3.28: Original Segment (a) mit Bauteil links im Bild, Gradienten Maske (b) sowie mit der Maske überlagertes original Bild (c). Klassifizierung erfolgte als Fehler.

Aus Tabelle 3.4 wird ersichtlich, dass sich das Modell relativ sicher ist, falls es einen Fehler als fehlerhaft klassifiziert (siehe Precision). Bei der Klasse Pulver scheint sich das Modell etwas unsicherer zu sein, da es hier zwar relativ viele der tatsächlichen Pulversegmente gefunden hat, jedoch dafür auch öfters Bilder fälschlicherweise als Pulver klassifiziert hat.

Tabelle 3.4: F_1 -Score des finalen Modells auf den Validierungsdaten

Klasse	Precision	Recall	F1-Score
Pulver	0.8696	0.9783	0.9207
Bauteil	0.8667	0.8744	0.8705
Fehler	0.9403	0.8750	0.9065
Accuracy			0.8996
Macro Average Accuracy	0.8922	0.9092	0.8992
Weighted Average Accuracy	0.9019	0.8996	0.8994

3 Praktische Umsetzung

Für die abschließende Beurteilung wurde nun das Modell auf dem Testdatensatz, welcher bisher unangetastet blieb, evaluiert. Da es beim maschinellen Lernen Aufgrund der zufällig initialisierten Gewichte zu unterschiedlichen Ergebnissen kommen kann, wurde das hier gewählte Modell weitere vier mal trainiert und abgespeichert. Diese fünf abgespeicherten Modelle wurde dann auf dem Testdatensatz evaluiert. Anschließend wurden anhand dieser Werte der Durchschnitt und die Standardabweichung gebildet (siehe Tabelle 3.5). Die Ergebnisse der einzelnen Modelle sind zusätzlich im Anhang A zu finden. Für die weitere Verwendung innerhalb dieser Arbeit wurde das bisher verwendete Modell beibehalten. Die Entscheidung für dieses Modell wurde aufgrund der Konfusionsmatrix und des F1-Scores der Validierungsdaten getroffen, da dieses Modell auf allen drei Klassen in etwa gleich gute Ergebnisse lieferte. Der Testdatensatz hatte in dieser Entscheidungsfindung keinen Einfluss.

Tabelle 3.5: F_1 -Score des finalen Modells auf den Testdaten

Klasse	Precision	Recall	F1-Score
Pulver	0.8418±0.0279	0.9804±0.0071	0.9057±0.0190
Bauteil	0.9039±0.0150	0.7540±0.0291	0.8216±0.0127
Fehler	0.8343±0.0150	0.8607±0.0213	0.8471±0.0113
Accuracy			0.8574±0.0080
Macro Average Accuracy	0.8600±0.0070	0.8650±0.0076	0.8581±0.0084
Weighted Average Accuracy	0.8618±0.0064	0.8574±0.0080	0.8552±0.0082

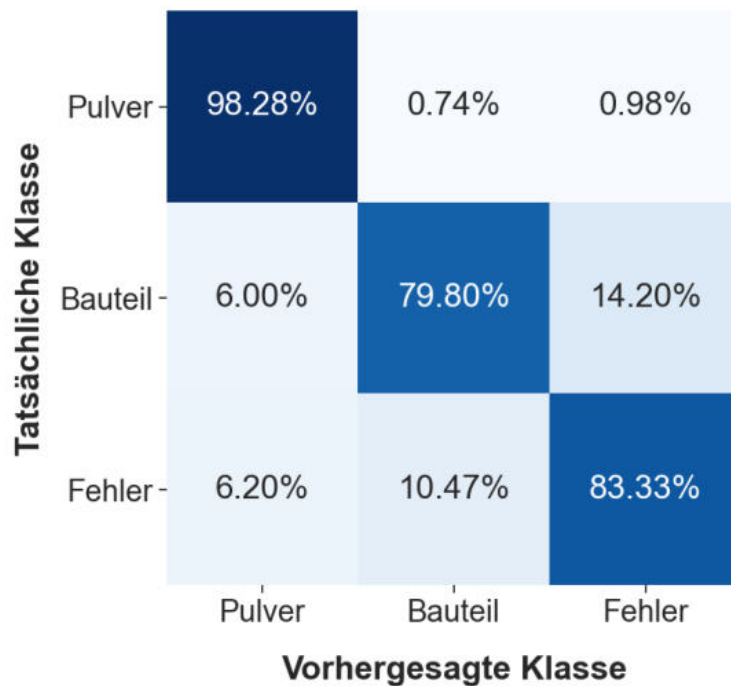


Abbildung 3.29: Konfusionsmatrix des finalen Modells auf den Testdaten

Die Konfusionsmatrix 3.29 deutet auf eine leichte Überanpassung des Modells auf die Validierungsdaten. Die Rate der korrekt klassifizierten Pulveraufnahmen ist annähernd konstant geblieben. Dafür hat sich die Korrektklassifizierungsrate der Bauteil- und Fehleraufnahmen im Testdatensatz um etwa 5-8 % gegenüber dem Validierungsdatensatz verschlechtert. Auffällig ist hier ebenfalls, dass das Modell tendenziell Bauteil- und Fehleraufnahmen schlechter auseinanderhalten kann, als etwa Pulver und Bauteile. Um die Entscheidungsfindung des Modells besser zu verstehen wurden auch hier die falsch klassifizierten Aufnahmen näher betrachtet.

3 Praktische Umsetzung

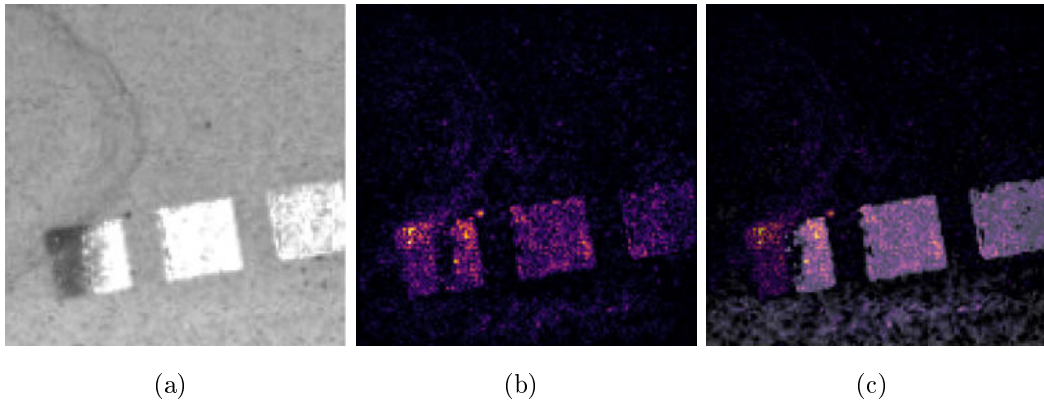


Abbildung 3.30: Original Segment (a) mit Bauteilen unten und einem Fehler links im Bild, Gradienten Maske (b) sowie mit der Maske überlagertes original Bild (c). Klassifizierung erfolgte als Bauteil.

Weder die schwarze Verbrennung am Bauteil (Abbildung 3.30(a)), noch die darüber liegende Mulde im Pulverbett wurden vom CNN erkannt. Stattdessen hat das Modell diese Aufnahme als Bauteil eingestuft.

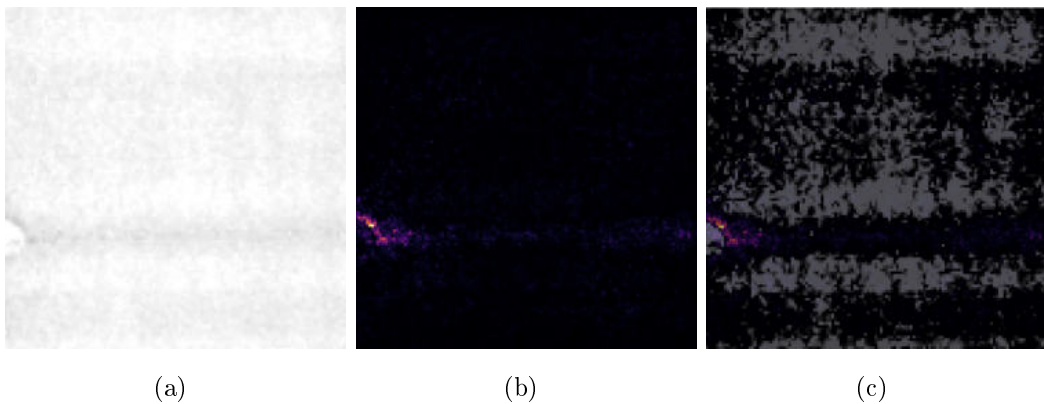


Abbildung 3.31: Original Segment (a) mit einem Bauteil links im Bild, Gradienten Maske (b) sowie mit der Maske überlagertes original Bild (c). Klassifizierung erfolgte als Pulver.

Das Bauteil am linken Rand der Aufnahme (Abbildung 3.31(a)) wurde vom Modell zwar teilweise erkannt, jedoch wurde die Aufnahme dennoch als Pulveraufnahme kategorisiert. Bei dieser Aufnahme fällt zudem auf, dass das Pulverbett in einigen Bereichen sehr stark überbelichtet ist.

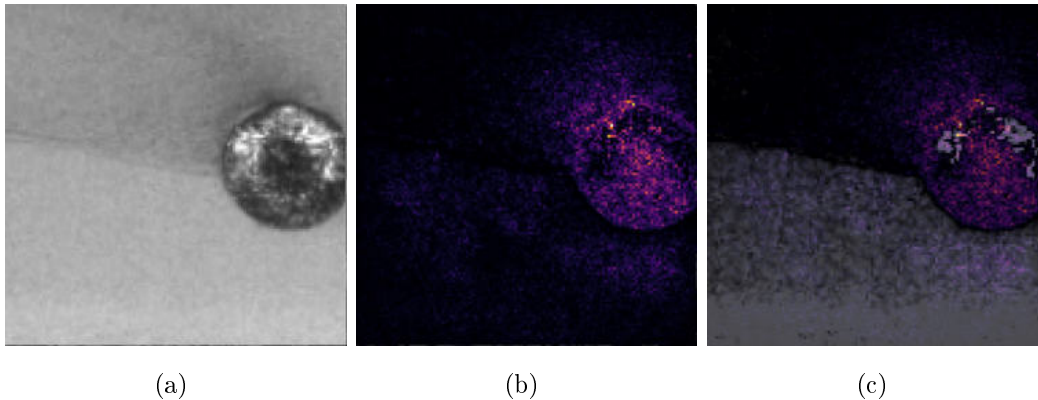


Abbildung 3.32: Original Segment (a) mit einem verbrannten Bauteil und ungleichmäßig verteiltem Pulver im Bild, Gradienten Maske (b) sowie mit der Maske überlagertes original Bild (c). Klassifizierung erfolgte als Bauteil.

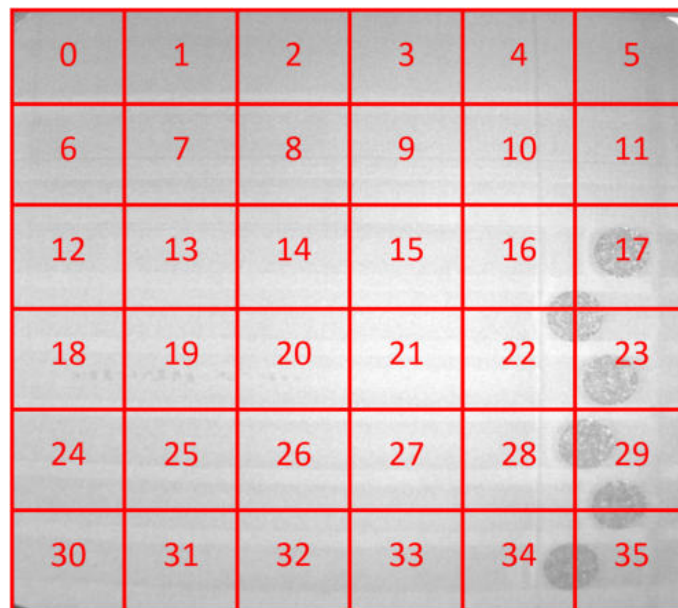
Abbildung 3.32 verdeutlicht zudem wieder, dass das Modell zum einen Probleme damit hat, subtile Strukturen zu erkennen und sich dann stattdessen auf die eher starken Kontraste im Bild konzentriert. Auffällig ist außerdem, dass deutliche Fehler wie Streifen oder sogenannte Deltafehler kaum falsch klassifiziert wurden. Stattdessen sind vor allem Bauteile und Aufnahmen, welche zu mehr als einer Klasse gehören für das Modell schwierig einzuordnen.

3.5.3 Fazit des Trainings auf Segmenten

Das auf Bildsegmenten trainierte Modell funktioniert deutlich besser als die ersten Trainingsversuche, welche das ganze Pulverbett betrachteten. Der Versuch, das Neuronale Netz durch kleine Ausschnitte gezielter zu trainieren war somit zielführend. Dennoch scheint die hier verwendete Methodik, auch aufgrund des sehr begrenzten Datentechnischen Umfangs, an seine Grenzen zu stoßen. Aufnahmen welche mehreren Klassen zugehören sowie die Kombination aus Softmax und Argmax Funktion zur Ermittlung der Klassenzugehörigkeit stellen dabei einen Flaschenhals dar. Ebenfalls ist die Betrachtung eines einzelnen Bildes ohne weitere Informationen zur zeitlichen Entwicklung ein einschränkender Faktor, da es in den Segmenten oft Situationen gibt, die durch ein einzelnes Bild nicht eindeutig zugeordnet werden können.

3.6 Zeitliche Betrachtung der Klassifizierungen

Mit dem entworfenen Klassifizierer ist es nun möglich einzelne Segmente eines Pulverbetts zu klassifizieren. Durch aufteilen des Pulverbetts in einzelne Teilbereiche, wie in Abbildung 3.33 verdeutlicht, ist es möglich für diese jeweiligen Segmente eine Klassifizierung durchzuführen.



0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Abbildung 3.33: Aufteilung einer Pulverbettaufnahme in einzelne, nicht überlappende Segmente.

Wie schon im vorherigen Fazit erwähnt, weist das trainierte Modell zwei grundsätzliche Schwächen auf. Zum einen betrachtet es immer nur ein einzelnes Bild für sich genommen, unabhängig von dessen zeitlichem Zusammenhang mit nachfolgenden oder vorherigen Bildern. Dies ist vor allem beim Labeling der Schichtaufnahmen, aber auch bei der Evaluation des Klassifizierers, deutlich geworden. So hat der Klassifizierer nach wie vor Probleme, subtile Strukturen zu erkennen, welche vor allem zu Beginn eines Fehlers auftreten. Dies liegt daran, dass Fehler sich üblicherweise erst über mehrere Schichten hinweg entwickeln und somit im weiteren Verlauf deutlicher sichtbar werden (siehe dazu das Beispiel an Segment Nummer 17 in Abbildung 3.34).

3.6 Zeitliche Betrachtung der Klassifizierungen

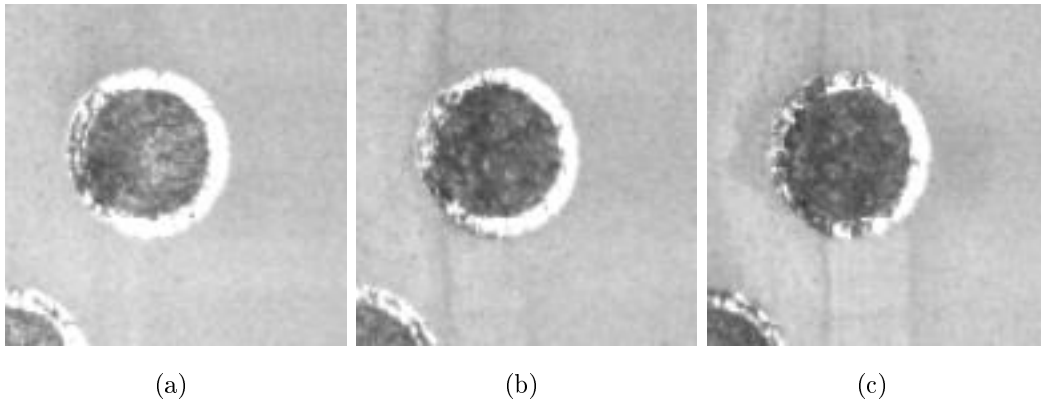


Abbildung 3.34: Zeitliche Beobachtung eines entstehenden Fehlers im Pulverbettsegment 17 (Abbildung 3.33) nach 6,2 mm (a), 7,2 mm (b) und 8,2 mm (c) Bauhöhe eines vertikalen Zylinders. Man beachte dabei die deutlicher werdenden vertikalen Streifen.

Die andere Problematik des Modells liegt in der angewendeten Methodik, da hier ein Multiclass Klassifizierer aufgebaut und trainiert wurde. Dieser Ansatz funktioniert zwar, jedoch erfolgt beim Multiclass Klassifizierer am Ende die Auswertung mittels einer Softmax Funktion, welche die Ausgangswerte des letzten Fully Connected Layers auf einen Wertebereich von 0 bis 1 normalisiert. In diesem Fall, erhält somit jedes Bildsegment drei unterschiedliche Wahrscheinlichkeitswerte, welche die Zugehörigkeit zu einer der drei Klassen widerspiegeln. Die endgültige Klassenzugehörigkeit wird mit Hilfe des Mehrheitsprinzips ausgewählt, d.h. die größte Klassenzugehörigkeit bestimmt das endgültige Label für das Segment. Hier liegt jedoch das Problem. Das CNN erkennt beim Beispiel in Abbildung 3.34 sowohl das Bauteil als auch den Fehler, was dazu führen kann, dass in einer Schicht die Zugehörigkeit zur Klasse der Bauteile angenommen wird und der nächsten die Zugehörigkeit zur Klasse der Fehler wechselt. Dies wird an Segmentnummer 17 (siehe Abbildung 3.33) in Abbildung 3.35 veranschaulicht. Vor allem zu Beginn des Drucks wechselt dabei die Klassenzugehörigkeit des Segments häufig zwischen der Klasse Bauteil und Fehler. Die Schichtnummer stellt dabei die gedruckte Ebene entlang der Z-Achse dar, wobei bei Schicht 0 die Grundplatte angenommen wird.

3 Praktische Umsetzung

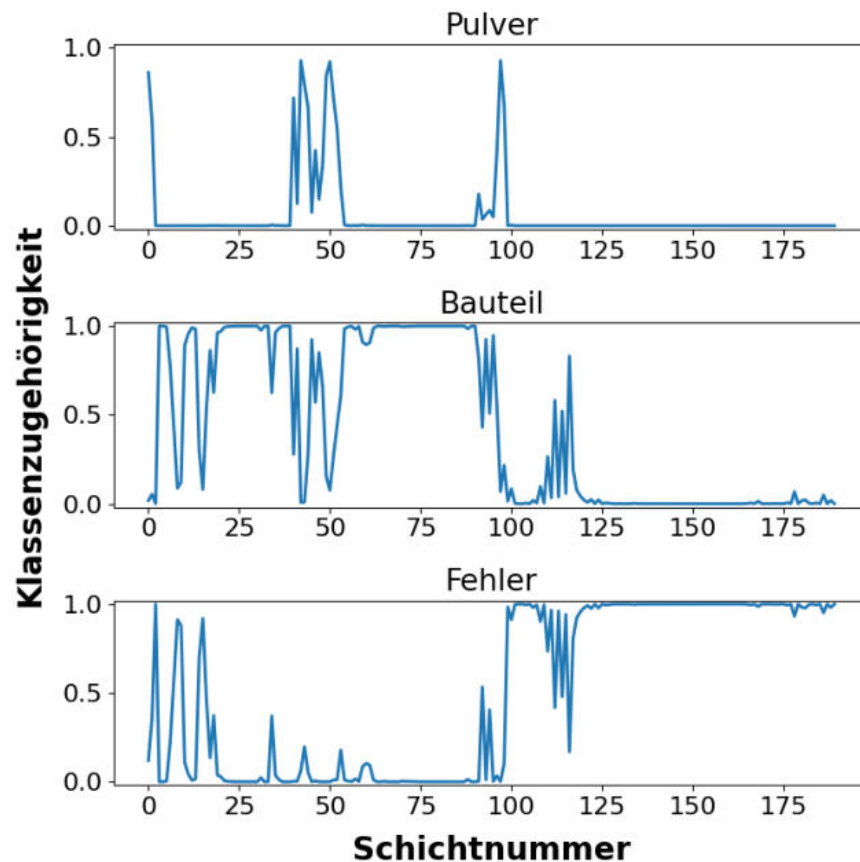


Abbildung 3.35: Zeitliche Entwicklung der Klassenzugehörigkeit von Segment 17. Vor allem zu Beginn des Drucks wechselt die Klassenzugehörigkeit häufig zwischen Bauteil und Fehler.

Auffällig sind hier die starken Schwankungen zwischen zwei aufeinanderfolgenden Schichten (Abbildung 3.35). Diese kommen vor allem zu Beginn des 3D-Drucks dadurch zustande, dass das Pulverbett sich in den ersten 10 bis 20 Schichten noch im Aufbau befindet und somit die Grundplatte noch nicht vollständig mit Pulver bedeckt ist. Aus dem Graph geht außerdem hervor, dass der Fehler in Segment 17 zwischenzeitlich schwächer wird und das Neuronale Netz deshalb das Segment eher der Klasse Bauteil zuordnet.

3.6 Zeitliche Betrachtung der Klassifizierungen

Um dieses oszillieren zwischen zwei Klassen zu verringern und robustere Klassifizierungen zu erhalten ist somit eine zeitliche Betrachtung notwendig. Hierzu wurden die Klassenzugehörigkeitswerte der letzten fünf Schichten des Segments betrachtet. Da davon ausgegangen werden kann, dass das CNN grundsätzlich funktioniert, wurde den Klassifizierungswerten der aktuell gedruckten Schicht die höchste Gewichtung vergeben. Andererseits nimmt die Bedeutung und der Einfluss der einzelnen Schichten stärker ab, je weiter dieser in der Vergangenheit liegen. Deshalb wurde ein gewichteter Mittelwert aus den Ausgabewerten des CNN unter Einbezug der letzten fünf Segmente gebildet. In Tabelle 3.6 ist die verwendete Gewichtung der einzelnen Schichten zu sehen.

Tabelle 3.6: Gewichtung von fünf Aufeinanderfolgender Schichten

Schicht	Gewichtung
s	0.4
s_{-1}	0.3
s_{-2}	0.2
s_{-3}	0.05
s_{-4}	0.05

Dieser gewichtete Mittelwert wurde auf den zeitlichen Verlauf von Segment 17 angewendet (Abbildung 3.36).

3 Praktische Umsetzung

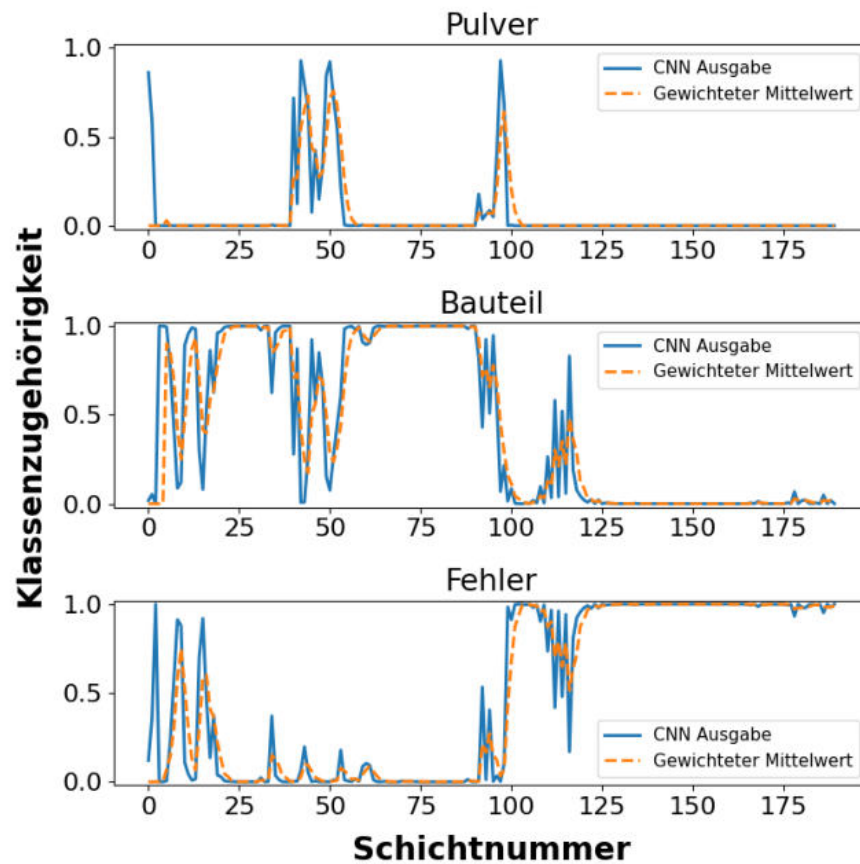


Abbildung 3.36: Zeitliche Entwicklung der Klassenzugehörigkeit von Segment 17 durch das CNN (blau) sowie durch den gewichteten Mittelwert (orange).

Die auf diese Weise erhaltene Klassifizierung schwankt zum einen geringer zwischen zwei Klassen und zum anderen sind die auftretenden Ausreißer deutlich weniger stark ausgeprägt. Diese zeitliche Glättung der CNN Ausgabe wurde auf den einzelnen Segmenten eines Pulverbetts angewendet, mit Ausnahme der vier Ecken, da das CNN in diesen tendenziell eher falsche Klassifizierungen vornimmt. Zudem werden typischerweise keine Bauteile in diesen Eckbereichen gebaut.

3.6 Zeitliche Betrachtung der Klassifizierungen

Um nun ein Label für die komplette Schicht zu erhalten, wurde die Zugehörigkeit der Klasse Fehler eines jeden Segments betrachtet. Als Parameter wurde anhand der Trainingsdaten ein Schwellwert von 70 % festgelegt, d.h. falls eines der Segmente einer Schicht zu mehr als 70 % der Klasse Fehler angehört, wurde der kompletten Schicht das Label Fehler vergeben, ansonsten lautete das Label fehlerfrei. Auf diese Weise sollen eventuelle Fehlklassifizierungen minimiert werden (Abbildung 3.37).

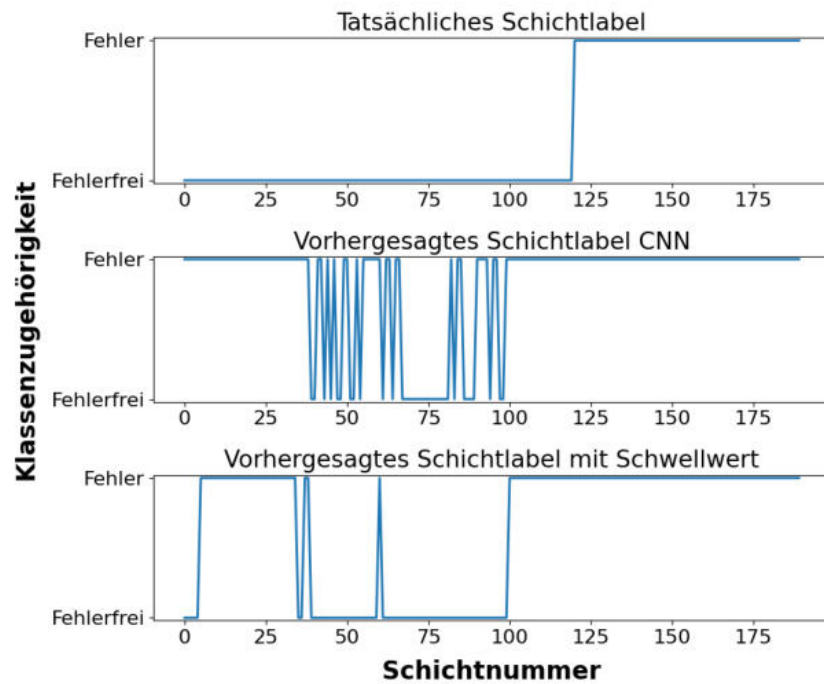


Abbildung 3.37: Das tatsächliche Schichtlabel (oben), sowie die Schichtweise Auswertung des reinen CNN (mitte) und das vergebene Schichtlabel durch Auswertung des CNN mittels gewichtetem Mittelwert und eines zusätzlichen Schwellwertes von 70 % pro Segment. Die fehlerhafte Klassifizierung in den ersten 20 Schichten kommt dabei dadurch zustande, dass das Pulverbett sich hier noch im Aufbau befindet.

3 Praktische Umsetzung

Abschließend wurden auch hier wieder einige Baujobs der Testdaten zur Evaluation herangezogen (siehe Beispiel in Abbildung 3.38).

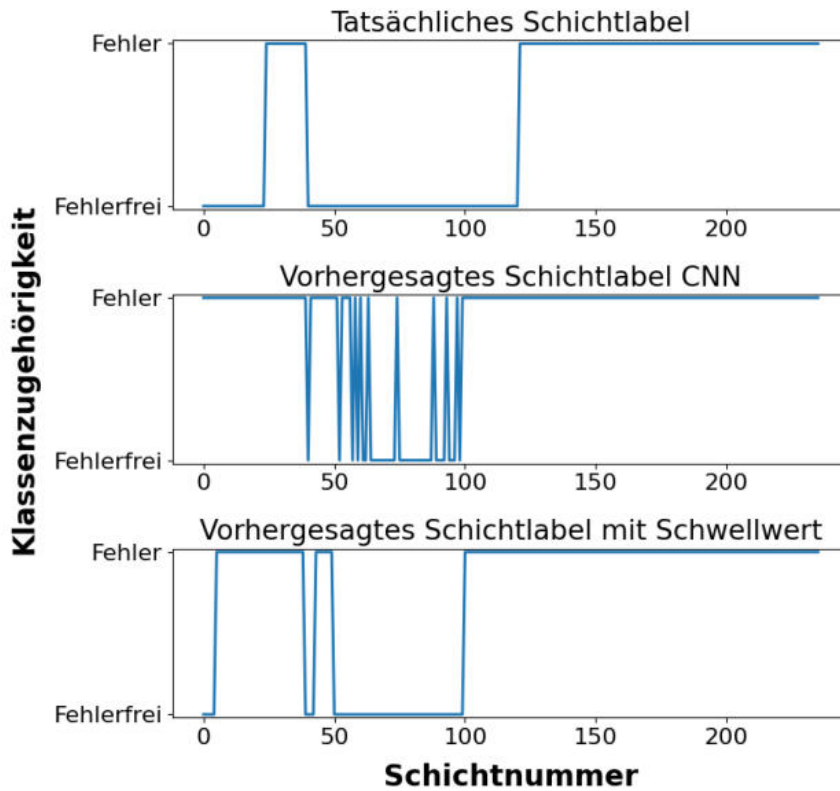


Abbildung 3.38: Das tatsächliche Schichtlabel (oben), sowie die Schichtweise Auswertung des reinen CNN (mitte) und das vergebene Schichtlabel durch Auswertung des CNN mittels gewichtetem Mittelwert und eines zusätzlichen Schwellwertes von 70 % pro Segment.

Dabei fällt auf, dass die Klassifizierung mittels Schwellwert sowie gewichtetem Mittelwert besser funktioniert als die Schichtweise Klassifizierung durch das reine CNN. Dennoch erkennt das CNN vor allem Schichten am Anfang eines Drucks als fehlerhaft an, da diesem hier der notwendige zeitliche Bezug fehlt.

4 Fazit und Ausblick

Das Ziel dieser Arbeit bestand darin, auf Basis von Bilddaten den 3D-Druckprozess des SLM Verfahrens zu überwachen und auftretende Fehler noch während des Prozesses zu erkennen. Die angewendete Methodik, ein CNN zur Bilderkennung einzusetzen, hat sich dabei nur teilweise als zielführend erwiesen. Der Grund hierfür lag dabei hauptsächlich an zwei während der Entwicklung auftretenden Problemen. Zum einen hatte der Datensatz zwar auf den ersten Blick einen großen Umfang von etwa 25000 Schichtaufnahmen, jedoch musste dieser bei genauerer Betrachtung stark verkleinert werden, um eine Überanpassung des Neuronalen Netzes während der Trainingsphase zu vermeiden. Dies lag vor allem daran, dass die einzelnen Schichten eines Baujobs starke Abhängigkeiten von einander aufwiesen, da zwischen diesen nur wenige Mikrometer lagen. Erschwerend kam hier zusätzlich hinzu dass die eigentlichen Schichtaufnahmen nur einen geringen Detailgrad aufwiesen. Die andere große Problematik bestand darin, dass ein reines CNN bei der Klassifizierung zeitliche Abhängigkeiten nicht abbilden kann. So konnte das CNN zwar grundsätzlich Fehler erkennen, jedoch wurden dadurch auch Fehler am Anfang eines Baujobs erkannt, welche eigentlich keine Fehler sind. Bei der Einzelbildbetrachtung können diese zwar als Fehler klassifiziert werden, werden diese Aufnahmen jedoch unter dem Kontext betrachtet, dass das Pulverbett sich noch im Aufbau befindet, sind diese Schichtaufnahmen keine Fehler. Diese fehlende zeitliche Komponente konnte zwar teilweise durch die gewichtete Mittelwertbildung über mehrere Ebenen hinweg kompensiert werden, behebt jedoch nicht die eigentliche Problematik der Einzelbild Klassifizierung.

4 Fazit und Ausblick

Für eine zuverlässigere und robustere Klassifizierung sollte bei zukünftigen hierauf aufbauenden Arbeiten auf ein Hybridmodell gesetzt werden, welches einem CNN den fehlenden zeitlichen Bezug hinzufügt, wie beispielsweise ein ConvLSTM. Allerdings wird dafür eine deutlich größere Trainingsdatenmasse benötigt, da hierbei jeder Druckauftrag als einzelner Datensatz betrachtet werden müsste. Alternativ könnte man aus den Baujobs einzelne Sequenzen von beispielsweise 30 Layern extrahieren und damit ein hybrides CNN trainieren. Zusätzlich wäre es dann möglich durch eine tiefere Datenintegration Abschätzungen zu treffen, ob etwa ein Fehler in der Ecke der Prozesskammer das Bauteil bis zum Ende des Drucks erreichen kann oder ob der Druck vorzeitig abgebrochen werden sollte. Diese Abschätzung zur Ausbreitungsgeschwindigkeit eines Fehlers wäre so auch mit dem hier entwickelten Verfahren möglich, jedoch sollten dafür die Schichtaufnahmen eine höhere Auflösung besitzen und gleichzeitig die einzelnen Felder eventuell noch kleinere Bereiche abdecken.

Literaturverzeichnis

- [1] T. DeRoy, H. L. Wei, J. S. Zuback, T. Mukherjee, J. W. Elmer, J. O. Milewski, A. M. Beese, A. Wilson-Heid, A. De, and W. Zhang. Additive manufacturing of metallic components – process, structure and properties. *Progress in Materials Science*, 92:112–224, 2018. ISSN 00796425. doi: 10.1016/j.pmatsci.2017.10.001.
- [2] Martin Friedrich, Dominik Rietzel, and Gerd Witt. Seriennahe bauteileigenschaften für die automobile produktentwicklung durch additive tooling / series-like product properties for automotive product development by additive tooling. In Wieland Kniffka, Michael Eichmann, Gerd Witt, Wieland Kniffka, Michael Eichmann, and Gerd Witt, editors, *Rapid.Tech – International Trade Show & Conference for Additive Manufacturing*, pages 361–369. Carl Hanser Verlag GmbH & Co. KG, München, 2016. ISBN 978-3-446-45017-2. doi: 10.3139/9783446450608.032.
- [3] Andreas Gebhardt and Jan-Steffen Hötter. *Additive manufacturing: 3D printing for prototyping and manufacturing*. Hanser Publications, Munich and Hanser Publishers and Cincinnati, 2016. ISBN 978-1-56990-582-1. URL <http://www.sciencedirect.com/science/book/9781569905821>.
- [4] Tanisha Pereira, John V. Kennedy, and Johan Potgieter. A comparison of traditional manufacturing vs additive manufacturing, the best method for the job. *Procedia Manufacturing*, 30:11–18, 2019. ISSN 23519789. doi: 10.1016/j.promfg.2019.02.003.
- [5] Stefan Kleszczynski, Joschka Zur Jacobsmühlen, Jan Sehart, and Gerd Witt. Error detection in laser beam melting systems by high resolution imaging. *23rd Annual International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference, SFF 2012*, 2012.

- [6] C. Y. Yap, C. K. Chua, Z. L. Dong, Z. H. Liu, D. Q. Zhang, L. E. Loh, and S. L. Sing. Review of selective laser melting: Materials and applications. *Applied Physics Reviews*, 2(4):041101, 2015. doi: 10.1063/1.4935926.
- [7] Aydin Deliktas. Projektphasenbericht i: Analyse multimodaler prozessdaten aus der additiven fertigung unter nutzung von verfahren des maschinellen lernens projektphasenbericht i, 13. März 2020.
- [8] N.-B Chang, Kaixu Bai, and Ni-Bin Chang. *Multisensor data fusion and machine learning for environmental remote sensing*. CRC Press Taylor & Francis Group, Boca Raton, FL and London and New York NY, 2018. ISBN 978-1-4987-7433-8. doi: 10.1201/b20703.
- [9] Khosrow Kaikhah and Sandesh Doddameti. Discovering trends in large datasets using neural networks. *Applied Intelligence*, 24(1):51–60, 2006. ISSN 0924-669X. doi: 10.1007/s10489-006-6929-9.
- [10] Aydin Deliktas. Projektphasenbericht ii: Analyse multimodaler prozessdaten aus der additiven fertigung unter nutzung von verfahren des maschinellen lernens phasenbericht ii, 30. September 2020.
- [11] Din en iso 13919-1:2018-11, schweißen_- elektronen- und laserstrahl-schweißverbindungen_- leitfaden für bewertungsgruppen für unregelmäßigkeiten_- teil_1: Stahl, nickel, titan und seine legierungen (iso/dis_13919-1:2018); deutsche und englische fassung pren_iso_13919-1:2018.
- [12] H. L. Wei, H. K. D. H. Bhadeshia, S. A. David, and T. DebRoy. Harnessing the scientific synergy of welding and additive manufacturing. *Science and Technology of Welding and Joining*, 24(5):361–366, 2019. ISSN 1362-1718. doi: 10.1080/13621718.2019.1615189.
- [13] Michael Karg. *Laserstrahlschmelzen von Metallen - Aktuelle Entwicklungen im Bereich der Additiven Fertigung*. 2012.

- [14] Dirk Herzog, Vanessa Seyda, Eric Wycisk, and Claus Emmelmann. Additive manufacturing of metals. *Acta Materialia*, 117:371–392, 2016. ISSN 13596454. doi: 10.1016/j.actamat.2016.07.019.
- [15] Thomas G. Spears and Scott A. Gold. In-process sensing in selective laser melting (slm) additive manufacturing. *Integrating Materials and Manufacturing Innovation*, 5(1):16–40, 2016. ISSN 2193-9764. doi: 10.1186/s40192-016-0045-4.
- [16] Mahesh Mani, Brandon Lane, Alkan Donmez, Shaw Feng, Shawn Moylan, and Ronnie Fesperman. Measurement science needs for real-time control of additive manufacturing powder bed fusion processes. URL <http://dx.doi.org/10.6028/NIST.IR.8036>.
- [17] B. K. Foster, E. W. Reutzel, A.R. Nassar, B. T. Hall, S.W. Brown, C.J. Dickman. Center for Innovative Material Processing through Direct Digital Deposition (CIMP-3D) Applied Research Laboratory, editor. *Optical, layer-wise monitoring of powder bed fusion*, Austin, Texas, 2015. URL <https://sffsymposium.engr.utexas.edu/sites/default/files/2015/2015-24-Foster.pdf>.
- [18] F. Bayle and M. Doubenskaia. Selective laser melting process monitoring with high speed infra-red camera and pyrometer. SPIE Proceedings, pages 698505–698505–8. SPIE, 2012. doi: 10.1117/12.786940.
- [19] Marco Grasso, Vittorio Laguzza, Quirico Semeraro, and Bianca Maria Colosimo. In-process monitoring of selective laser melting: Spatial detection of defects via image data analysis. *Journal of Manufacturing Science and Engineering*, 139(5), 2017. ISSN 1087-1357. doi: 10.1115/1.4034715.
- [20] Roberta Angelone, Alessandra Caggiano, Roberto Teti, Adriaan Spierings, Alexandre Staub, and Konrad Wegener. Bio-intelligent selective laser melting system based on convolutional neural networks for in-process fault identification. *Procedia CIRP*, 88:612–617, 2020. ISSN 22128271. doi: 10.1016/j.procir.2020.05.107.
- [21] Sebastian Raschka and Vahid Mirjalili. *Machine Learning mit Python und Scikit-learn und TensorFlow: Das umfassende Praxis-Handbuch für Data*

- Science, Deep Learning und Predictive Analytics*. mitp Professional. mitp, Frechen, 2., aktualisierte und erweiterte auflage edition, 2017. ISBN 978-3-95845-733-1. URL <https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=5210623>.
- [22] Adrian Rosebrock. *Deep learning for computer vision with Python: Practitioner bundle*, volume vol. 2 of *Deep learning for computer vision with Python*. PyImageSearch, [United States], 1st edition 1.3 edition, 2017. ISBN 9781722487836.
- [23] Andreas Christian Müller and Sarah Guido. *Introduction to machine learning with Python: A guide for data scientist*. O'Reilly, Beijing, first edition edition, October 2016. ISBN 978-1-449-36941-5.
- [24] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. Second edition edition, 2019. ISBN 978-1-492-03264-9.
- [25] Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan Hines, C. Bayan Bruss, and Reza Farivar. Towards automated machine learning: Evaluation and comparison of automl approaches and tools. pages 1471–1479, 2019. doi: 10.1109/ICTAI.2019.00209. URL <http://arxiv.org/pdf/1908.05557v2>.
- [26] Bharath Ramsundar and Reza Bosagh Zadeh. *TensorFlow for deep learning: From linear regression to reinforcement learning*. O'Reilly, Beijing and Boston and Farnham and Sebastopol and Tokyo, first edition edition, March 2018. ISBN 978-1-491-98045-3.
- [27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, Cambridge, Massachusetts and London, England, 2016. ISBN 9780262035613. URL <http://www.deeplearningbook.org/>.
- [28] Suzana Herculano-Houzel. The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in human neuroscience*, 3:31, 2009. doi: 10.3389/neuro.09.031.2009.

- [29] Sebastian Raschka. Single-layer neural networks and gradient descent. abgerufen am 01.03.2021, 24.03.2015. URL https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html.
- [30] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer series in statistics. Springer, New York, NY, second edition, corrected at 12th printing 2017 edition, 2017. ISBN 0387848576.
- [31] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, NY, 2006. ISBN 0-387-31073-8. URL <http://www.loc.gov/catdir/enhancements/fy0818/2006922522-d.html>.
- [32] Tomasz Szandała. Review and comparison of commonly used activation functions for deep neural networks. 903, 2021. doi: 10.1007/978-981-15-5495-7. URL <http://arxiv.org/pdf/2010.09458v1>.
- [33] Hamed Habibi Aghdam and Elnaz Jahani Heravi. *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. Springer International Publishing, Cham, 2017. ISBN 978-3-319-57550-6. URL <https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4862504>.
- [34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. URL <http://arxiv.org/pdf/1512.03385v1>.
- [35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. URL <http://arxiv.org/pdf/1409.1556v6>.
- [36] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. URL <http://arxiv.org/pdf/1502.03167v3>.
- [37] S. Kevin Zhou, Hayit Greenspan, and Dinggang Shen. *Deep Learning for Medical Image Analysis*. Elsevier Science, Saint Louis, 2017. ISBN 978-0-128-10408-8. URL <https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4789490>.

- [38] Srivastava, Nitish and Hinton, Geoffrey and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, pages 1929–1958, 2014. URL <https://dl.acm.org/doi/10.5555/2627435.2670313>.
- [39] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. URL <http://arxiv.org/pdf/1506.02158v6>.
- [40] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. URL <http://arxiv.org/pdf/1703.01365v2>.
- [41] Tensorflow. Integrated gradients | tensorflow core. abgerufen am 10.02.2021. URL https://www.tensorflow.org/tutorials/interpretability/integrated_gradients.
- [42] DMG MORI. Lasertec 30 dual slm - additive manufacturing maschinen von dmg mori. abgerufen am 20.12.2020. URL <https://de.dmgmori.com/produkte/maschinen/additive-manufacturing/pulverbettverfahren/lasertec-30-slm>.
- [43] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851.
- [44] OpenCV. Opencv: Geometric image transformations. abgerufen am 10.01.2021. URL https://docs.opencv.org/4.5.1/da/d54/group__imgproc__transform.html.
- [45] Young-Jin Cha, Wooram Choi, and Oral Büyüköztürk. Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32(5):361–378, 2017. ISSN 10939687. doi: 10.1111/mice.12263.
- [46] Amir Rezaie, Radhakrishna Achanta, Michele Godio, and Katrin Beyer. Comparison of crack segmentation using digital image correlation measurements and deep learning. *Construction and Building Materials*, 261:120474, 2020. ISSN 09500618. doi: 10.1016/j.conbuildmat.2020.120474.

- [47] Hongyan Xu, Xiu Su, Yi Wang, Huaiyu Cai, Kerang Cui, and Xiaodong Chen. Automatic bridge crack detection using a convolutional neural network. *Applied Sciences*, 9(14):2867, 2019. doi: 10.3390/app9142867.
- [48] Shengyuan Li and Xuefeng Zhao. Image-based concrete crack detection using convolutional neural network and exhaustive search technique. *Advances in Civil Engineering*, 2019:1–12, 2019. ISSN 1687-8086. doi: 10.1155/2019/6520620.

Abbildungsverzeichnis

2.1	Aufbau Laserstrahlschmelzmaschine	4
2.2	Schema Schweißraupe	5
2.3	Pulver- und Bauteilfehler	7
2.4	Weiterer Pulver- und Bauteilfehler	8
2.5	Maschinenfehler	9
2.6	Deltafehler	10
2.7	Visuelle Darstellung der Datenaufbereitung	14
2.8	Graphische Veranschaulichung von Over- und Underfitting	15
2.9	Beispiel einer Konfusionsmatrix	21
2.10	Perzeptron	22
2.11	Globales und lokales Minimum	24
2.12	Sigmoid und ReLU Aktivierungsfunktionen	25
2.13	Beispiel einer 2D Faltung	28
2.14	Beispiel des MAXPooling Verfahrens	29
3.1	Beispiel einer Schichtaufnahme	37
3.2	Verteilung der Schichtaufnahmen nach zugewiesener Klasse	39
3.3	Augmentiertes Schichtbild	40
3.4	Validierungs- und Trainingskurven des ersten Modells	41
3.5	Konfusionsmatrix des ersten Modells	42
3.6	Gradienten Bilder des ersten Modells	43
3.7	Validierungs- und Trainingskurven des zweiten Modells	46
3.8	Konfusionsmatrix des zweiten Modells	47
3.9	Gradienten Bilder des zweiten Modells	48
3.10	Original und entzerrtes Pulverbett	50
3.11	Validierungs- und Trainingskurven des dritten Modells	51

Abbildungsverzeichnis

3.12	Konfusionsmatrix des dritten Modells	52
3.13	Gradienten Bilder des dritten Modells	53
3.14	Falsch klassifizierte Schichtaufnahmen	54
3.15	Beginn eines 3D-Drucks	55
3.16	Validierungs- und Trainingskurven des vierten Modells	56
3.17	Konfusionsmatrix des vierten Modells	57
3.18	Gradienten Bilder des vierten Modells	58
3.19	Prinzip einer Bildsegmentierung	60
3.20	Labeling Tool V1	61
3.21	Einfluss der Feldgröße auf ein Segment	62
3.22	Beispiel eines Bauteilfehlers	64
3.23	Labeling Tool V2	65
3.24	Validierungs- und Trainingskurven des segmentierten Modells	68
3.25	Konfusionsmatrix des finalen Modells auf dem Validierungsdatensatz	69
3.26	Erstes Gradienten Bild des finalen segmentierten Modells	70
3.27	Zweites Gradienten Bild des finalen segmentierten Modells	70
3.28	Drittes Gradienten Bild des finalen segmentierten Modells	71
3.29	Konfusionsmatrix des finalen Modells auf den Testdaten	73
3.30	Erstes Gradienten Bild des finalen Modells der Testdaten	74
3.31	Zweites Gradienten Bild des finalen Modells der Testdaten	74
3.32	Drittes Gradienten Bild des finalen Modells der Testdaten	75
3.33	Aufteilung einer Pulverbettaufnahme in Segmente	76
3.34	Zeitliche Entwicklung eines Fehlers	77
3.35	Zeitliche Entwicklung des Segments 17	78
3.36	Zeitliche Entwicklung des gewichteten Mittelwerts von Segments 17	80
3.37	Schichtweise Klassifizierung eines Baujobs in Fehlerfrei und Fehlerhaft	81
3.38	Weiteres Beispiel zur schichtweisen Klassifizierung eines Baujobs in Fehlerfrei und Fehlerhaft	82
A.1	Konfusionsmatrix des finalen Modells auf den Testdaten: Training 1	105
A.2	Konfusionsmatrix des finalen Modells auf den Testdaten: Training 2	106
A.3	Konfusionsmatrix des finalen Modells auf den Testdaten: Training 3	107
A.4	Konfusionsmatrix des finalen Modells auf den Testdaten: Training 4	108

A.5 Konfusionsmatrix des finalen Modells auf den Testdaten: Training 5 . 109

Tabellenverzeichnis

3.1	Festgelegte Label und ihre Definitionen	38
3.2	Festgelegte Label und ihre Definitionen für das Labeling Tool V2 . . .	65
3.3	Architektur des finalen Modells	67
3.4	F_1 -Score des finalen Modells auf den Validierungsdaten	71
3.5	F_1 -Score des finalen Modells auf den Testdaten	72
3.6	Gewichtung von fünf Aufeinanderfolgender Schichten	79
4.1	F1-Score des finalen Modells auf den Testdaten: Training 1	105
4.2	F1-Score des finalen Modells auf den Testdaten: Training 2	106
4.3	F1-Score des finalen Modells auf den Testdaten: Training 3	107
4.4	F1-Score des finalen Modells auf den Testdaten: Training 4	108
4.5	F1-Score des finalen Modells auf den Testdaten: Training 5	109

Abkürzungsverzeichnis

AM	Additive Manufacturing
CNN	Convolutional Neural Network
FC	Fully Connected Layer
IG	Integrated Gradients
ML	Machine Learning
ReLU	Rectified Linear Units
SGD	Stochastic Gradient Descent
SLM	Selective Laser Melting

Danksagung

An dieser Stelle bedanke ich mich bei Herr Prof. Dr. Michael Möckel für das ermöglichen dieser Masterarbeit und deren tatkräftige Betreuung. Ebenfalls möchte ich mich beim Herrn Jorrit Voigt für seine fachliche Unterstützung bedanken. Von Seiten der Hochschule bedanke ich mich bei der Arbeitsgruppe Angewandte Lasertechnik und Photonik unter Leitung von Prof. Dr. Ralf Hellmann, welche die Datengrundlage lieferten und ohne deren Unterstützung diese Masterarbeit nicht möglich gewesen wäre.

Zuletzt danke ich noch meinen Eltern, welche mir dieses Studium ermöglicht, und mich dabei immer unterstützt haben.

Anhang

A Ergebnisse des Testdatensatzes für die einzelnen Modelle

Folgende Tabellen stellt die Ergebnisse des finalen Modells auf den Testdaten dar. Verwendet wurde in dieser Arbeit dabei das Modell des ersten Trainings.

Training 1

Tabelle 4.1: F1-Score des finalen Modells auf den Testdaten: Training 1

Klasse	Precision	Recall	F1-Score
Pulver	0.8717	0.9828	0.9240
Bauteil	0.8847	0.7980	0.8391
Fehler	0.8387	0.8303	0.8360
Accuracy			0.8648
Macro Average Accuracy	0.8650	0.8714	0.8664
Weighted Average Accuracy	0.8652	0.8648	0.8632

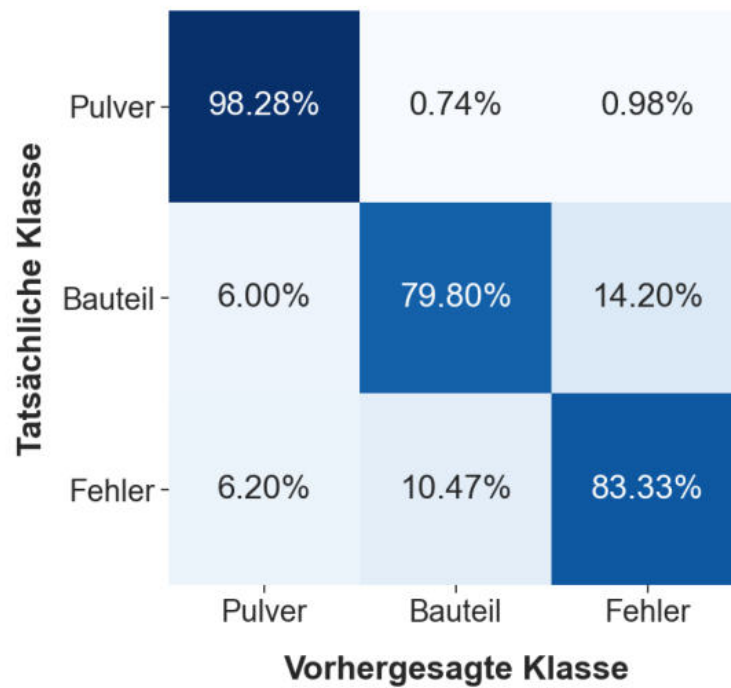


Abbildung A.1: Training 1: Konfusionsmatrix des finalen Modells auf den Testdaten.

Training 2

Tabelle 4.2: F1-Score des finalen Modells auf den Testdaten: Training 2

Klasse	Precision	Recall	F1-Score
Pulver	0.8493	0.9804	0.9101
Bauteil	0.8866	0.7660	0.8219
Fehler	0.8309	0.8397	0.8353
Accuracy			0.8547
Macro Average Accuracy	0.8556	0.8620	0.8558
Weighted Average Accuracy	0.8566	0.8547	0.8526

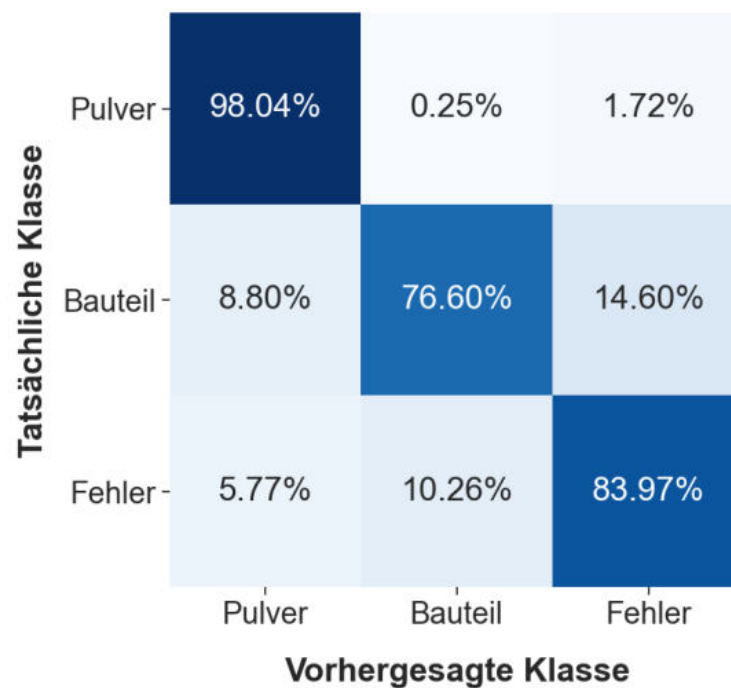


Abbildung A.2: Training 2: Konfusionsmatrix des finalen Modells auf den Testdaten.

Training 3

Tabelle 4.3: F1-Score des finalen Modells auf den Testdaten: Training 3

Klasse	Precision	Recall	F1-Score
Pulver	0.8403	0.9804	0.9050
Bauteil	0.9159	0.7620	0.8319
Fehler	0.8512	0.8803	0.8655
Accuracy			0.8670
Macro Average Accuracy	0.8691	0.8742	0.8675
Weighted Average Accuracy	0.8715	0.8670	0.8650

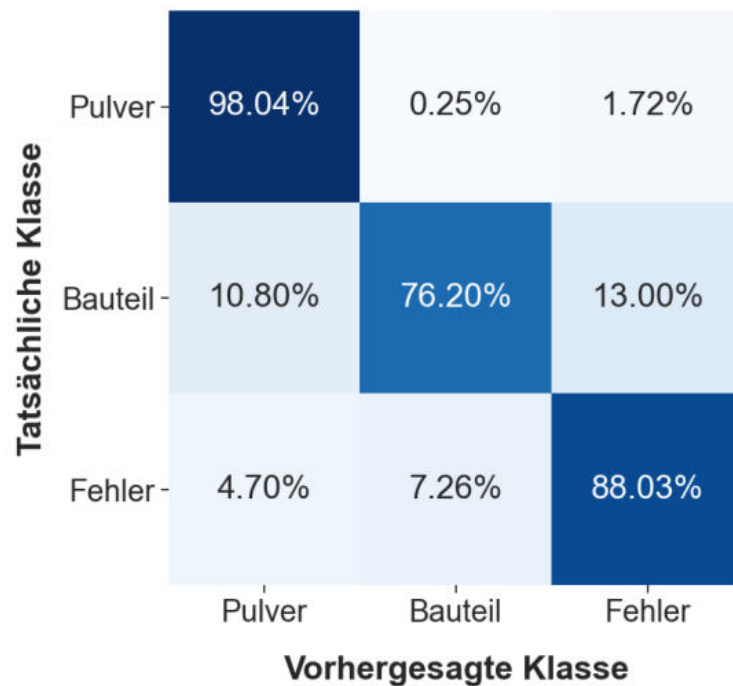


Abbildung A.3: Training 3: Konfusionsmatrix des finalen Modells auf den Testdaten.

Training 4

Tabelle 4.4: F1-Score des finalen Modells auf den Testdaten: Training 4

Klasse	Precision	Recall	F1-Score
Pulver	0.8577	0.9902	0.9192
Bauteil	0.9182	0.7180	0.8058
Fehler	0.8074	0.8868	0.8452
Accuracy			0.8561
Macro Average Accuracy	0.8611	0.8650	0.8568
Weighted Average Accuracy	0.8626	0.8561	0.8529

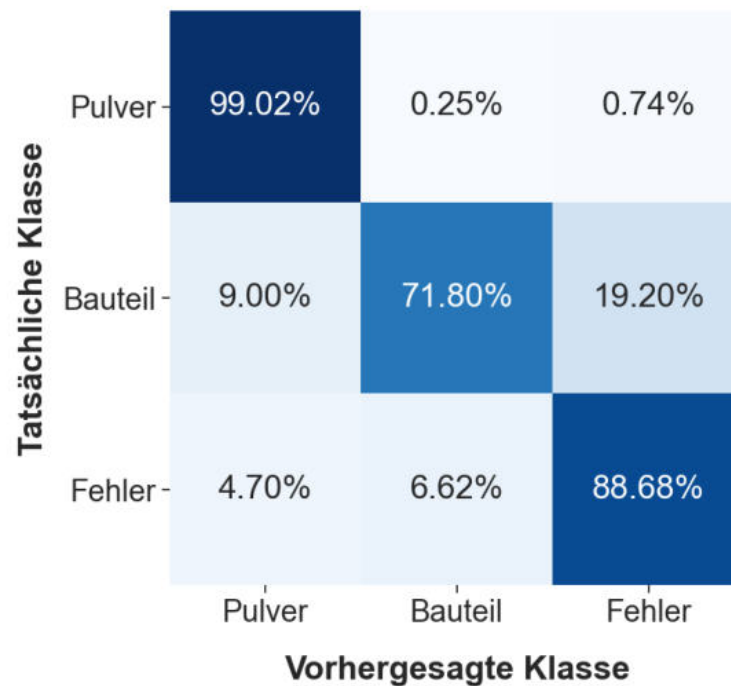


Abbildung A.4: Training 4: Konfusionsmatrix des finalen Modells auf den Testdaten.

Training 5

Tabelle 4.5: F1-Score des finalen Modells auf den Testdaten: Training 5

Klasse	Precision	Recall	F1-Score
Pulver	0.7900	0.9681	0.8700
Bauteil	0.9144	0.7260	0.8094
Fehler	0.8434	0.8632	0.8532
Accuracy			0.8445
Macro Average Accuracy	0.8493	0.8525	0.8442
Weighted Average Accuracy	0.8534	0.8445	0.8423

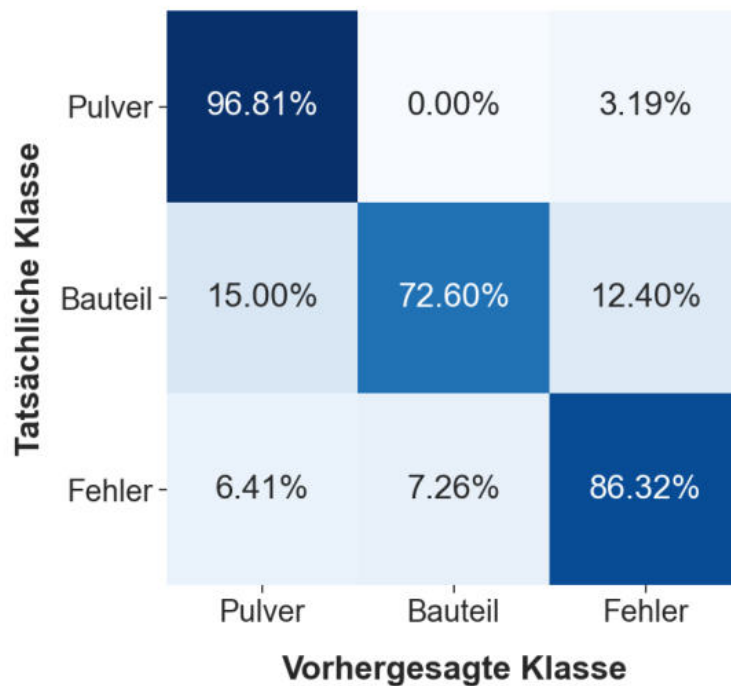


Abbildung A.5: Training 5: Konfusionsmatrix des finalen Modells auf den Testdaten.