

Bachelorarbeit

Fabian Kohnle

Matrikelnr.: 2208146

Bildverarbeitungsbasierte Überwachung von Zerspanungswerkzeugen mit maschinellem Lernen in einer industriellen Umgebung

wbk

Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)
Kaiserstraße 12
76131 Karlsruhe

Prof. Dr.-Ing. Jürgen Fleischer

Prof. Dr.-Ing. Gisela Lanza

Prof. Dr.-Ing. habil. Volker Schulze

wbk

Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)
Kaiserstraße 12
76131 Karlsruhe
Prof. Dr.-Ing. Jürgen Fleischer
Prof. Dr.-Ing. Gisela Lanza
Prof. Dr.-Ing. habil. Volker Schulze



Bachelorarbeit

für Herrn Fabian Kohnle, Ludwig-Wilhelm-Str. 2, 76131 Karlsruhe
Matrikel Nr. 2208146

Bildverarbeitungsbasierte Überwachung von Zerspanungswerkzeugen mit maschinellem Lernen in einer industriellen Umgebung

Image Processing Based Cutting Tool Monitoring Using Machine Learning in an Industrial Environment

Aufgrund der Komplexität der direkten Werkzeugverschleißmessungen während der Zerspanung konzentrierten sich die Bemühungen der wissenschaftlichen Gemeinschaft und der Fertigungsindustrie in den letzten Jahren auf Ansätze zur autonomen Messung, Überwachung und voraussagenden Wartung.

Im Rahmen dieser Arbeit werden verschiedene Image Processing Ansätze zur Werkzeugklassifizierung und zur Verschleißidentifizierung implementiert, trainiert und nach ihrer Präzision und Anwendbarkeit in einem realen Szenario ausgewertet.

Die Arbeit wird folgenden Forschungspunkten adressieren:

- Identifizierung Werkzeugart: Wendeschneidplatte oder Fräser
- Identifizierung verschlissenes oder gebrochenes Werkzeug
- Identifizierung quantitativer abrasiver Werkzeugverschleiß.

Hierfür wird der Bearbeitungsprozess Außenlängsdrehen, unter Verwendung des Stahls 42CrMo4, beschichteter Wendeschneidpatten und eines USB-Mikroskops herangezogen. Image Processing Methoden werden in Python unter Verwendung von Machine Learning und insbesondere TensorFlow und Keras programmiert. Direkt auswirkende Faktoren wie die Kamerapositionierung, Beleuchtung und intrinsische Modellparameter wie Hyperparameter und Optimierungsalgorithmen werden für optimale Ergebnisse iterativ variiert.

Diese Arbeit wird dazu beitragen, einen Überblick darüber zu geben, wie neue Techniken auf Basis von Image Processing und maschinellem Lernen mit klassischen Bearbeitungsprozessen kombiniert werden können, um so zu einer Steigerung der Prozessleistung und einer Verringerung von Maschinenstillstandzeiten beizutragen.

Interne Nr. der Arbeit: FWT 3525
Tag der Ausgabe: 2021-10-15
Abgabe bis spätestens: 2022-01-17
Anleitung: Germán González Fernández, M.Sc.

Karlsruhe, 2021-10-15

Prof. Dr.-Ing. habil. Volker Schulze

Eigenständigkeitserklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, 17. Januar 2022



Fabian Kohnle

Danksagung

An dieser Stelle möchte ich mich bei alle denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und motiviert haben.

Ich möchte meinem Betreuer Germán González danken, der mir dieses interessante und vielversprechende Thema vorgeschlagen hat. Gleichzeitig möchte ich mich für die enge und freundschaftliche Zusammenarbeit bedanken, wodurch ich die nötige Unterstützung bekommen habe, um dies Bachelorarbeit fertigzustellen.

Herrn Prof. Dr.-Ing. habil. Volker Schulze möchte ich auch danken, der durch seine Betreuung diese Arbeit ermöglicht hat.

Ein besonderer Dank gilt meinen Familienmitgliedern und Freunden, die mich beim Korrekturlesen sehr unterstützt haben.

Abschließend möchte ich mich bei meinen Eltern Marianne und Alfred bedanken, die mir mein Studium durch ihre großzügige Unterstützung ermöglicht haben.

Kurzfassung

Werkzeugverschleiß ist ein Kostentreiber bei spanenden Bearbeitungsverfahren. Nicht nur durch die teuren Werkzeuge selbst werden Kosten verursacht, sondern auch durch Maschinenstillstandzeiten, die durch das Wechseln der Werkzeuge verursacht werden. Zusatzkosten entstehen durch Nachbearbeitung beschädigter Oberflächen, Schrotteilen und durch Beschädigungen an der Werkzeugmaschine. Deshalb ist es notwendig verschlissene Werkzeuge bei einem bestimmten Verschleißgrad zu tauschen. Diese Verschleißidentifikation wird in der Regel manuell durch den Menschen und nicht automatisiert durchgeführt. Dabei besteht die Gefahr, dass die Werkzeuge zu früh oder zu spät getauscht werden. Um dieses Problem zu beheben, kann ein Machine Learning (ML) Ansatz verwendet werden, um den Verschleiß zu erkennen und zu identifizieren. In dieser Arbeit wird mithilfe von Tensorflow und Keras ein ML-Bilderkennungsprogramm entwickelt, das die Art des Werkzeugs bestimmt und die Verschleißfläche quantitativ auswertet. Im ersten Schritt wird ein Klassifizierungsalgorithmus basierend auf einem Convolutional Neural Network (CNN) angewendet. Dieser kann identifizieren, ob das verwendete Werkzeug zu einem Schaftfräser oder einer Wendeschneidplatte (WSP) gehört. Im Anschluss wird mithilfe eines Semantic Segmentation Ansatzes, unter Verwendung eines U-Nets, der Verschleiß von Wendeschneidplatten quantitativ bestimmt. Bei der Identifizierung des Verschleißes wurde ein Intersection over Union (IoU) Wert von ca. 0,8 erreicht. Um die Anwendungsmöglichkeiten zu demonstrieren, werden diese Bilderkennungsprogramme mithilfe eines Raspberry Pi in eine Werkzeugmaschine implementiert und angewendet. Die Verschleißmessung erfolgt mit vorheriger Kalibrierung und Bestimmung der Pixelgröße über das Zählen der Pixel, die als Verschleißfläche klassifiziert wurden. Die Auswertung der Verschleißmessung ergab unter konstanten Bedingungen einen gleichbleibenden Wert für die Verschleißfläche bei unterschiedlichen Messungen. Diese Arbeit trägt dazu bei, einen Überblick darüber zu geben, wie vollautomatisierte Verschleißidentifizierungstools, basierend auf Machine Learning Ansätzen, Anwendung in einer industriellen Umgebung finden können.

Abstract

Tool wear is a cost driver in machining processes. Not only by the expensive tools itself costs are incurred but also by machine downtimes, caused by changing the tools. Additional costs can also arise from having to rework damaged surfaces, scrap parts or damage to the machine tool itself. For this reason, it is necessary to change the worn tools at a certain degree of wear. This wear identification is usually carried out manually by humans, not automatically. Therefore, there is a risk that the tools will be exchanged too early or too late. To solve this problem, a machine learning approach can be used to detect and identify the tool wear. In this work, with the help of Tensorflow and Keras, a machine learning image recognition program is developed that determines the type of tool and can quantitatively determine the wear area. In the first step, a classification algorithm based on a convolutional neural network (CNN) is applied. This can identify whether the used tool belongs to an end mill or an indexable insert. Subsequently, the wear of indexable inserts is quantitatively determined with the help of a semantic segmentation approach using a U-Net. When the wear was identified, an Intersection over Union (IoU) value of approximately 0.8. To demonstrate the possible applications, these image recognition programs were implemented and used in a machine tool using a Raspberry Pi. The wear measurement was carried out after prior calibration and determination of the pixel size by counting the pixels that were classified as wear area. The evaluation of the wear measurement under constant conditions resulted in a steady value for the tool wear area for different measurements. This thesis helps to provide an overview of how fully automated wear identification tools based on machine learning approaches can be used in an industrial environment.

Inhaltsverzeichnis

Abkürzungen	III
1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	2
1.3 Aufbau der Arbeit	2
2 Grundlagen	3
2.1 Werkzeugverschleiß	3
2.2 Machine Learning	5
2.2.1 Unterschied von Artificial Intelligence, Machine Learning und Deep Learning	5
2.2.2 Neuronale Netzwerke (Neural Networks)	6
2.2.3 Supervised und Unsupervised Learning	8
2.2.4 Backpropagation	9
2.2.5 Cost Function	9
2.2.6 Gradient Descent	9
2.2.7 Overfitting	10
2.2.8 Convolutional Neural Network	11
2.2.9 Computer Vision	15
2.2.10 Tensorflow und Keras	15
2.2.11 Semantic Segmentation	16
2.2.12 Evaluation Metrics	18
3 Stand der Technik	20
3.1 Aktuelle Anwendungsbereiche ML-Bilderkennung	20
3.1.1 Anwendung im Finanzwesen	20
3.1.2 Anwendung in der Medizin	20
3.1.3 Anwendung beim autonomen Fahren	22
3.1.4 Anwendung im Alltag	22
3.2 Einsatz der ML-Bilderkennung im Bereich von Zerspanungswerkzeugen	23
4 Eigener Ansatz	25
4.1 Image Classification Werkzeugart	25
4.1.1 Datenbasis erstellen	25
4.1.2 Python-Programmablauf	26
4.2 Identifizierung von abrasivem Werkzeugverschleiß	30

4.2.1 Datenbasis erstellen	30
4.2.2 Python-Programmablauf	34
5 Ergebnisse	39
5.1 Implementierung der Bilderkennung mittels Raspberry Pi in einer Werkzeugmaschine	39
5.1.1 Werkzeugklassifizierung	40
5.1.2 Quantitative Werkzeugverschleißerkennung	41
5.1.3 Überprüfung der Genauigkeit der Verschleißflächenmessung	47
6 Bewertung	50
7 Zusammenfassung und Ausblick	51
7.1 Zusammenfassung	51
7.2 Ausblick	51
Abbildungsverzeichnis	I
Tabellenverzeichnis	IV
Literaturverzeichnis	V

Abkürzungen

Formelzeichen	Größe	Einheit
AI	Artificial Intelligence	-
ML	Machine Learning	-
DL	Deep Learning	-
CNN	Convolutional Neural Network	-
FCN	Fully Convolutional Network	-
GPU	Graphics Processing Unit	-
CPU	Central Processing Unit	-
TPU	Tensor Processing Unit	-
API	Application Programming Interface	-
CV	Computer Vision	-
WSP	Wendeschneidplatte	-
VB	Verschleißmarkenbreite	-
OP	Overall Pixel Accuracy	-
$J(\theta_0, \theta_1)$	Cost Function	-
$h_\theta(x_i)$	Hypothesis Function	-
θ_i	Parameter des Modells	-
y_i	Zielvariable	-
m	Anzahl an Trainingsbeispielen	-
i	Laufindex	-
dim_s	Größe der betrachteten Schicht	-
dim_F	Größe des Filters	-
dim_{SF}	Resultierende Größe der Schicht nach der Faltung	-
dim_{SP}	Größe der Padding Schicht	-
TP	True Positives	-
TN	True Negatives	-
FP	False Positives	-
FN	False Negatives	-
α	Lernrate	-

1 Einleitung

1.1 Motivation

Der Verschleiß eines Werkzeugs während der Bearbeitung eines Werkstücks ist eine der größten Störgrößen im Herstellungsprozess, da dieser sich direkt auf die Qualität des Produktes auswirkt. In Zeiten immer höherer Anforderungen an die Genauigkeit ist es notwendig, frühzeitig verschlissene Werkzeuge zu identifizieren und auszutauschen. Die konventionelle Verschleißerkennung kann diesen Anforderungen nicht mehr gerecht werden. Dabei werden die Werkzeuge in der Regel aus der Maschine ausgebaut, extern gemessen und wieder eingebaut, wodurch Ungenauigkeiten in der Maßhaltigkeit beim Zurückbauen auftreten können. Oftmals wird der Verschleiß mit dem Auge diagnostiziert, was zwar keine Ungenauigkeit durch das Umbauen hervorruft, jedoch den Verschleiß nur qualitativ klassifiziert. Dadurch entstehen Risiken beim Bearbeitungsprozess, wenn ein Werkzeug trotz fortgeschrittenem Verschleiß weiterverwendet wird.

Industrie 4.0 läutet ein neues Zeitalter in der industriellen Produktion ein. Damit verbunden ist die möglichst vollständige Automatisierung der Produktion. Dies ist nur durch die Vernetzung der Maschinen und durch neue datenbasierte Verarbeitungstechniken möglich. In den letzten Jahren fokussierten sich viele Forscher auf den Bereich Artificial Intelligence (AI), Machine Learning (ML) und Deep Learning (DL). Deep Learning stellt dabei die Grundlage von modernen Bilderkennungssystemen zur Verfügung, die es ermöglicht, Objekte zu erkennen und zu klassifizieren.

Deep Learning bietet das Potential, durch kamerabasierte Bilderkennung Werkzeuge nach ihrer Art zu klassifizieren und den Verschleiß eines Werkzeugs automatisch zu erkennen und zu evaluieren. Mithilfe einer Werkzeugklassifizierung können fehlerhaft in die Maschine implementierte Werkzeuge erkannt werden, um so die Sicherheit zu erhöhen und Kosten durch Beschädigungen zu minimieren. Außerdem können Informationen aus der Werkzeugklassifizierung in der Anwendung von Verschleißerkennungsprogrammen genutzt werden. Die Verschleißerkennung stellt in der Industrie 4.0 eine bedeutsame Methode mit großem wirtschaftlichem Faktor dar. Dabei kann durch eine akkurate Verschleißerkennung die Standzeit eines Werkzeugs deutlich gesteigert werden, da durch Messen des Verschleißes das Werkzeug nicht zu früh ausgetauscht wird. Damit können maßgeblich Kosten eingespart werden. Ein weiterer Faktor ist die Prozesssicherheit. Mithilfe einer Verschleißerkennung kann verhindert werden, dass zu stark verschlissene Werkzeuge im Einsatz sind. Dadurch werden Qualitätsminderungen und Nachbearbeitungen verhindert. Eine weitere Anwendungsmöglichkeit findet die Verschleißerkennung in der Forschung. Um neue Beschichtungsarten zu testen, wird eine große Anzahl an Schnittversuchen zur Verschleißanalyse durchgeführt. Eine automatische Verschleißerkennung kann dabei die Zeit der Durchführung der Versuche deutlich verkürzen, indem nicht jedes einzelne Bild manuell ausgewertet werden muss.

Die Verschleißerkennung mittels Machine Learning verspricht damit eine Steigerung der Wirtschaftlichkeit mit gleichzeitiger Steigerung der Prozesssicherheit und der Qualität.

1.2 Zielsetzung

In der vorliegenden Arbeit geht es darum, mithilfe von Machine Learning und Image Processing Ansätzen eine Werkzeugklassifizierung und quantitative Werkzeugverschleißerkennung durchzuführen. Ziel ist es, eine Verbesserung im autonomen Produktionsprozess zu erreichen. Dabei wird erwartet, dass die Werkzeugklassifizierung eine hohe Trefferquote erzielt, um so durch die richtige Klassifizierung die Prozesssicherheit zu erhöhen. Ein weiteres Ziel besteht darin, einen Semantic Segmentation Ansatz zu verwenden, um den Werkzeugverschleiß quantitativ zu identifizieren. Diese Art der Messung kann das manuelle Messen durch den Menschen ersetzen. Dabei wird erwartet, dass durch eine Flächenmessung im Gegensatz zur klassischen Verschleißmarkenbreitenmessung die Genauigkeit erhöht wird, da bei dieser Methode, im Gegensatz zur lokalen Verschleißmarkenbreite, die komplette Verschleißfläche eines Werkzeugs betrachtet wird. Insgesamt soll diese Arbeit dazu beitragen, einen Überblick darüber zu geben, wie neue Techniken auf Basis von Image Processing und Machine Learning mit klassischen Bearbeitungsprozessen kombiniert werden können, um so zu einer Steigerung der Prozessleistung und einer Verringerung der Maschinenstillstandzeiten beizutragen.

1.3 Aufbau der Arbeit

Da die meisten Fachbegriffe auf Englisch sind, werden die englischen Fachbegriffe, anstatt der deutschen Fachbegriffe in dieser Arbeit verwendet. Diese Arbeit ist in sieben Kapitel gegliedert. Im ersten Kapitel geht es um die Einleitung in das Thema „Bildverarbeitungsbasierte Überwachung von Zerspanungswerkzeugen mithilfe von Machine Learning“. In Kapitel zwei werden die benötigten Grundlagen zum Verständnis der Machine Learning Ansätze eingeführt. Kapitel drei gibt einen Überblick zum aktuellen Stand der Technik. Dabei wird aufgeführt, in welchen aktuellen Anwendungsbereichen ML-Bilderkennungsansätze verwendet werden. Zudem werden aktuelle Forschungen zur ML-Verschleißidentifizierung dargestellt. In Kapitel vier wird der eigene Lösungsansatz vorgestellt. Es wird darauf eingegangen, wie Datenbasen für die ML-Bilderkennung erstellt werden können und wie der jeweilige Programmablauf für die Werkzeugklassifizierung und quantitative Verschleißerkennung funktioniert. Dazu werden alle nötigen Informationen zum Training und der Auswertung aufbereitet. Das fünfte Kapitel stellt die Ergebnisse des eigenen Ansatzes dar. Dabei werden die Bilderkennungsprogramme mittels Raspberry Pi in eine Werkzeugmaschine implementiert und angewendet. Zudem wird ein Überblick über die Genauigkeit und Funktionalität der Programme gegeben. Anschließend werden die Ergebnisse im sechsten Kapitel bewertet. In Kapitel sieben wird eine Zusammenfassung der Arbeit und ein Ausblick über mögliche Weiterentwicklungen dieser Thematik präsentiert.

2 Grundlagen

2.1 Werkzeugverschleiß

Beim Zerspanungsprozess werden die aktiven Teile der Werkzeuge mechanisch, thermisch und chemisch beansprucht. Diese Beanspruchungen können, wie beim Drehen, zeitlich konstant sein oder, wie beim Fräsen, zeitlich wechselnd sein.

Je nach Beanspruchung resultieren unterschiedliche Verschleißmechanismen, die unterschiedliche Verschleißformen bedingen. Zu den Verschleißmechanismen zählen Abrasion, Adhäsion und Triboochemischer Verschleiß.

Abrasion ist die Folge der Gleitung zwischen den Wirkpartnern durch harte Bestandteile im Werkstoff. Diese Form der Verschleißursache ist ein rein mechanischer Vorgang. Dies kann durch hohe Oberflächentemperaturen des Schneidkeils auf der Span- und Freifläche begünstigt werden, indem sich eine Schneidstoffentfestigung einstellt (Denkena & Tönshoff, 2011).

Adhäsion beschreibt das Übergehen von Teilchen des Schneidstoffs auf den Span, hervorgerufen durch atomare Bindungskräfte an Mikrokontaktstellen. Dabei lassen sich zwei Vorgänge unterscheiden:

1. Kaltverschweißung (Pressschweißung) am Schneidkeil. Die damit gebildeten Ablagerungen werden später aus dem Schneidstoff herausgerissen, was mit Stoffverlust verbunden ist.
2. Schneidstoffteilchen verschweißen direkt mit der Schnittfläche und der Spanunterseite und werden dabei getrennt (Denkena & Tönshoff, 2011).

Triboochemischer Verschleiß kann auf zwei Ursachen zurückgeführt werden. Diese sind Diffusion und Oxidation. Mit steigender Temperatur nimmt die Beweglichkeit der Moleküle und Atome des Werkstoffes zu. Dadurch kann es zu thermisch aktiverter Diffusion von Bestandteilen des Schneidstoffs in dem Werkstoff oder umgekehrt kommen. Aufgrund der chemischen Reaktion der eingewanderten Teilchen oder der Entfernung der Teilchen können weiche Schichten entstehen, die abgetragen werden (Denkena & Tönshoff, 2011). Oxidation ist eine Folge hoher Temperaturen, die vor allem am Rand der Kontaktzone des Werkzeugs und des Werkstoffes zu Kerben und Ausbrüchen führt (Bartenschlager et al., 2013).

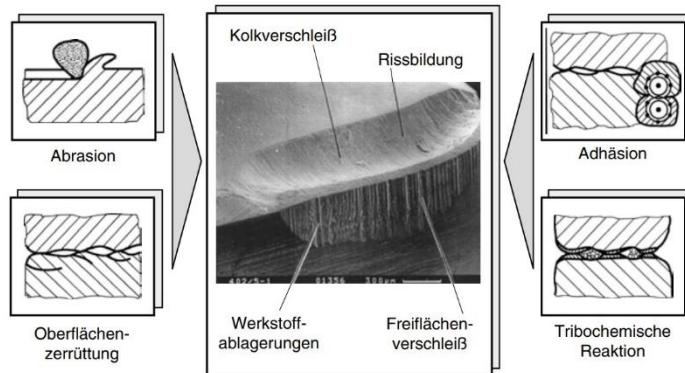


Abbildung 1: Verschleißmechanismen und -Formen (Denkena & Tönshoff, 2011)

Die von den Verschleißmechanismen hervorgerufenen Verschleißformen können Freiflächenverschleiß, Kolkverschleiß, Kantenverschleiß und Schneidkantenbruch sein.

Freiflächenverschleiß ist an den Freiflächen von Haupt- und Nebenschneide zu finden. An diesen Flächen gleitet die Schnittfläche des Werkstücks vorbei. Dadurch entsteht durch die beschriebenen Verschleißmechanismen ein Abrieb, der in einer Verschleißfläche resultiert. Die Ausdehnung dieser Verschleißfläche wird als Verschleißmarkenbreite VB bezeichnet.

Beim Kolkverschleiß bildet sich ein Muldenförmiger Abtrag von Schneidstoff auf der Spanfläche aus. Dadurch wird die Schneidkante geschwächt und erhöht die Gefahr von Schneidkantenausbrüchen. (Denkena & Tönshoff, 2011)

Ein Kantenverschleiß kann durch unterbrochene Schnitte hervorgerufen werden. Wenn der Schneidstoff zu spröde oder den Bearbeitungsanforderungen nicht gewachsen ist, kann es zum Schneidkantenbruch führen. Zudem kann durch nicht rechtzeitiges Wechseln des Werkzeugs aufgrund zu hohen Verschleißes ebenfalls ein Schneidkantenbruch verursacht werden (Bartenschlager et al., 2013).

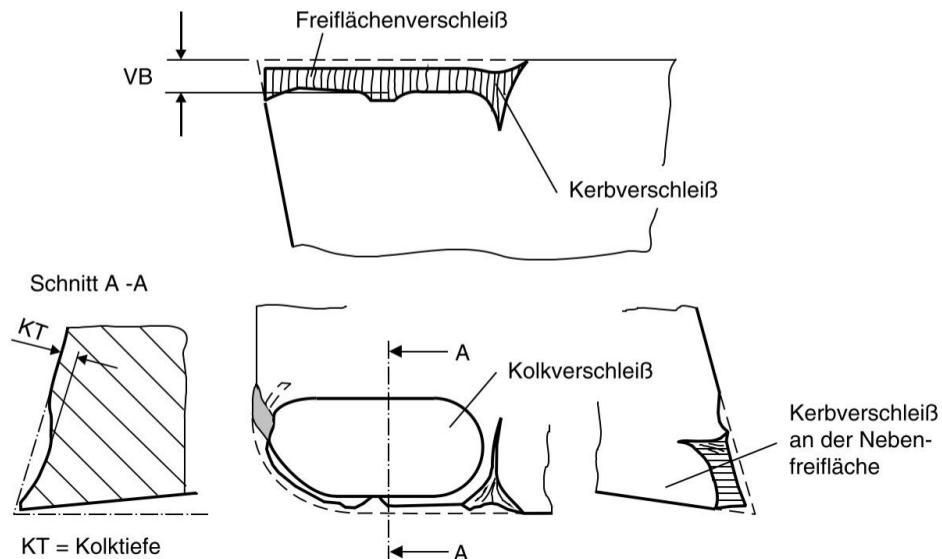


Abbildung 2: Verschleißformen beim Drehen (Denkena & Tönshoff, 2011)

2.2 Machine Learning

2.2.1 Unterschied von Artificial Intelligence, Machine Learning und Deep Learning

Eine klare Abgrenzung zwischen Artificial Intelligence, Machine Learning und Deep Learning gibt es nicht, da die Übergänge fließend sind. Artificial Intelligence ist ein Teilgebiet der Informatik. Sie kann Informationen aus Eingabedaten erkennen und damit menschliche kognitive Fähigkeiten imitieren. Diese Intelligenz kann durch programmierte Abläufe oder Machine Learning erzeugt werden (Rudolph, 2021). Ein Beispiel dafür ist das Finden eines Substantivs in einem Satz, indem hartkodierte Grammatikregeln verwendet werden. Ein weiteres Beispiel ist, durch Verwendung von *if* und *else* Bedingungen, das Umfallen eines Roboters zu verhindern. Diese Systeme werden jedoch heutzutage als schwach intelligent angesehen (Bhalley, 2021). Im Vergleich dazu lernt beim Machine Learning ein Algorithmus durch Wiederholung eine Aufgabe selbstständig auszuführen. Dabei wird kein Lösungsweg modelliert, sondern der Algorithmus lernt selbstständig die Struktur der Daten zu erkennen, indem er sich an einem vorgegebenem Gütekriterium und dem Informationsgehalt der Daten orientiert. Beim Deep Learning werden Lernalgorithmen verwendet, die nach dem Vorbild der Nervenzellenverbindungen im menschlichen Gehirn entwickelt wurden. Man spricht von Neuronalen Netzwerken (Neural Networks), die ähnlich zu den Neuronen und Synapsen im Gehirn über mehrere Reihen von Datenknoten und gewichteten Verbindungen verfügen (Rudolph, 2021).

Deep Learning ist ein Teil von Machine Learning und Machine Learning ist ein Teil von Artificial Intelligence. Abbildung 3 zeigt schematisch, wie diese Teilgebiete untergeordnet sind und deren Eigenschaften.

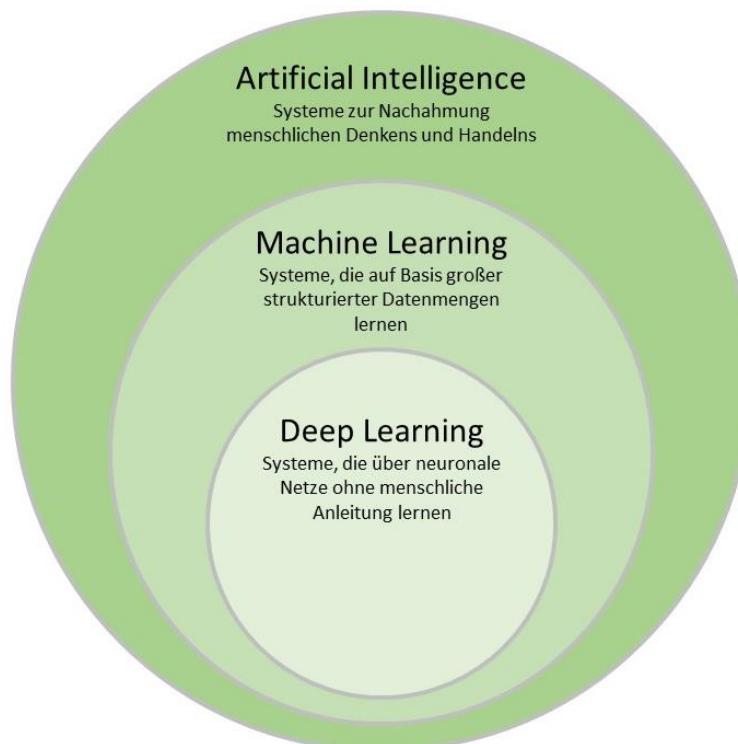


Abbildung 3: Einordnung Artificial Intelligence, Machine Learning und Deep Learning (IONOS Digital Guide, 2020)

2.2.2 Neuronale Netzwerke (Neural Networks)

Neural Networks imitieren das Verhalten des menschlichen Gehirns. Sie ermöglichen es Computern, Muster zu erkennen und allgemeine Probleme im Bereich Artificial Intelligence, Machine Learning und Deep Learning zu lösen. Dabei sind Neural Networks ein Teil des Machine Learnings und der Kern aller Deep Learning Algorithmen (IBM Cloud Education, 2020).

Das einfachste Neural Network das existiert ist das Perzeptron. Es besteht aus einem oder mehreren Inputwerten x , die über Gewichte w mit dem Neuron verbunden sind. Das Neuron enthält eine Summation und eine Activation Function, um einen Output y zu erzeugen. Der Bias Term b kann die Activation Function entweder nach links oder rechts auf der x-Achse verschieben, um ein Versetzen auf positive oder negative Werte zu ermöglichen. Das Perzeptron hat eine Input Layer und eine Hidden Layer, die gleichzeitig als Output Layer fungiert (Bergs et al., 2020).

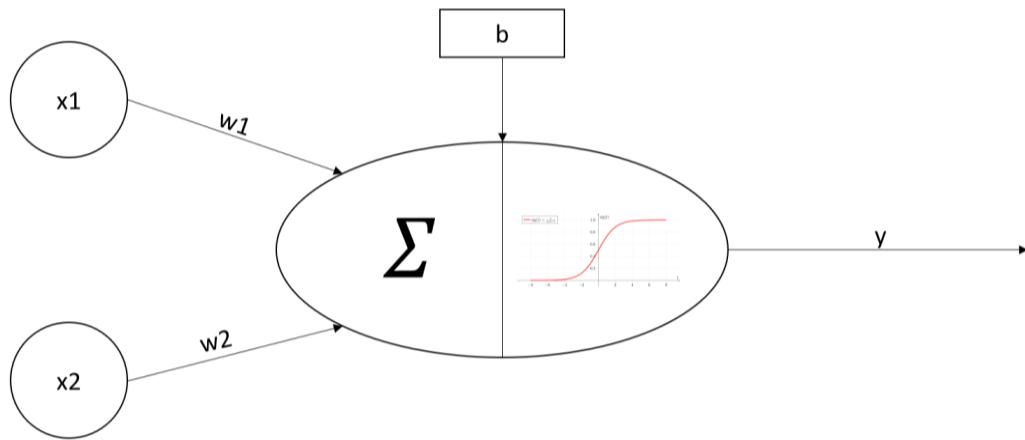


Abbildung 4: Schematische Darstellung eines Perzeptrons (Bergs et al., 2020)

Ein Künstliches Neuron berechnet die gewichtete Summe der Inputs, addiert den Bias-Term hinzu und entscheidet, ob es den Wert 1 oder 0 weitergibt.

$$Y = \sum (\text{weight} * \text{input}) + \text{bias} \quad \text{Formel 2.2.2.1}$$

Die Activation Function ermöglicht es dem Netzwerk, beliebige glatte Funktionen anzunähern. Diese ist immer nichtlinear, beziehungsweise stückweise linear, da andernfalls das Netzwerk nur eine Folge von linearen Transformationen wäre. Vergleicht man dies in Analogie zu biologischen Neuronen, steht die Activation Function für „feuern“. Dies bedeutet, dass, wenn ein bestimmter Schwellenwert des Inputs überschritten wird, abhängig davon ein Output ausgegeben wird (Choo et al., 2020).

Die typischen Activation Functions sind:

Sigmoid Function:

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

Formel 2.2.2.2

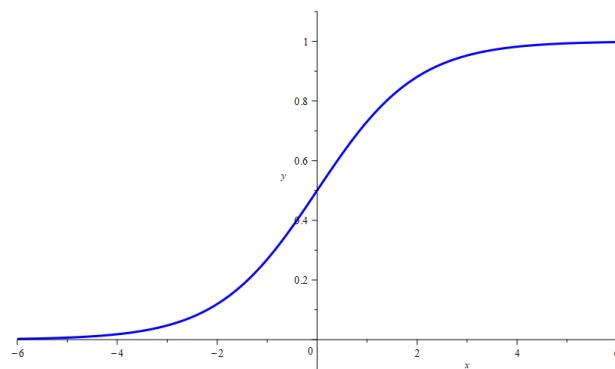


Abbildung 5: Sigmoid Function

Tanh Function:

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Formel 2.2.2.3

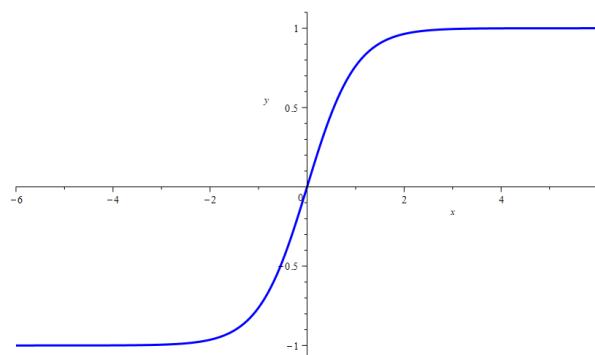


Abbildung 6: Tanh Function

ReLU (rectified linear unit) Function:

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

Formel 2.2.2.4

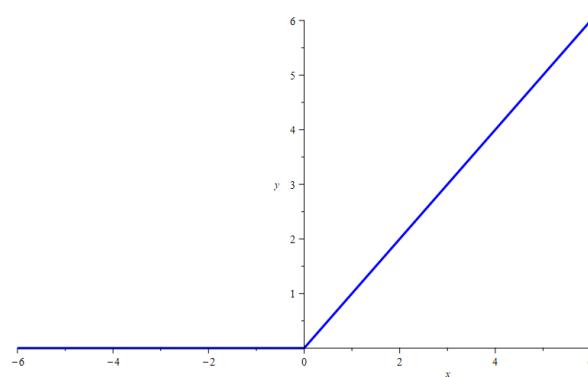


Abbildung 7: ReLu Function

Ein Multilayer Perzeptron ist eine Verschaltung mehrerer Perzeptronen zu einem Netzwerk mit mindestens einem Hidden Layer und einem Output Layer, die mehrere Perzeptronen parallel haben können. Ein einzelnes Perzeptron kann nur unkomplizierte Aufgaben lernen. Um komplexere Aufgaben lernen zu können, werden komplexere Multilayer Perzeptronen benötigt. Ein Neural Network mit mehr als zwei Hidden Layer wird als Deep Neural Network bezeichnet (Bergs et al., 2020).

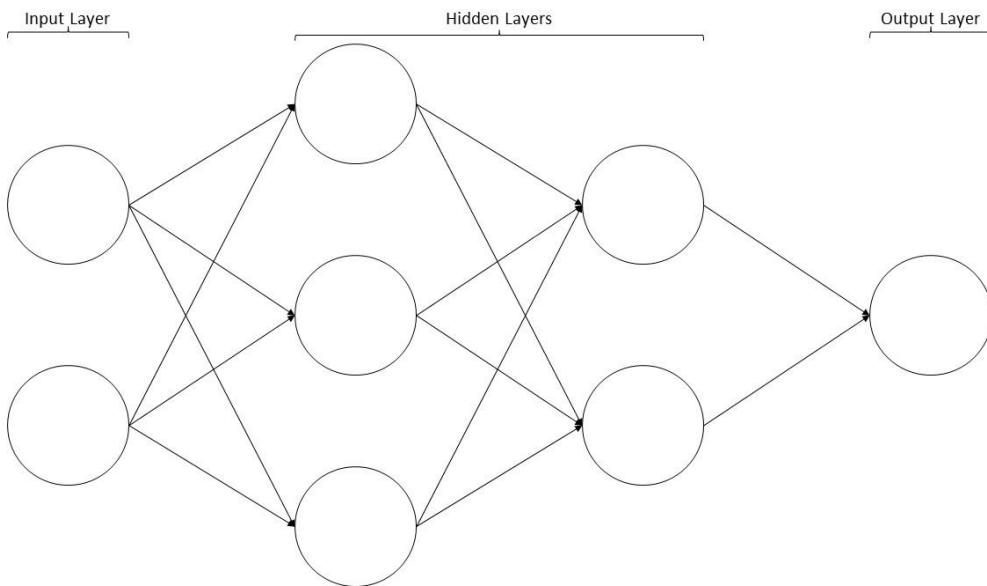


Abbildung 8: Schematische Darstellung eines Deep Neural Networks (Bergs et al., 2020)

Jeder einzelne Knoten repräsentiert ein künstliches Neuron mit einer Summation und einer Activation Function, um die gewichteten Inputs zu verarbeiten.

2.2.3 Supervised und Unsupervised Learning

Beim Supervised Learning geht man davon aus, dass es eine Korrelation zwischen Input und Output gibt. Es ist von Anfang an bekannt, wie der korrekte Output aussehen soll. Dadurch können durch den Vergleich der modellbasierten Prognosen mit den korrekten Outputs Prognosefehler erkannt und das Modell schrittweise optimiert werden (Brühl, 2019). Supervised Learning kann aufgeteilt werden in Regression und Klassifizierung. Bei einem Regressionsproblem wird versucht, Ergebnisse innerhalb einer kontinuierlichen Ausgabe vorherzusagen, was bedeutet, EingabevARIABLEN einer kontinuierlichen Funktion zuzuordnen. Im Gegensatz dazu wird bei einem Klassifizierungsproblem versucht, die EingabevARIABLEN in diskreten Kategorien abzubilden, um damit diskrete Ausgaben zu erhalten (Ng, 2021).

Bei einer Betrachtung des Unsupervised Learnings stellt man fest, dass dabei wenig oder keine Vorstellung vorhanden ist, wie die Ergebnisse aussehen sollen. Dabei werden Strukturen aus den Daten abgeleitet, von denen die Wirkung der Variablen nicht bekannt ist. Um Strukturen ableiten zu können, werden die Daten, basierend auf den Beziehungen zwischen den Variablen, in den Daten gruppiert. Beim Unsupervised Learning gibt es kein Feedback basierend auf den Vorhersageergebnissen (Ng, 2021).

2.2.4 Backpropagation

Backpropagation ist ein Verfahren des Supervised Learnings, um die Gewichtungen (Weights) anzupassen. Die Änderung der Weights wird bestimmt, indem die Abweichung der berechneten Ausgabe zur erwarteten Ausgabe berechnet wird. Die Anpassung wird ausgehend vom Output-Layer in Richtung Input-Layer schichtweise vorgenommen (Lämmel & Cleve, 2020). Während des Trainings durchläuft eine Reihe an Daten mit dem Backpropagation-Algorithmus als Input das Netz. Der resultierende Output wird dann mit dem gewünschten Output verglichen, wodurch ein Fehler entsteht. Der Fehler wird dann zurück propagiert, Schicht für Schicht, bei gleichzeitigem Aktualisieren der Weights um den Betrag, den sie zum Fehler beigetragen haben (Bergs et al., 2020).

2.2.5 Cost Function

Beim Machine Learning wird versucht, aus verfügbaren Daten eine Funktion zu finden, die am besten die Inputs zu den Outputs zuordnet. Dies geschieht approximativ. Ein Beispiel für ein Modell, das die Zielfunktion approximiert und Zuordnungen von Inputs zu Outputs durchführt, wird als Hypothesis Function bezeichnet (Jason Brownlee, 2019). Um die Genauigkeit der Hypothesis Function zu messen, wird die Cost Function (auch Loss Function genannt), benutzt. Diese nutzt die durchschnittliche Differenz aller Ergebnisse der Hypothese mit den Inputs x und den tatsächlichen Outputs y . Ein Beispiel für eine Cost Function ist die „Squared error function“ (Formel 2.2.5.1) (Ng, 2021).

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x_i) - y_i)^2 \quad \text{Formel 2.2.5.1}$$

mit

$J(\theta_0, \theta_1)$ = Cost Function

$h_\theta(x_i) = \theta_0 + \theta_1 x$ = Hypothesis Function

θ_i = Parameter des Modells

y_i = Zielvariable

m = Anzahl an Trainingsbeispielen

i = Laufindex

2.2.6 Gradient Descent

Ziel des Gradient Descent ist es, die Cost Function $J(\theta_0, \theta_1)$ zu minimieren. Zuerst werden θ_0 und θ_1 auf einen bestimmten Wert festgelegt. θ_0 und θ_1 werden dann dauerhaft verändert (Formel 2.2.6.1), um $J(\theta_0, \theta_1)$ zu minimieren und bestenfalls an ein Minimum zu gelangen.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1), \quad j \in (0; 1) \quad \text{Formel 2.2.6.1}$$

Wird beispielsweise an einem bestimmten Punkt in der Funktion gestartet, wie in Abbildung 9 dargestellt, wird zunächst die Ableitung der Cost Function gebildet. Die Steigung der Tangente gibt die Richtung an. Anschließend werden die Schritte in Richtung des steilsten Abstiegs gemacht, während die Größe der jeweiligen Schritte durch die Lernrate α bestimmt wird. Je nachdem, wo sich der Startpunkt befindet, ist es möglich bei unterschiedlichen Minima zu landen,

wie Abbildung 9 zeigt. Abbildung 10 stellt den Prozess des Gradient Descent und wie sich die Hypothesis Function in Abhängigkeit dazu verändert, bis ein Minimum erreicht ist, dar. Dieser Prozess wird anhand des Beispiels, wie sich Häuserpreise in Relation zur Wohnfläche vorhersagen lassen, verdeutlicht. (Ng, 2021).

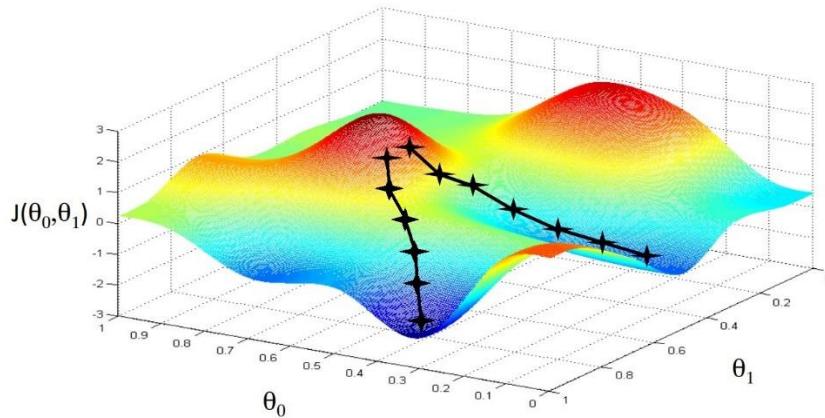


Abbildung 9: Zwei mögliche Wege des Gradient Descent (Ng, 2021)

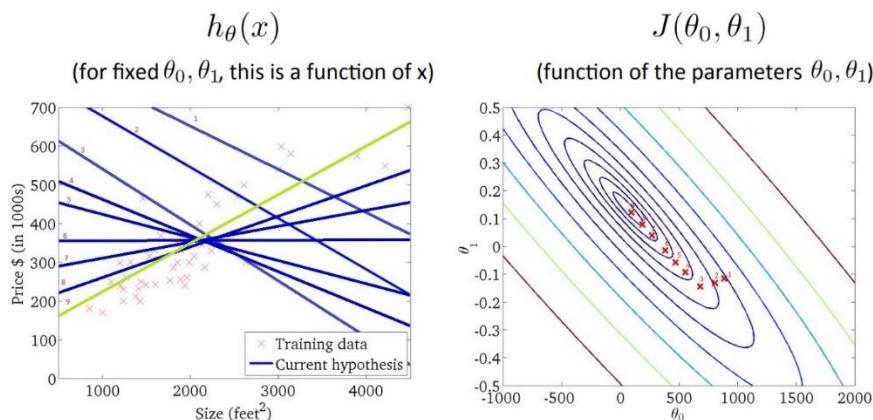


Abbildung 10: Zusammenhang zwischen Cost Function und Hypothesis Function (Ng, 2021)

2.2.7 Overfitting

Stimmt ein statistisches Modell genau mit den Trainingsdaten überein, wird dies Overfitting genannt. Dies verursacht das Problem, dass der Algorithmus nicht mehr auf Daten außerhalb des Trainingsdatensatzes reagieren kann. Wenn ein Modell zu lange trainiert wird oder es zu komplex ist, kann es sein, dass das Modell von Rauschen oder unrelevanten Informationen lernt. Overfitting ist dadurch zu erkennen, dass die Trainingsdaten eine kleine Fehlerrate haben, während die Validierungsdaten eine hohe Fehlerrate aufweisen. Eine Möglichkeit Overfitting vorzubeugen, ist den Trainingsprozess früher zu unterbrechen, auch „early stopping“ genannt. Zusätzlich kann die Komplexität des Modells verringert werden, indem weniger relevante Inputs eliminiert werden. Wird der Prozess zu früh unterbrochen und schließt damit zu viele erhebliche Merkmale aus, kann dies zu Underfitting führen. Dies geschieht, wenn das Modell nicht lange genug trainiert wurde oder die Input Variablen nicht aussagekräftig genug sind, um eine aussagekräftige Vorhersage zu treffen. Ziel ist es, das Optimum zwischen Underfitting und Overfitting zu finden (IBM Cloud Education, 2021).

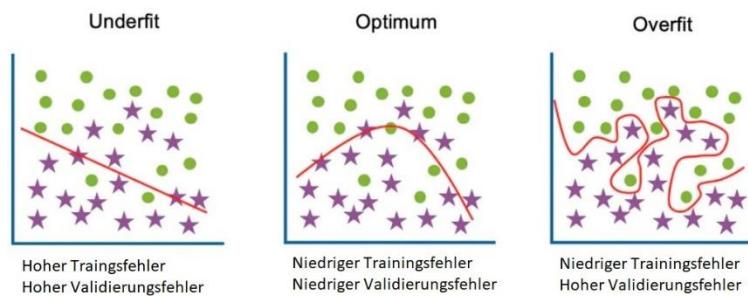


Abbildung 11: Vergleich zwischen Underfitting, Optimum und Overfitting (IBM Cloud Education, 2021)

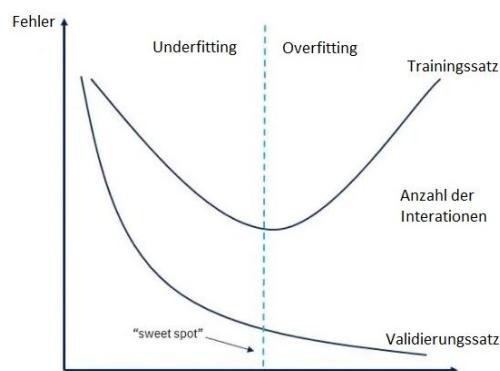


Abbildung 12: Optimum zwischen Overfitting und Underfitting (IBM Cloud Education, 2021)

Um Overfitting zu vermeiden, gibt es mehrere Möglichkeiten:

- Early stopping: Unterbricht das Training, bevor das Modell anfängt, von Rauschen zu lernen. Gefahr: Underfitting (IBM Cloud Education, 2021).
- Mit mehr Daten trainieren: Wenn mehr Daten zum Trainieren zur Verfügung stehen, wird die Genauigkeit des Modells erhöht, indem mehr Möglichkeiten vorhanden sind, um die dominanten Beziehungen zwischen Input und Output herauszufiltern (IBM Cloud Education, 2021).
- Data Augmentation: Aus einem Bild können beispielsweise durch Drehen und Strecken mehrere Bilder erstellt und somit der Datensatz erweitert werden (Shorten & Khoshgoftaar, 2019).
- Feature Selection: Feature Selection beschreibt den Prozess des Erkennens der wichtigsten Merkmale eines Trainingssatzes und der Eliminierung der unrelevanten und redundanten Merkmale, um das Modell zu vereinfachen (IBM Cloud Education, 2021).
- Regularization: Wenn nicht bekannt ist, welche Merkmale beim Feature Selection Prozess eliminiert werden können, hilft Regularization weiter. Regularization fügt den Input Parametern mit größeren Koeffizienten einen „Nachteil“ hinzu. Dadurch wird der Betrag der Varianz im Modell verringert (IBM Cloud Education, 2021).

2.2.8 Convolutional Neural Network

Fully-Connected Networks, bei denen alle Neuronen in einem Layer mit allen Neuronen im nächsten Layer verbunden sind, können zwar für eine Bildverarbeitung eingesetzt werden, beanspruchen jedoch eine lange Zeit zum Trainieren. Selbst ein graues Bild mit geringer Auflösung von 32x32 Pixeln und 1024 Neuronen im ersten Layer und der gleichen Anzahl in zwei

weiteren Hidden Layer führen zu mehr als zwei Millionen trainierbaren Parametern. Eine höhere Auflösung und mehr Layer würden dies undurchführbar machen. Deshalb werden für die Bildverarbeitung Convolutional Neural Networks (CNN) eingesetzt. Diese bieten durch mehrere Hidden Layer Typen eine Lösung für das Effizienzproblem (Bergs et al., 2020).

Ein CNN ist ein vorwärts gerichtetes Neural Network mit einer besonderen Struktur. Es besitzt mehrere Schichten und kann dadurch Muster besser erkennen als einfache, vorwärts gerichtete Neural Networks. CNNs sind vorrangig für die Bilderkennung entwickelt worden, da normale vorwärts gerichtete Neural Networks schnell an ihre Leistungsgrenzen stoßen. Zusätzlich können normale vorwärts gerichtete Neural Networks Objekte nicht mehr erkennen, wenn diese im Eingabebild verschoben sind. Diese Probleme können durch eine Dimensionsreduktion, die innerhalb des CNNs stattfindet, behoben werden. Die grundlegende Operation ist die Convolution Operation (Faltung) (Lämmel & Cleve, 2020).

Die Faltungsoperation wird im Convolutional Layer eingesetzt. Diese ist eine von drei Arten von Schichten, die in einem CNN zusammengeschaltet werden:

1. Convolutional Layer (Faltungsschicht)
2. Pooling Layer (Zusammenfassung)
3. Fully-Connected Layer (voll-vernetzte Schichten zur Identifikation oder Klassifikation)

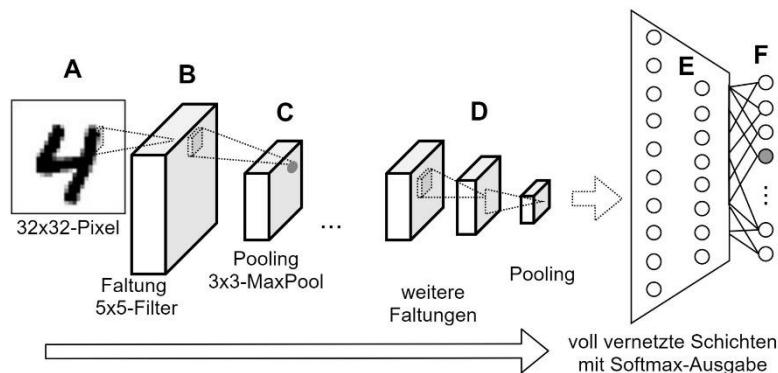


Abbildung 13: Struktur eines Convolutional Neural Networks (Lämmel & Cleve, 2020)

A: Eingabe-Bild: In der Regel sind dies RGB Bilder und werden dadurch in drei RGB-Farb-Ebenen aufgeteilt. Bei graufarbigem Bildern besteht die Eingabe nur aus einer Ebene.

B: Convolutional Layer: Anwendung eines Filters auf die davorliegende Schicht.

C: Pooling Layer: Die Daten der vorangegangenen Schicht werden stark verdichtet. Bsp.: Bei einem 3x3-Fenster wird nur der Maximalwert des Bereichs weitergeleitet.

D: Die Kopplung aus Convolutional Layer und Pooling Layer kann mehrmals hintereinander geschaltet dem Netz hinzugefügt werden.

E: Ein normales vorwärts gerichtetes Netz beendet die Verarbeitung

F: Im Output Layer wird das Ergebnis der Klassifikation anhand der Neuronen sichtbar

Convolution Layer:

Über das Ausgangsbild wird ein Filter, dessen Dimension deutlich kleiner als die Dimension des Ausgangsbildes ist, Pixel für Pixel geführt. Die Zahlenwerte für die Filter werden zufällig gewählt und dann während des Trainings angepasst.

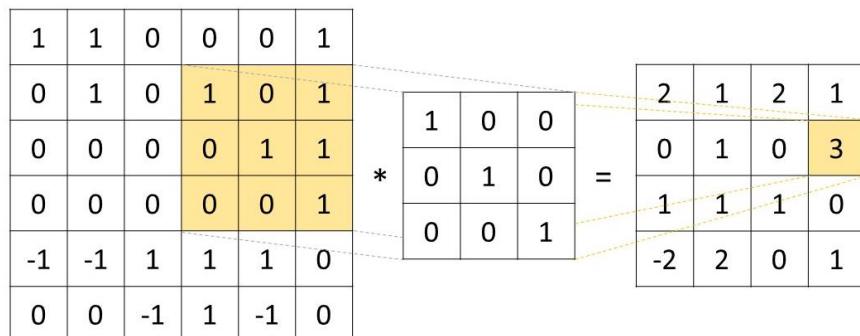


Abbildung 14: Schematische Darstellung einer Convolution Operation (Bergs et al., 2020)

Die Werte im Input-Ausschnitt werden mit den entsprechenden Werten im Filter multipliziert und addiert. Dieser Wert wird in der Ausgabe an die Position übernommen, an der der jeweilige Ausschnitt in der Eingabe positioniert ist. Die resultierende Größe der Schicht nach der Faltung kann mit Formel 2.2.8.1 berechnet werden, wobei dim_s die Größe der betrachteten Schicht, dim_f die Größe des Filters und dim_{sf} die Größe der Schicht nach der Faltung ist (Lämmel & Cleve, 2020).

$$dim_{sf} = dim_s - dim_f + 1 \quad \text{Formel 2.2.8.1}$$

Betrachtet man eine quadratische Schicht mit einer Größe von 28x28 Pixeln, dann besitzt diese nach der Faltung mit einem Filter der Größe 5x5 Pixel die Größe 25x25 Pixel. Durch die Verkleinerung der Schichten während der Faltung entsteht ein Informationsverlust. Dieser kann ausgeglichen werden, indem die Ränder um die Ausgangsschicht mit Werten aufgefüllt werden. Man spricht von Padding. Die Größe der Padding Schicht ist durch die Größe des Filters gegeben (Formel 2.2.8.2) (Lämmel & Cleve, 2020).

$$dim_{sp} = \frac{dim_f - 1}{2} \quad \text{Formel 2.2.8.2}$$

Pooling Layer:

Beim Pooling wird die Schichtgröße, die die Eigenschaften des Originalbilds beinhaltet, stark verkleinert. Die Ausgangsschicht des Convolutional Layers (Feature Map) ist empfindlich gegen unterschiedliche Positionen des gesuchten Objekts im Bild. Dieses Problem tritt auf, wenn die Dimension reduziert wird, aber gleichzeitig die wesentlichen Informationen beibehalten werden. Beim Pooling werden die vorhandenen Eigenschaften summiert und komprimiert weitergegeben. Die neu erzeugte Feature Map ist stabiler gegen unterschiedliche Positionen des Objekts auf dem Bild. Dabei gibt es zwei Verfahren des Poolings: Average Pooling und Max Pooling. Beim Average Pooling wird der Durchschnitt aller Werte des Ausschnitts auf der Feature Map gebildet, während beim Max Pooling nur der Maximalwert des Ausschnitts verwendet wird. Zusätzlich kann mithilfe der Schrittweite die Verkleinerung der Schicht gesteuert werden, wie in Abbildung 15 dargestellt (Lämmel & Cleve, 2020).

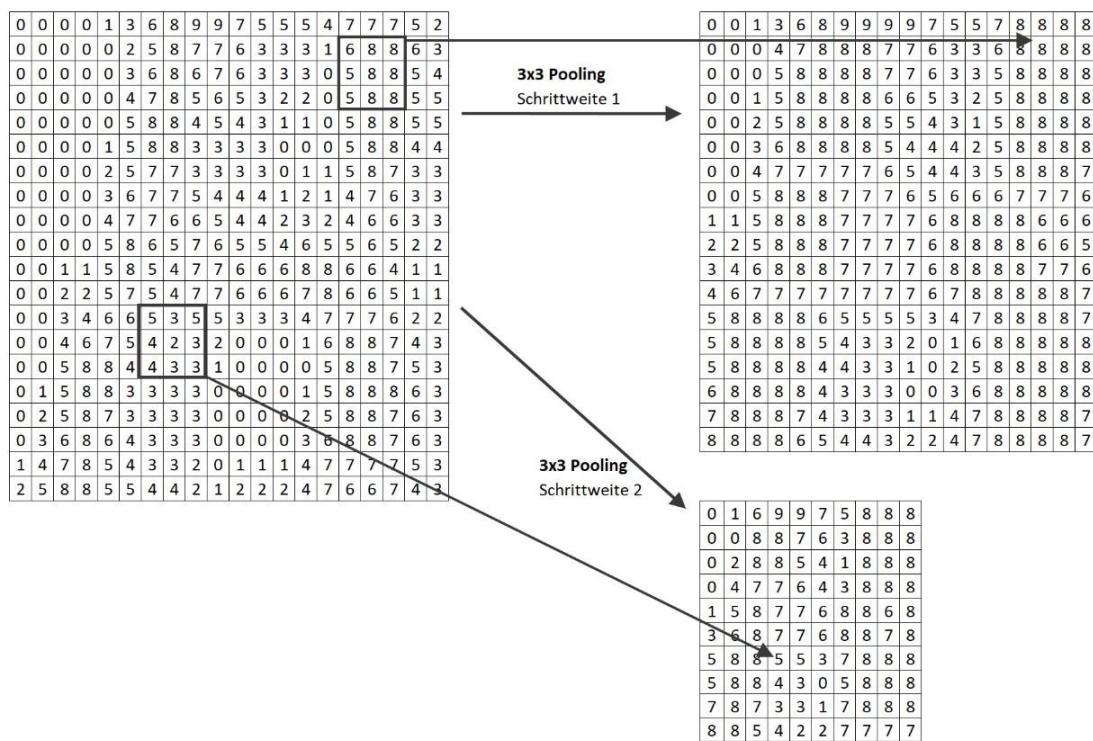


Abbildung 15: Max-Pooling mit unterschiedlichen Schrittweiten (Lämmel & Cleve, 2020)

Fully Connected Layer:

Nachdem das Programm die Convolution Layer und Pooling Layer durchlaufen hat, gelangt es in die Fully Connected Layer. Diese bestehen aus zwei voll vernetzten Hidden Layer und einer voll vernetzten Output Layer. Die Ausgabeschicht entspricht der Zahl der Klassen, die das Netz erkennen soll, wobei die Vorhersage der jeweiligen Klassen nach dem Softmax-Prinzip erfolgt. Dabei werden die Aktivierungen aller Neuronen des Output Layers normiert, um prozentuale Aussagen über die Zugehörigkeit einer Klasse zu treffen. Die Klasse, mit der höchsten prozentualen Zugehörigkeit wird dann vorhergesagt (Lämmel & Cleve, 2020).

Dropout:

Dropout ist eine Technik, um Overfitting zu verhindern und gehört dabei zu den Regularization Methoden. Während des Trainings wird der Output mancher Layer zufällig ausgelassen oder vernachlässigt. Dadurch resultiert für jede Kombination ein unterschiedliches Netzwerk und erreicht dadurch eine robustere Lösung (Verdhan, 2021).

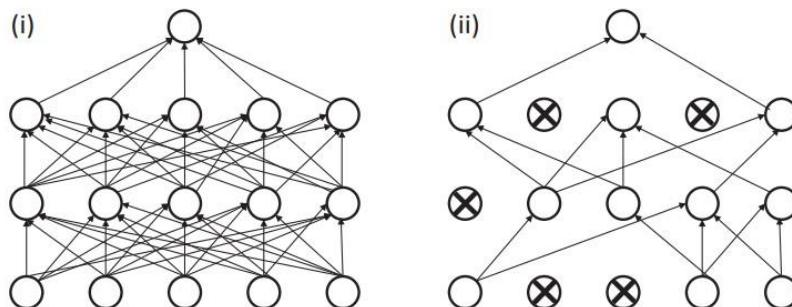


Abbildung 16: Netzwerke ohne Dropout (i) und mit Dropout (ii) (Verdhan, 2021)

2.2.9 Computer Vision

Definition: Computer Vision (CV) bezeichnet Systeme, die Objekte in digitalem Stand- und Bewegtbildmaterial erkennen und entsprechend verarbeiten (Heller, 2020).

Sie ist ein Teilgebiet der AI und orientiert sich an der menschlichen Fähigkeit Bilder zu erfassen, zu verarbeiten und zu analysieren. Die typischen Aufgaben sind (Radeck, 2018):

- Object Classification: Zuordnung von Objekten zu bestimmten Kategorien
- Object Localization: Beschreibung der Position eines Objekts im Bild
- Content Based Image Retrieval: Große Mengen an Daten können unabhängig von den Metadaten (z.b. Dateiname) nach bestimmten Inhalten durchsucht werden.
- Motion Analysis: Bewegungsanalyse
- Reconstruction
- Image Captioning

2.2.10 Tensorflow und Keras

2.2.10.1 Tensorflow

Tensorflow ist eine Softwarebibliothek für Machine Learning und Deep Neural Networks, die vom Google Brain Team entwickelt wurde. Tensorflow kombiniert die Rechenalgebra von Komplizierungsoptimierungstechniken und erleichtert damit die Berechnung vieler mathematischer Ausdrücke, bei denen die Dauer der Berechnung ein Problem darstellt (Zaccone, 2016).

Die Hauptaufgaben sind:

- Definieren, optimieren und die effiziente Berechnung mathematischer Ausdrücke mit mehrdimensionalen Arrays (Tensoren).
- Unterstützung der Programmierung von Deep Neural Networks und Machine Learning Techniken.
- Transparente Nutzung von GPU (Graphics Processing Unit) Berechnung, automatisierte Verwaltung und Optimierung des Speichers und der verwendeten Daten. Der Programm Code kann sowohl durch CPU (Central Processing Unit) oder GPU ausgeführt werden. Gleichzeitig kann Tensorflow selbstständig herausfinden, welche Berechnungen auf der CPU oder der GPU ausgeführt werden sollen.

2.2.10.2 Keras

Keras ist die High-Level API (Application Programming Interface) von Tensorflow 2. Sie ist eine Bibliothek, die mächtige und abstrakte Bausteine für Deep Learning zur Verfügung stellt. Keras ermöglicht es, die Skalierbarkeit und die plattformübergreifenden Möglichkeiten von Tensorflow komplett auszuschöpfen. Dabei kann Keras auf einer TPU (Tensor Processing Unit) oder auf großen GPU-Clustern ausgeführt werden oder ebenso im Browser oder auf mobilen Geräten (*About Keras*, 2021).

2.2.11 Semantic Segmentation

Es gibt mehrere Arten der Bilderkennung: Classification, Classification plus Localization, Object Detection und Image Segmentation. Während bei Classification nur das Objekt klassifiziert werden soll, wird bei Object Detection die Position und die Art der Objekte auf dem Bild erkannt. Image Segmentation versucht zusätzlich eine genaue Grenze um die Objekte in einem Bild zu ziehen.

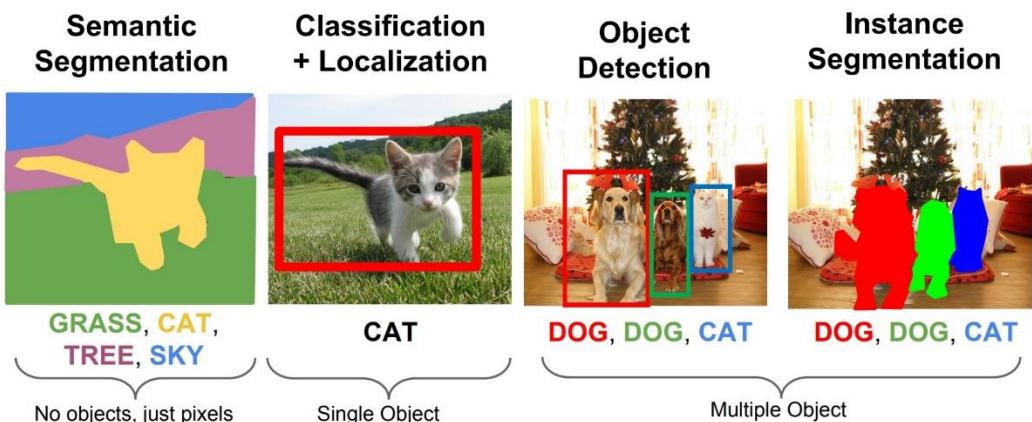


Abbildung 17: Unterschiedliche Aufgaben von Computer Vision (Matcha, 2021)

Image Segmentation ist ein Klassifizierungsproblem auf Pixelbene. Dabei werden die einzelnen Pixel so klassifiziert, dass sie einer bestimmten Klasse angehören. Es gibt zwei Arten von Image Segmentation:

- Semantic Segmentation:

Semantic Segmentation ist die Klassifizierung jedes Pixels, zu einem bestimmten Label. Dabei wird nicht zwischen Objekten unterschieden, die dem gleichen Label angehören. Sind beispielsweise zwei Objekte auf einem Bild, bekommen diese das gleiche Label.

- Instance Segmentation:

Im Vergleich zu Semantic Segmentation wird hier zusätzlich jedes Objekt unterschieden, auch wenn diese der gleichen Klasse angehören. Sind zum Beispiel zwei Hunde auf einem Bild zu sehen, werden im Vergleich zu Semantic Segmentation nicht beide mit „Hund“ gekennzeichnet, sondern mit „Hund rot“ und „Hund grün“ (Matcha, 2021).

2.2.11.1 U-Net

U-Net ist ein Fully Convolutional Network (FCN), das in erster Instanz für den biomedizinischen Einsatz entwickelt wurde (Ronneberger et al., 2015). Ein FCN ist ein CNN ohne Fully Connected Layers und führt dabei nur Faltungen (Convolutions) durch (Long et al., 2014). Der Vorteil der U-Net Architektur ist, dass selbst mit einem kleinen Datensatz präzise Segmentierungen erzeugt werden können. Das U-Net besteht aus einem Contraction Path (Encoder) und einem Expansion Path (Decoder). Jeder dieser Pfade besteht aus mehreren Layern, wobei der Contraction Path der typischen Architektur eines CNNs folgt. Dabei enthält dieser Pfad sich wiederholende Convolutions, Activation Functions, Max Pooling Operations und Dropouts, um die wichtigen Merkmale aus den Eingabebildern zu erfassen. Der Expansion Path kombiniert Low-Resolution Feature Maps mit den räumlichen Informationen aus dem Contraction Path und konstruiert damit eine High-Resolution Segmentation Map. Diese besteht dabei aus mehreren, sich wiederholenden

Layern, bestehend aus Up-Convolution-, Concatenation- (Verkettung), Dropout- und Convolution Operations (Ronneberger et al., 2015).

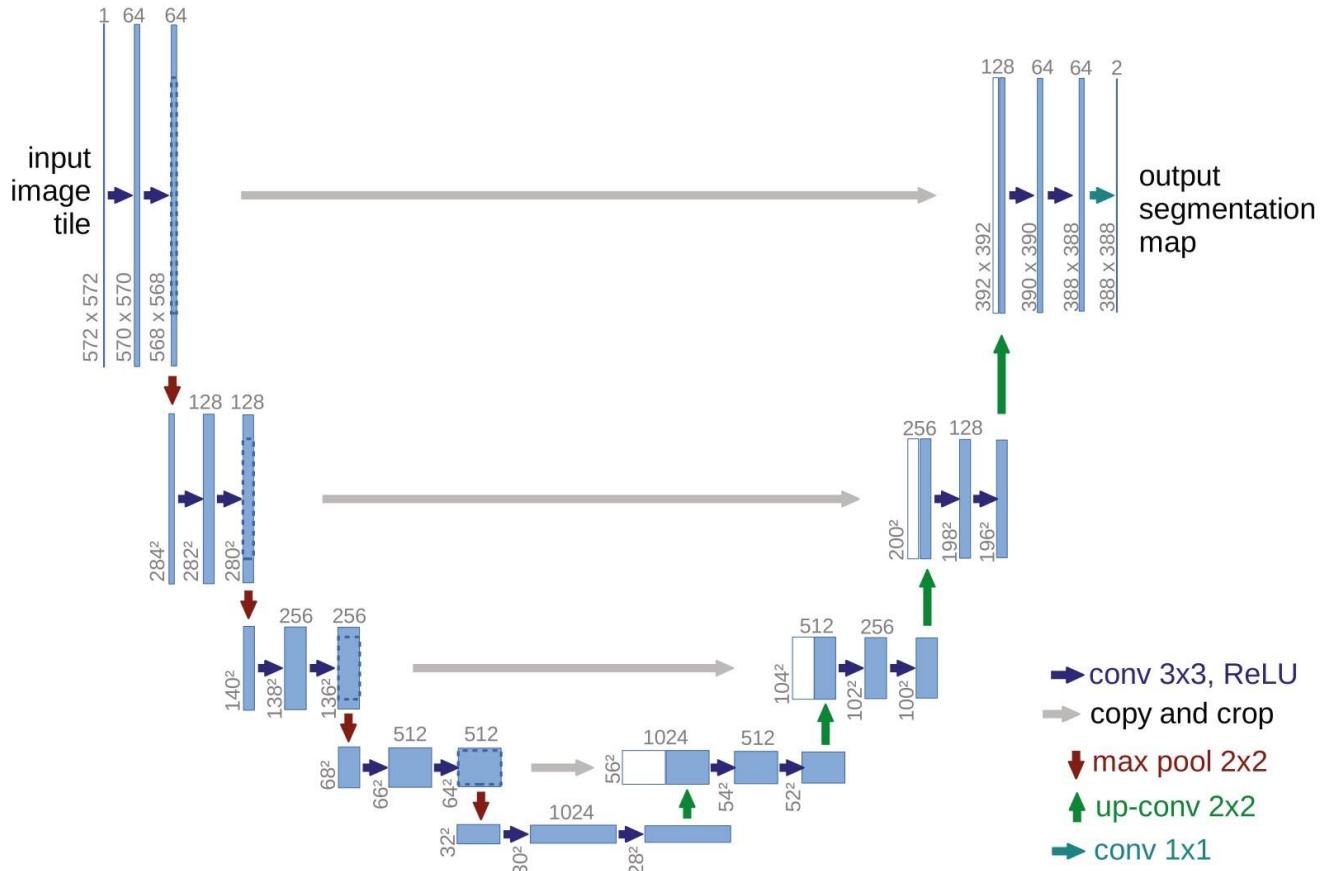


Abbildung 18: U-Net Architektur (Ronneberger et al., 2015)

Jede der blauen Boxen in Abbildung 18 repräsentiert eine Multi-Channel Feature Map. Die meisten Operationen sind 3x3 Convolution Operations gefolgt von einer ReLu Activation Function, die durch die blauen Pfeile dargestellt werden. Die nächste Operation ist eine Max Pooling Operation, die die x, y – Größe der Feature Map reduziert. Diese wird durch den nach unten gerichtetem Pfeil demonstriert. Die Max Pooling Operation propagiert dabei den Maximalwert eines 2x2 Fensters und verdoppelt die Anzahl der Feature Kanäle nach jedem Schritt. Diese Abfolge von Operationen führt zu einer räumlichen Kontraktion, wobei die Information was im Bild zu sehen ist schrittweise zunimmt und gleichzeitig die Information der Position der Objekte im Bild schrittweise abnimmt. Am Ende eines Contraction Paths werden alle Features einem einzigen Vektor zugeordnet. Eine U-Net Architektur zeichnet vor allem aus, dass diese zusätzlich einen Expansion Path enthält, um High-Resolution Feature Maps zu erzeugen. Diese Kombination aus Contraction Path und Expansion Path gibt dieser Architektur die U-Form, nach der das Netz benannt worden ist. Der Expansion Path besteht aus 2x2 Up-Convolution Operations, Concatenations mit High-Resolution Features des Contraction Paths und 3x3 Convolution Operations gefolgt von einer ReLu Activation Function. Bei jeder Concatenation wird die Feature Map jeweils an den Rändern beschnitten, da bei jeder Faltung die Randpixel verloren gehen. Diese Operation sorgt dafür, dass der Verlust der Positionsinformation der Objekte auf dem Bild kompensiert wird, damit Semantic Segmentation möglich ist. Zum Schluss des Expansion Paths wird ein 1x1 Convolution Layer verwendet, um den 64-komponentigen Feature Vektor zur gewünschten Klasse zuzuordnen (Ronneberger et al., 2015).

2.2.12 Evaluation Metrics

2.2.12.1 Overall Pixel Accuracy

Die Overall Pixel Accuracy (OP) ist die unkomplizierteste Methode die Richtigkeit eines Machine Learning Algorithmus zu überprüfen. Dabei wird der Prozentsatz der richtig klassifizierten Pixel angegeben.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \quad \text{Formel 2.2.12.1}$$

Gefahren bei dieser Metrik sind, dass hohe Werte ausgegeben werden, die nichts über die Performance eines Modells aussagen. Besitzt eine Datenbasis beispielsweise ein starkes Ungleichgewicht der Klassen, bei denen 90% der Proben zu Klasse A gehören und 10% der Proben zu Klasse B, dann würde die Accuracy bei 90% liegen, wenn der Algorithmus alle Proben der Klasse A zuweisen würde. Deshalb sollten die Klassen in einem Gleichgewicht aufgeteilt werden (Mishra, 2018).

2.2.12.2 Confusion Matrix

Eine Möglichkeit die Performance eines Modells bei einem Test-Datensatz zu evaluieren ist die Confusion Matrix. Dabei werden die vorhergesagten Werte über die aktuellen Werte aufgetragen.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad \text{Formel 2.2.12.2}$$

- True Positives (TP): Die aktuelle Klasse ist positiv und die vorhergesagte Klasse ist positiv. Korrekte Klassifizierung
- True Negatives (TN): Die aktuelle Klasse ist negativ und die vorhergesagte Klasse ist negativ. Korrekte Klassifizierung
- False Positives (FP): Die aktuelle Klasse ist negativ und die vorhergesagte Klasse ist positiv. Falsche Klassifizierung
- False Negatives (FN): Die aktuelle Klasse ist positiv und die vorhergesagte Klasse ist negativ. Falsche Klassifizierung

	Positiv	Negativ
Positiv	TP	FP
Negativ	FN	TN
Vorhergesagt		
Wahrer Wert		

Abbildung 19: Confusion Matrix (Mishra, 2018)

2.2.12.3 Intersection over Union (IoU)

IoU, auch bekannt als Jaccard Index, ist eine Metrik, um die Übereinstimmung der vorhergesagten Masken mit den selbst gelabelten Masken zu vergleichen. Dabei misst der IoU die übereinstimmenden Pixel der vorhergesagten Maske, mit der selbst gelabelten Maske, dividiert durch die Vereinigung von beiden (Rezatofighi et al., 2019).

$$IoU = \frac{Intersect}{Union} = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{Two overlapping circles}}{\text{Two circles}}$$

Abbildung 20: Intersection over Union (Bergs et al., 2020)

Mithilfe dieser Metrik kann beurteilt werden, wie gut ein Modell Vorhersagen treffen kann. Während bei einem Image Classification Problem die Accuracy Metrik angibt, wie erfolgreich ein Programm trainiert wurde, kann sich bei einem Semantic Segmentation Problem nicht auf diesen Wert verlassen werden. Das liegt daran, dass bei einer Overall Pixel (OP) Accuracy der Wert durch die korrekte Klassifizierung des Hintergrunds zu einem großen Wert verzerrt wird, was vor allem bei Bildern mit einer kleinen Anzahl an Klassen auftritt (Bergs et al., 2020).

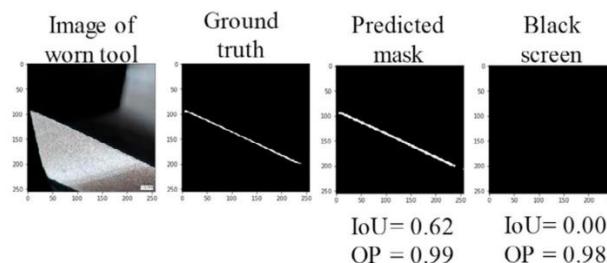


Abbildung 21: Vergleich zwischen IoU und OP Accuracy (Bergs et al., 2020)

Wie in Abbildung 21 zu sehen, ist der Wert der OP Accuracy selbst dann hoch, wenn die Maske komplett eliminiert wurde. Der IoU ist deshalb die bessere Metrik, um ein Semantic Segmentation Problem zu evaluieren, da nur der zu interessierende Bereich im Bild verglichen wird. Dieser ist unabhängig vom Hintergrund, wodurch keine Verfälschung der Werte auftritt.

3 Stand der Technik

3.1 Aktuelle Anwendungsgebiete ML-Bilderkennung

Aufgrund der breiten Anwendungsmöglichkeiten werden immer mehr AI-basierte Bilderkennungsmethoden in unterschiedlichen Forschungsbereichen eingesetzt. Zu diesen Gebieten zählen das Finanzwesen, die Medizin und das autonome Fahren (Wuttke, 2021). Auch im privaten und alltäglichen Gebrauch findet Machine Learning immer mehr Verwendung. Smartphone Hersteller werben immer mehr mit Machine Learning Implementierungen, um ihre Geräte „smarter“ zu gestalten.

3.1.1 Anwendung im Finanzwesen

Bilderkennung kann im Finanzwesen dafür eingesetzt werden, die Sicherheit zu erhöhen. Dazu verwenden Banken eine Gesichtserkennung, um die Identität der Kunden zu verifizieren. Außerdem können Banken eine Gesichtserkennung intern einsetzen, um Zugänge zu kontrollieren. Dies ersetzt dann den klassischen Schlüssel oder die Schlüsselkarte und nur berechtigte Mitarbeiter, deren Gesicht registriert ist, können in bestimmte Bereiche eintreten. Darüber hinaus bieten einige Banken heute schon ihren Kunden an, mittels Gesichtserkennung an Geldautomaten Bargeld abheben zu können (Wuttke, 2021).

Ein Beispiel für eine Anwendung von ML-Bilderkennung liefern Zhu et al. (2019). Das Finanzwesen ist geprägt von vielen unterschiedlichen Dokumenten, die für die Bearbeitung notwendig sind. Zhu et al. (2019) haben mithilfe eines CNNs eine Methode entwickelt, Finanzdokumente zu klassifizieren. Dabei wird versucht, Mehrwertsteuerrechnungen, finanzbezogene Dokumente und weitere finanzbezogene Bilder zu erkennen. Vor allem die große visuelle Variation in den einzelnen Klassen sowie die Ähnlichkeit zwischen den einzelnen Klassen stellen eine große Herausforderung dar. Die vorgeschlagene und erarbeitete Methode von Zhu et al. (2019) hat mehrere Vorteile, um diese Probleme zu kompensieren. Es werden zum einen die Originalbilder erhalten und mithilfe eines Modells vier Teilbilder segmentiert. Diese Methode kann die Ränder der Bilder entfernen, um so ein Overfitting-Problem zu verhindern. Zum anderen wird die Transfer Learning Technik angewendet, die bereits trainierte Modelle als Ausgangspunkt nutzt. Dadurch kann die Genauigkeit der Finanz-Bilder Erkennung deutlich gesteigert werden. Insgesamt ist die Leistung dieser Methode besser als vergleichbare Methoden. Dabei erreicht die vorgeschlagene Methode von Zhu et al. (2019) im Schnitt eine Accuracy von 92,25%. Ein VGG19-Modell erreicht im Vergleich nur 85,18% Accuracy im Schnitt (Zhu et al., 2019).

3.1.2 Anwendung in der Medizin

Im Bereich der Medizin dient die Anwendung von Bilderkennungsalgorithmen vor allem zur Auswertung von computertomographischen Bildern, Mikroskop Aufnahmen und Ultraschall Aufnahmen. Die Anwendung von Artificial Intelligence in der Medizin eröffnet neue Möglichkeiten in verschiedensten Bereichen. So ermöglicht es die roboterassistierte Chirurgie, medizinische Bildverarbeitung und Diagnostik, Überwachung chronischer Krankheiten, klinische Entscheidungsfindung und Krankenhausmanagement durchzuführen (Kleeberger, 2021). Artificial Intelligence findet beispielsweise heute schon Anwendung in der Endoskopie und in der Vorsorgekoloskopie. Damit können frühzeitig Krebserkrankungen entdeckt und behandelt werden (Messmann, 2021). Mithilfe eines Semantic Segmentation Ansatzes unter der Verwendung eines U-Nets kann ein Problem bei der Aufnahme von Röntgenbildern gelöst werden. Dabei verursachen metallische Körperimplantate Fehler und Unklarheiten in einem Röntgenbild. Durch

Semantic Segmentation können diese Quellen erkannt und ausgeglichen werden (Gottschalk et al., 2021). Ein Beispiel, wie ein derartiger Ausgleich aussehen kann ist in Abbildung 22 dargestellt.

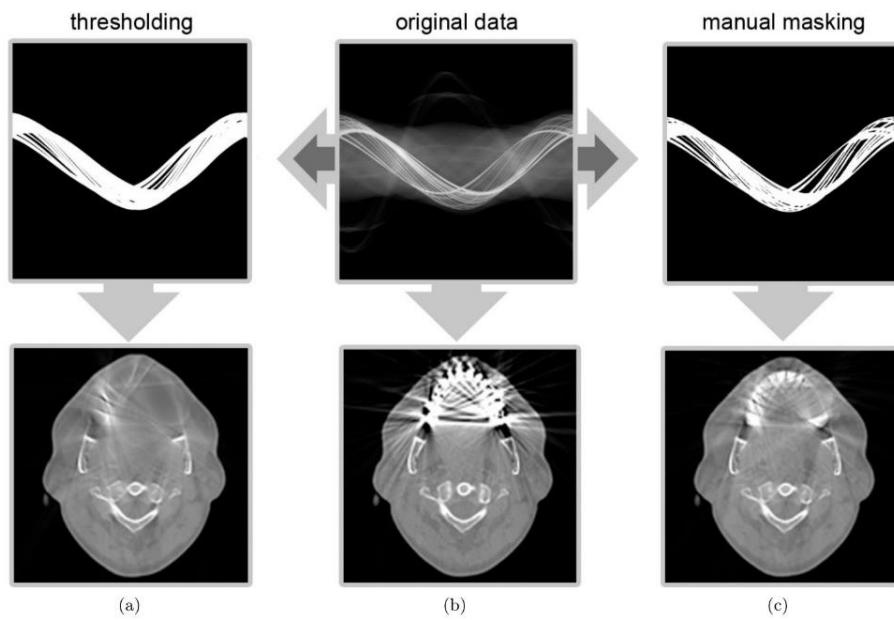


Abbildung 22: Ergebnisse eines CT-Scans eines menschlichen Kiefers mit Amalgamfüllungen.

- (a) mittels Artificial Intelligence erstellte Maske und das daraus rekonstruiertes Bild,
- (b) original aufgenommene Projektionsdaten und das rekonstruierte Bild,
- (c) manuell erstellte Maske und das daraus rekonstruierte Bild (Stille et al., 2013).

Ein Machine Learning Ansatz kann auch bei einer Untersuchung einer Leber mittels Ultraschallbildern angewendet werden. Dabei werden die Stellen in der Leber farblich hervorgehoben, die von einer Verfettung betroffen sind. Aufgrund der meist schlechten und undeutlichen Darstellungen von Ultraschallaufnahmen, ist eine Diagnose abhängig von der Erfahrung der Ärzte und Ärztinnen. Dabei besteht die Gefahr einer Fehldiagnose. Machine-Learning Ansätze können dazu beitragen, die Fehldiagnosequote zu verringern. Dadurch können schwerwiegende Krankheitsverläufe frühzeitig erkannt und behandelt werden (Kuzhipathalil et al., 2021).

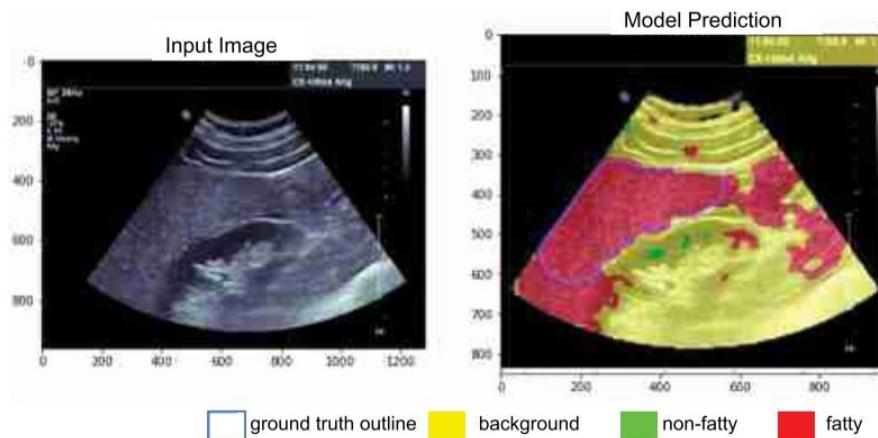


Abbildung 23: Ausgewertetes Bild einer Fettleber mittels Bilderkennung (Kuzhipathalil et al., 2021)

3.1.3 Anwendung beim autonomen Fahren

Die sich rasant weiterentwickelnde Automatisierung findet auch immer mehr Anwendung in der Automobilbranche. Der Trend zu mehr Fahrerassistenzsystemen, wie Spurhalteassistent, Abstandshaltetempomat, automatische Notbremssysteme und weiteren gipfelt im autonomen Fahren. Dabei wird es einem Fahrzeug mithilfe von vorhandenen Assistenzsystemen, basierend auf Sensortechnologie, GPS-Systemen und Kameras ermöglicht, ohne Steuerung durch den Menschen fahren zu können (Johanning, 2020). Durch den Einsatz von Semantic Segmentation können verschiedene Objekte auf der Straße und im Verkehr erkannt und klassifiziert werden. Dieser Einsatz ist essenziell, da Fahrzeuge ein Konzept ihrer Umgebung benötigen, um selbstständig fahren zu können (Stahl et al., 2019).

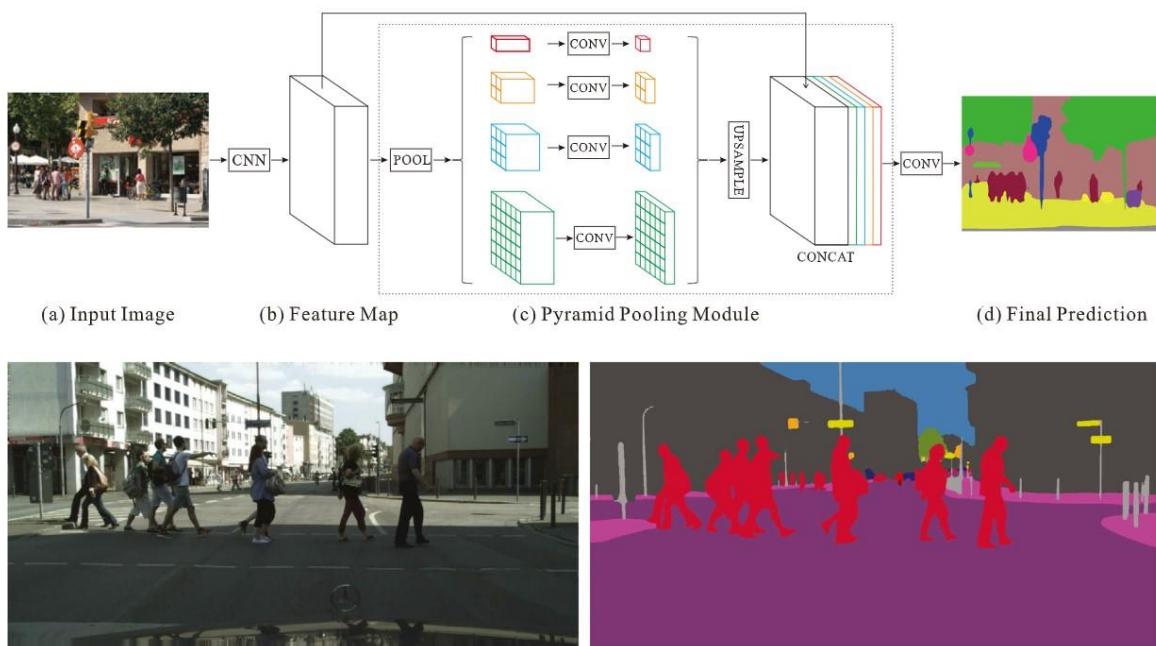


Abbildung 24: Beispiel eines Semantic Segmentation Ergebnisses basierend auf einem PSPNet
(Fujiyoshi et al., 2019)

3.1.4 Anwendung im Alltag

Heute besitzen die meisten Menschen ein Smartphone und tragen dieses bei sich. Durch immer leistungsstärkere Prozessoren wird es ermöglicht, eine ML-Bilderkennung auch im Smartphone anwenden zu können. Die Anzahl an Apps, die es ermöglichen mithilfe von aufgenommenen Fotos Informationen über den Inhalt des Fotos zu bekommen, steigt stetig. So gibt es beispielsweise eine App, die verschiedene Pflanzenarten erkennen kann (Schmidt & Steinecke, 2020). Mithilfe von Google Lens ist es möglich, ähnliche Produkte, wie Kleidung und Haushaltswaren, zu finden und zu ermitteln, wo es diese zu kaufen gibt. Es lassen sich Barcodes scannen, Informationen über ein Buch oder Gemälde in einem Museum herausfinden, verschiedene Pflanzenarten oder Tierarten klassifizieren und spezielle Informationen hierzu erhalten (*Informationen Zu Fotos Und Zur Umgebung Abrufen - Android - Google Fotos-Hilfe*, 2022).

3.2 Einsatz der ML-Bilderkennung im Bereich von Zerspanungswerkzeugen

Da die automatische Verschleißerkennung in Bezug zur Industrie 4.0 einen immer größeren Stellenwert einnimmt, bemühen sich aktuelle Forschungen, Fortschritte in diesem Gebiet zu erreichen.

In der Anwendung von Image Classification Algorithmen haben sich vor allem Convolutional Neural Networks (CNNs) durchgesetzt, da diese effizient bei Bilderkennung eingesetzt werden können. Image Classification Algorithmen lassen sich in den verschiedensten Anwendungsbereichen verwenden, wie zum Beispiel Gesichtserkennung oder Handschrifterkennung. Bergs et al. (2020) haben mithilfe von CNNs eine Werkzeugerkennung entwickelt, die die Label der einzelnen Werkzeuge vorhersagen kann. Sie haben es geschafft, eine Genauigkeit über 95% zu erreichen (Bergs et al., 2020).

Wu et al. (2019) haben mit dem Modell „ToolWearnet“ eine automatische Werkzeugverschleißklassifizierung entwickelt, das mithilfe eines CNNs unterschiedliche Werkzeugverschleißarten erkennt. Es basiert auf einem VGGNet-16 Modell, das häufig in der Bildklassifizierung genutzt wird. Im Vergleich zu einem VGGNet-16 Modell hat dieses eine minimal kleinere Erkennungsgenauigkeit, besitzt aber in Bezug zu Trainingszeit und Erkennungszeit eine bessere Leistung (Wu et al., 2019).

In den letzten Jahren ist, aufgrund der schnellen Entwicklung der Hardware und den damit verbundenen Anstieg der Rechengeschwindigkeit, vor allem Deep Learning in den Fokus der Forschungen getreten. Deep Learning wird deshalb auch bei der Verschleißerkennung verwendet und löst das traditionelle Computer Vision (CV), wie z.B. Color Thresholding, ab. Dies liegt insbesondere daran, dass das traditionelle CV schnell an seine Grenzen stößt. Bei der Betrachtung des traditionellen CV Feature Extraction, fällt auf, dass der Aufwand im Vergleich zu Deep Learning um ein Vielfaches größer ist. Dieser steigt mit steigender Anzahl an Klassen. Während beim Feature Extraction der Ingenieur durch langwieriges Trial-and-Error entscheidet, welche Features am besten die jeweilige Klasse beschreiben, geschieht dies beim Deep Learning automatisch (Mahony et al., 2020). Aufgrund dessen werden immer mehr Deep Learning Ansätze zur direkten Verschleißerkennung eingesetzt.

Bergs et al. (2020) haben in ihrer Studie mithilfe einer U-Net Architektur eine digitale Verschleißerkennung entwickelt und verschiedene Ansätze verglichen. Dabei haben Bergs et al. (2020) festgestellt, dass ein Deep Learning Ansatz zu zuverlässigeren Ergebnissen als beim traditionellen CV führt. Dieser ist zudem robuster gegen veränderte Lichtbedingungen. In Bezug auf den Deep Learning Ansatz wurde festgestellt, dass vor allem schlechte Lichtbedingungen und starke Verschwommenheit der Bilder große Herausforderung bei der Bilderkennung sind. Das unveränderte Originalbild stellt die besten Voraussetzungen zur Verfügung. Des Weiteren bestätigen Bergs et al. (2020), dass Intersection over Union (IoU) eine passende Metrik ist, um die Qualität der Semantic Segmentation Ergebnisse zu überprüfen (Bergs et al., 2020).

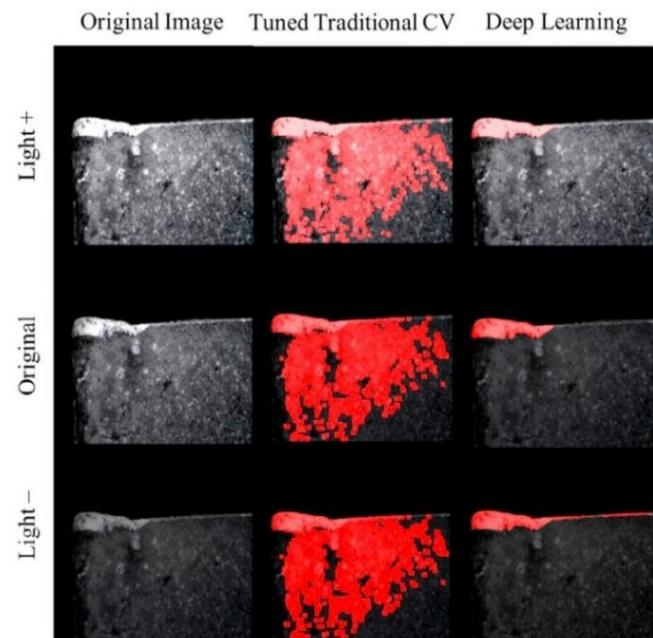


Abbildung 25: Verschleißerkennung mit höherer und niedrigerer Helligkeit. Vergleich zwischen traditionellem CV und einem Deep Learning Ansatz (Bergs et al., 2020)

In einer weiteren Untersuchung fügen Lin et al. (2021) mehrere Bilder zusammen, um bei Schafffräsern ein 360° Bild zu erhalten, das den kompletten Umfang des Fräzers abbildet. Damit kann nicht nur lokal der Verschleiß analysiert werden, sondern an jeder Schneide des Werkzeugs. Um den Verschleiß an den Schneiden zu untersuchen, wurden zusätzlich drei Modelle für Semantic Segmentation verwendet: SegNet, Autoencoder und U-Net. Dabei haben Lin et al. (2021) festgestellt, dass die besten Ergebnisse mit Hilfe des U-Net Modells erreicht werden können (Lin et al., 2021).

Index	Testing image	GT	U-Net	Autoencoder	SegNet
1					
2					
3					
4					

Abbildung 26: Vergleich unterschiedlicher Modelle für die Verschleißerkennung (Lin et al., 2021)

4 Eigener Ansatz

4.1 Image Classification Werkzeugart

4.1.1 Datenbasis erstellen

Die Grundlage eines jeden Image Classification Algorithmus ist eine Datenbasis, mit dem der Algorithmus die wichtigen Features der verschiedenen Bilder erkennen und trainieren kann. Bei der Anwendung eines Image Classification Algorithmus ist es notwendig, dass die Datenbasis dem späteren Anwendungsfall entspricht. Das heißt, Nahaufnahmen einer Wendeschneidplatte können nicht klassifiziert werden, wenn die Datenbasis aus Bildern besteht, die keine Nahaufnahmen sind. Da vor allem in der Industrie Dreh- und Fräswerkzeuge zum Einsatz kommen, werden im folgenden nur Wendeschneidplatten und Schaftfräser klassifiziert. Die Datenbasis besteht aus Nahaufnahmen von Wendeschneidplatten und Schaftfräsern und enthält insgesamt 493 Bilder. Diese bestehen aus 225 Bildern von Schaftfräsern und 270 Bildern von Wendeschneidplatten. Ziel ist es, eine gleich verteilte Anzahl an Bildern von jeder Klasse zu erreichen.

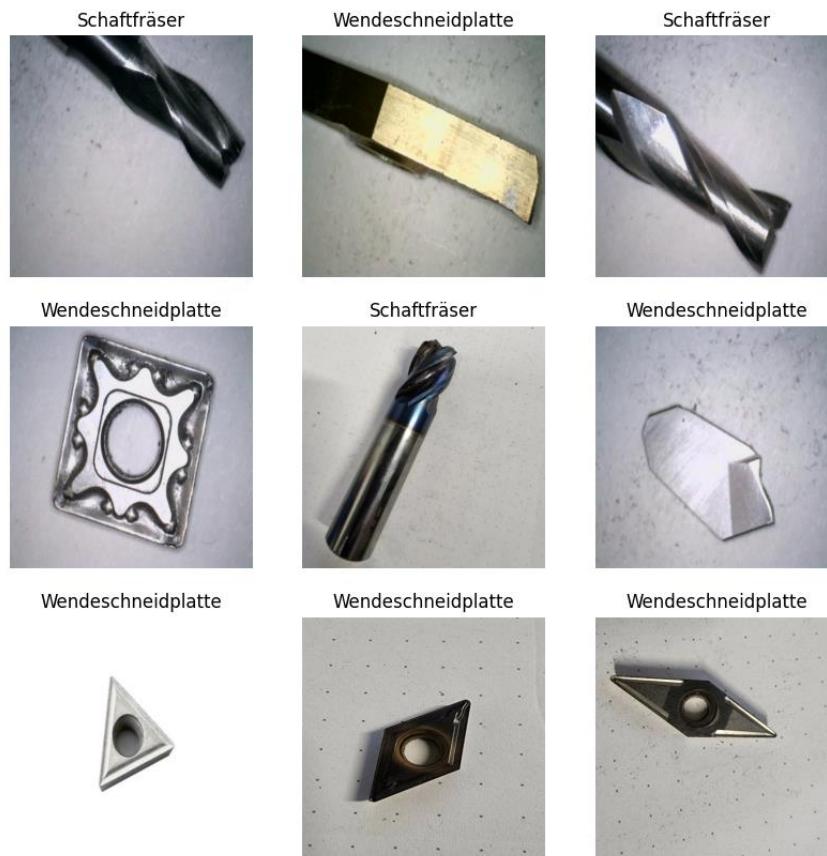


Abbildung 27: Beispiele aus der Datenbasis

Die verwendeten Bilder wurden mit einem Dino-Lite Mikroskop und mit einer Smartphone-Kamera aufgenommen. Außerdem wurden öffentliche Google-Bilder verwendet, um die Vielfalt der unterschiedlichen Wendeschneidplatten und Schaftfräser abdecken zu können.

4.1.2 Python-Programmablauf

Das Programm zur Werkzeugklassifizierung teilt sich in zwei Teile auf. In einem Teil des Programms wird das Modell trainiert und abgespeichert, im anderen Teil wird das Modell geladen und die Werkzeugklassifizierung durchgeführt. Die Grundlage des verwendeten Programm Codes stellt ein Tutorial zur Image Classification auf der offiziellen Tensorflow Internetseite zur Verfügung (TensorFlow, 2018).

4.1.2.1 Training

Zuerst wird die Datenbasis geladen. Diese wird im nächsten Schritt in Trainingsdaten und Validierungsdaten aufgeteilt und auf eine einheitliche Größe reduziert. Danach wird das Modell erstellt, mit dem trainiert werden soll. Dieses besteht zunächst aus drei Conv2D mit jeweils der Activation Function ReLu, drei MaxPooling2D, zwei Dense Layern und einer Flatten Layer. Nachdem das Modell definiert wurde, kann es kompiliert werden. Dabei wird als Optimizer „Adam“ verwendet, wobei dieser eine Art des klassischen Gradient Descent ist, aber mit deutlich besserer Leistung. Da ein Klassifizierungsproblem vorliegt, wird als loss „Sparse Categorical Crossentropy“ verwendet, um die Loss Funktion zu berechnen. Das Präfix „Sparse“ bezieht sich darauf, wie die Datenbasis geladen wird. Werden die einzelnen Klassen als Vektoren ([1, 0], [0, 1]) geladen, dann benötigt es den Zusatz „Sparse“ nicht. Werden diese als Integer ([1], [2]) geladen, wird dieser benötigt. Der Vorteil dabei ist, dass Zeit gespart werden kann, da nur ein Integer verwendet wird. Als Metrik wird „Accuracy“ verwendet, welche berechnet, ob die Bilder richtig klassifiziert worden sind. Je höher der Accuracy-Wert ist, desto besser. Nachdem diese Grundbedingungen definiert wurden, kann das Modell mit der Methode `model.fit()` trainiert werden. In dieser wird die Anzahl der Epochen benötigt, die iterativ bestimmt werden muss, um das beste Ergebnis zu erhalten (vgl. [Kapitel 4.2.2.2](#)). Im Anschluss wird der Verlauf des Modells als Plot dargestellt, um etwaige Probleme, wie Overfitting zu identifizieren und vorzubeugen. Wenn der Verlauf des Modells den erwarteten Ergebnissen entspricht, wird dieses gespeichert, um das trainierte Modell anwenden zu können.

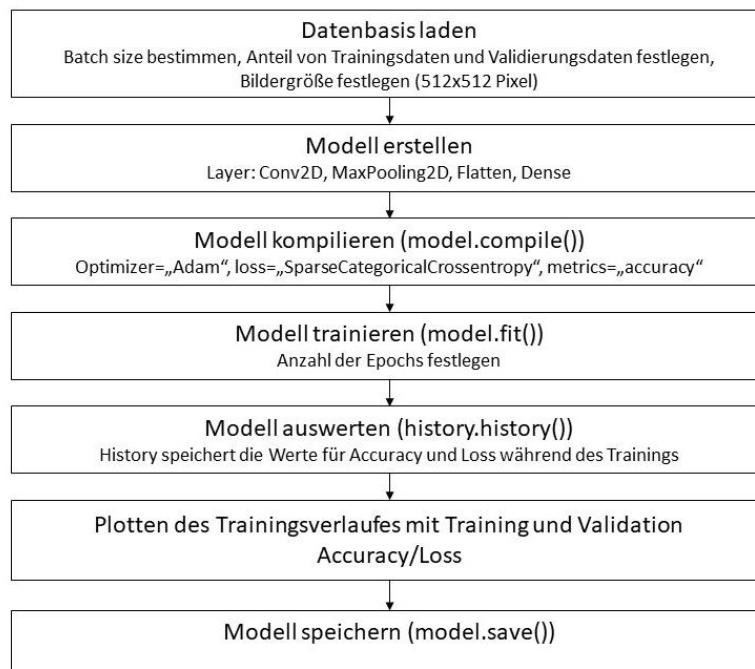


Abbildung 28: Ablauf des Trainingsprogramms

Nach der Ausführung des Programms, fällt auf, dass der Plot Overfitting aufzeigt.

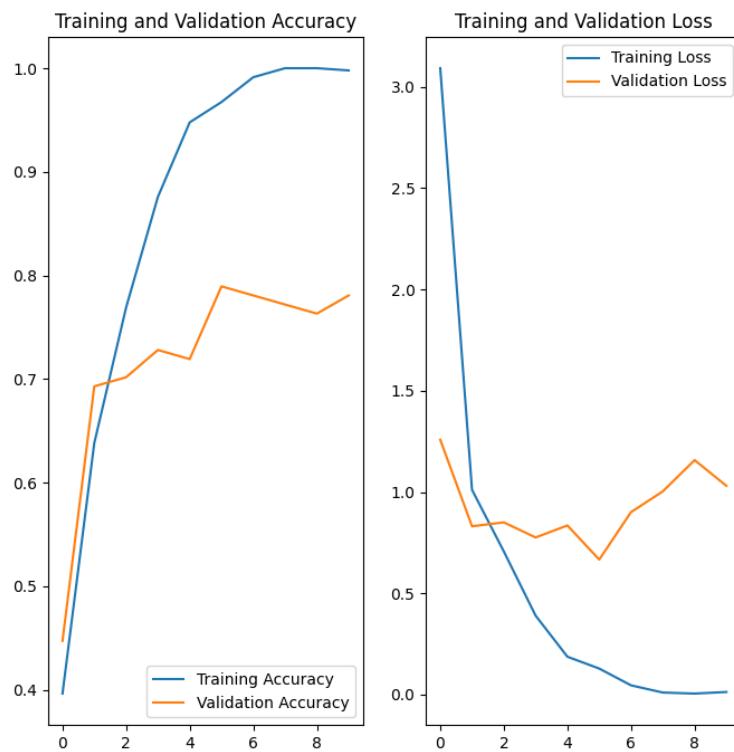


Abbildung 29: Trainingsverlauf mit Overfitting

Das Programm wurde nur zehn Epochen ausgeführt, da mit steigender Anzahl Overfitting begünstigt wird. Um Overfitting entgegenzuwirken, wird Data Augmentation verwendet. Dabei werden die Bilder mithilfe eines zusätzlichen Layers im Modell gespiegelt, um zehn Prozent gedreht und um zehn Prozent vergrößert.

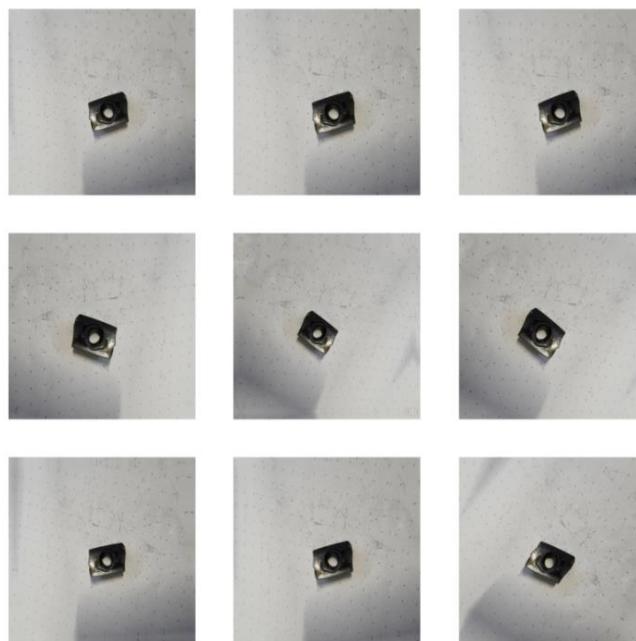


Abbildung 30: Beispiel Data Augmentation mit einer Wendeschneidplatte

Zusätzlich wird das Dropout-Layer in das Modell hinzugefügt, das dafür sorgt, dass während des Trainingsvorgangs mehrere Outputs aus dem Layer zufällig ausgelassen werden. Wird beispielsweise Dropout(0.2) verwendet, bedeutet dies, dass zwanzig Prozent der Outputs aus dem Layer fallen. Zusätzlich kann die Anzahl der Epochen erhöht werden, um das Ergebnis weiterhin zu verbessern. Das Resultat ist eine deutliche Verbesserung des Trainingsergebnisses und kann dementsprechend für die Anwendung zur Bildklassifizierung verwendet werden.

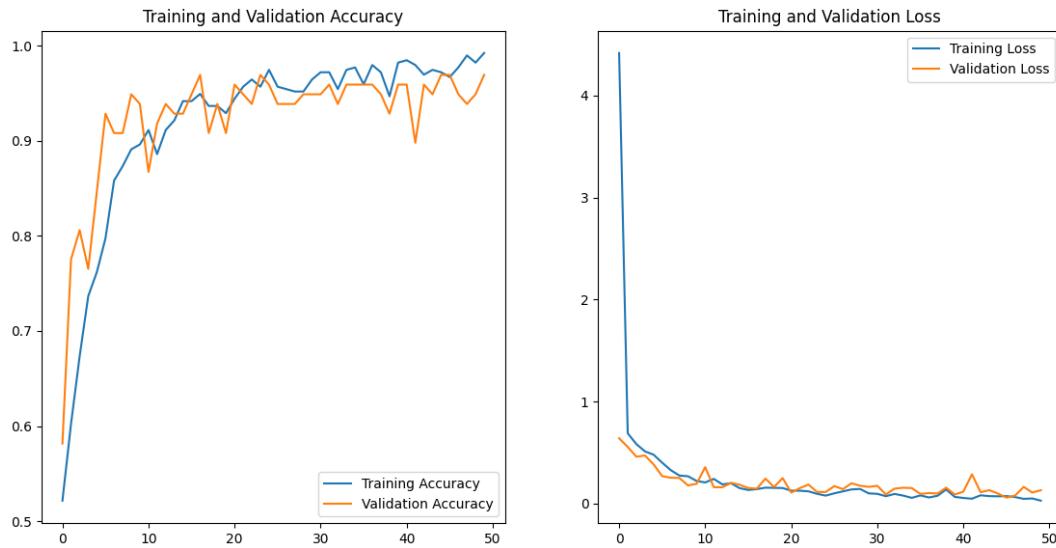


Abbildung 31: Trainingsverlauf ohne Overfitting mit 50 Epochen

4.1.2.2 Anwendung

Die Klassifizierung wird durchgeführt, indem das aufgenommene Bild geladen oder durch eine Kameraimplementierung direkt aufgenommen und auf einheitliche Größe reduziert wird. Anschließend wird das Modell geladen und das Bild mittels `model.predict()` Methode ausgewertet. Als Ergebnis erscheint das aufgenommene Bild mit der Klassenzugehörigkeit. Zusätzlich wird eine Prozentzahl angegeben, die angibt, wie groß die Sicherheit der Vorhersage ist.

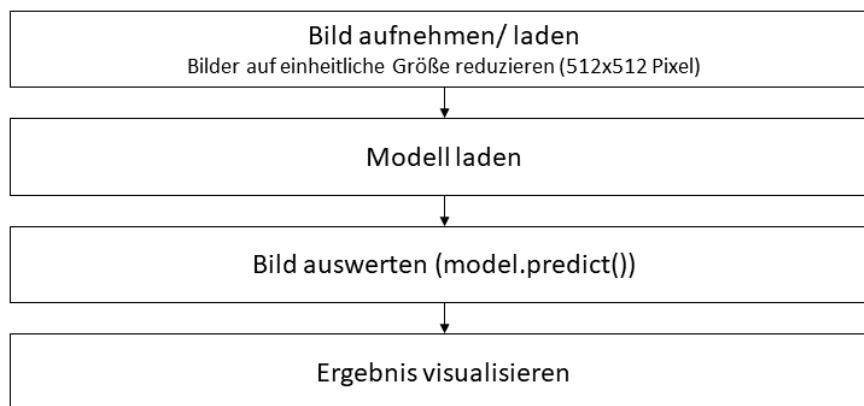


Abbildung 32: Ablauf des Auswertungsprogramms

This image most likely belongs to Schaftfräser with a 100.00 percent confidence. This image most likely belongs to Wendeschneidplatte with a 99.99 percent confidence.

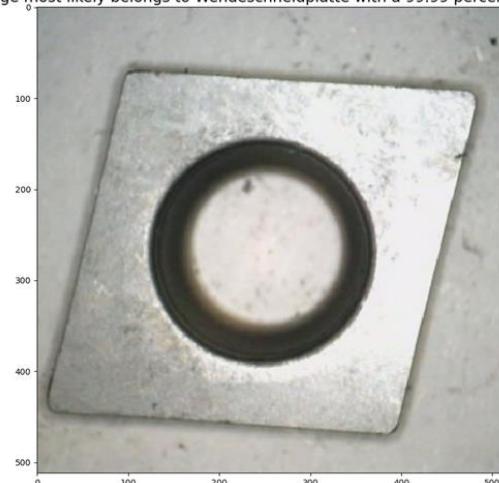


Abbildung 33: Klassifizierung eines Schaftfräisers und einer Wendeschneidplatte

4.2 Identifizierung von abrasivem Werkzeugverschleiß

Die quantitative Verschleißerkennung kann mithilfe von Semantic Segmentation Modellen durchgeführt werden. Sie sind in der Lage einzelne Bildbereiche zu erkennen und zu klassifizieren. Da während der Bearbeitung mit Zerspanungswerkzeugen hauptsächlich Freiflächenverschleiß stattfindet, ist diese Art des Verschleißes der Hauptindikator wann ein Werkzeug gewechselt werden sollte. Im Folgenden konzentriert sich die Verschleißerkennung nur auf diese Art des Verschleißes.

4.2.1 Datenbasis erstellen

Damit ein Semantic Segmentation Modell zuverlässige Vorhersagen treffen kann, bedarf es einer großen und eindeutigen Datenbasis, mit dem das Programm trainiert werden kann. Dafür werden zu jedem aufgenommenem Bild die „Ground Truth“ benötigt. Das sind manuell erstellte Masken eines Bildes, die das zu erkennende Objekt in einem Bild markieren. Die Ground Truth markiert die Größe und Position des Verschleißes an einer Wendeschneidplatte.

4.2.1.1 Bilder aufnehmen

Die Aufnahme der Bilder der verschlissenen Wendeschneidplatten wird mit einem Dino-Lite Digital-Mikroskop mit einer Vergrößerungsrate von 10 – 140x aufgenommen. Es kann Bilder mit einer Auflösung von maximal 2592x1944 Pixeln aufnehmen. Zusätzlich ist es mit vier LED-Lichtern ausgestattet, die flexibel ein- und ausgeschaltet und jeweils in ihrer Helligkeit gedimmt werden können.

Da für die Datenbasis nur verschlissene Wendeschneidplatten notwendig sind, können diese Bilder mithilfe einer Vorrichtung abseits einer CNC-Maschine aufgenommen werden. Wie in Abbildung 34 zu sehen, wurde als Anschlag eine 3D-gedruckte Haltevorrichtung verwendet, die eine Schräge von 7° besitzt, damit die verschlissene Fläche des Werkzeugs parallel zum Mikroskop ausgerichtet ist. Das Mikroskop selbst wurde mit einem Ständer in Position gehalten. Bei der Aufnahme der Bilder wurden die vier LED-Lichter den jeweiligen Bedingungen angepasst, sodass der Verschleiß klar zu erkennen war.

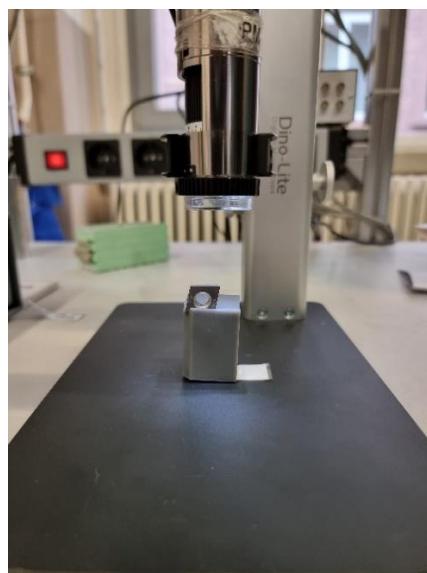


Abbildung 34: Vorrichtung zur Aufnahme der Verschleißbilder

Der Verschleiß der Wendeschneidplatten wurde durch Außenlängsdrehen des Stahls 42CrMo4 hervorgerufen. Bei der Bearbeitung wurde kein Kühlsmierstoff verwendet und mit konservativen Prozessparametern gearbeitet, wodurch nur Freiflächenverschleiß verursacht wurde.

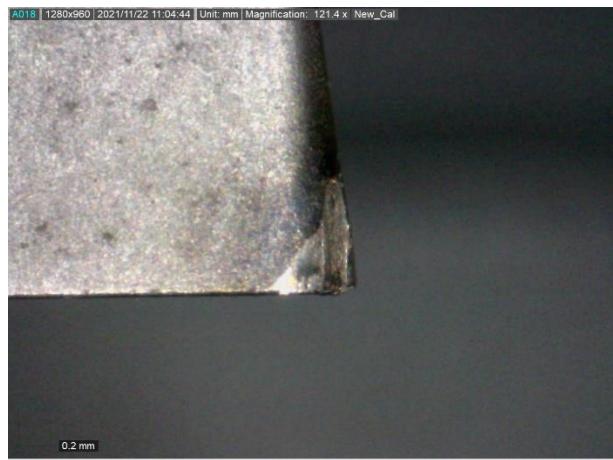


Abbildung 35: Beispiel eines aufgenommenen Bildes mit dem Dino-Lite Digital-Mikroskop

4.2.1.2 Erstellen der Ground Truth Masken

Für eine korrekte Auswertung werden gelabelte Bilder benötigt. Es gibt zahlreiche Programme, um diese Labels zu erstellen. Sie sind in der Regel frei zugänglich. Vor dem Labeln werden die Bilder auf eine einheitliche Größe reduziert und zugeschnitten. Die Größe der Bilder beträgt danach 512x512 Pixel.

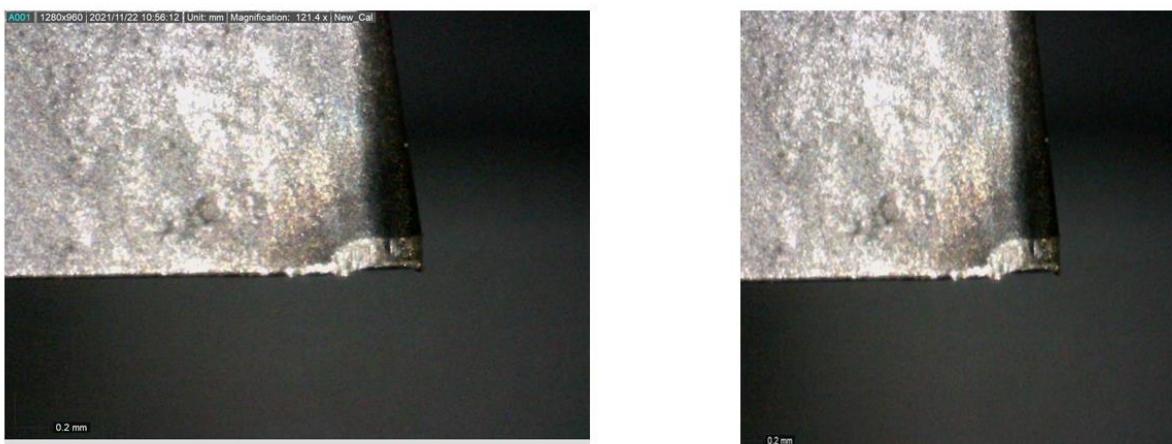


Abbildung 36: Aufgenommenes Originalbild mit einer Auflösung von 1280x960 Pixeln (links) und komprimiertes, zugeschnittenes Bild mit einer Auflösung von 512x512 Pixeln (rechts)

Zur Label Erstellung wird die Internetseite apeer.com verwendet (Apeer.Com). Der Annotationsprozess läuft so ab, dass der Datensatz mit den zu verwendenden Bildern hochgeladen wird. Im nächsten Schritt können die Bilder dann gelabelt und zum Schluss als „Binary Mask“ exportiert werden. Die exportierten Bilder müssen jedoch noch einmal nachbearbeitet werden, da diese standardmäßig einen Farbraum von 0 bis 255 besitzen. Da die Masken nur Werte von 0 und 1 besitzen, muss der Farbraum von 0 bis 1 eingestellt werden, da ansonsten nur schwarze Bilder zu sehen sind. Der Wertebereich der Bilder kann mithilfe des Programms „Fiji“ von GNU General Public License eingestellt werden (Schindelin et al., 2012). Ein

Beispiel des Labeling Prozesses ist in Abbildung 37 dargestellt. Das Resultat inklusive Nachbearbeitung mit der Fiji-Software ist in Abbildung 38 dargestellt.

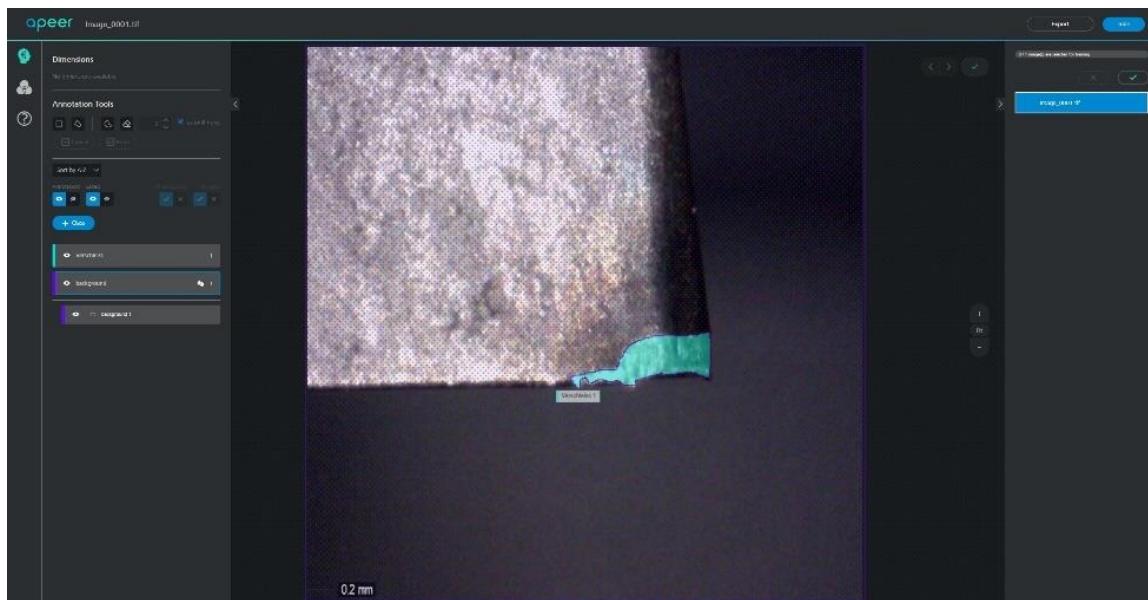


Abbildung 37: Apeer.com Bearbeitungsoberfläche

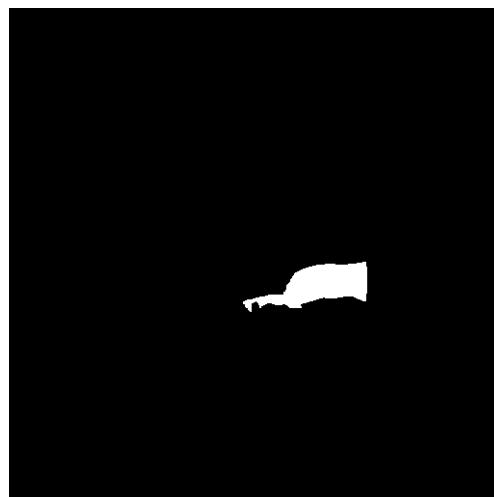


Abbildung 38: Maske eines Verschleißbildes

4.2.1.3 Data Augmentation

Semantic Segmentation Modelle benötigen große Datenbasen. Es ist mit einem großen Aufwand verbunden, einzelne Bilder zu labeln und zu bearbeiten. Bei mehr als 2000 Bildern würde dies zu viel Zeit in Anspruch nehmen. Deshalb ist es üblich eine kleinere vorhandene Datenbasis durch verschiedene Operationen zu erweitern. Diese Operationen können sein: Drehen, Schärfen, Spiegeln, Unschärfen, usw. Mithilfe der Python Bibliothek „Pillow“ ist es möglich, Bilder zu drehen und verschiedene Filter anzuwenden.

Augmentation Technik	Einstellungen
Rotation	+10°, +180°, -10°
Zoom in	80%
Gaussian Blur	Standard
Kontrasterhöhung	40%
Helligkeitserhöhung	20%

Tabelle 1: Verwendete Data Augmentation Operationen

Es ist maßgebend, dass alle Augmentation Operationen mögliche Variationen der aufgenommenen Bilder im Anwendungsfall widerspiegeln. Eine höhere Belichtung oder Verdunkelung ist nicht notwendig, wenn dies im Anwendungsfall nicht vorkommt. Unübliche Einstellungen sind in der Regel für das Training der Modelle herausfordernd und können das Ergebnis möglicherweise verschlechtern. Dies gilt auch, wenn eine zu große Data Augmentation betrieben wird. Ab einem gewissen Punkt verschlechtert es die Leistung eines Modells, da das Programm beginnt, von Bildrauschen oder unwichtigen Bildinformationen zu lernen. Hinzu kommt, dass mit steigender Größe einer Datenbasis, die Dauer des Trainierens proportional ansteigt. Mit einer kleineren Datenbasis können so Ressourcen gespart werden, jedoch mit dem Risiko ein schlechteres Ergebnis zu erzielen. Es ist demnach notwendig, die richtige Größe der benötigten Datenbasis zu finden, um das beste Ergebnis bei geringem Ressourcenverbrauch zu erreichen.

Die verwendete Datenbasis besitzt 314 Originalbilder mit entsprechend 314 Originalmasken. Die Auflösung beträgt 512x512 Pixel. Die Verwendung höheren Auflösung als 512x512 Pixel führt zu keinem signifikant genaueren Ergebnis, daher ist diese Auflösung aufgrund des geringeren Ressourcenaufwandes angemessen. Nach der Data Augmentation besteht die Datenbasis aus 2512 Bildern und gleich vielen Masken.

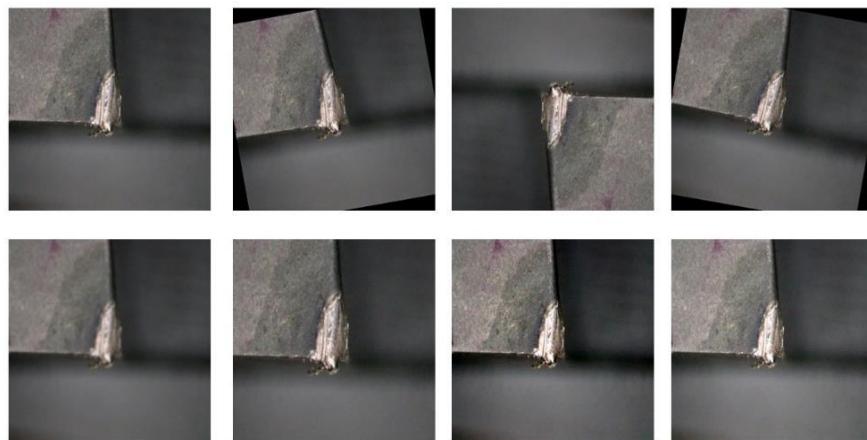


Abbildung 39: Originalbild (links oben) und sieben erweiterte Bilder

Die Ergebnisse eines Semantic Segmentation Programms mit Data Augmentation verglichen mit einem Programm ohne Data Augmentation zeigen, dass ohne Data Augmentation das Programm Schwierigkeiten hat die Kontur des Verschleißes akkurat wiederzugeben. Beide Programme sind mit gleichen Parametern trainiert worden. Auch der IoU-Score spiegelt die bessere Performance des Modells mit Data Augmentation wider. Eine Zunahme um knapp 18% ist eine deutliche Erhöhung. Der Wert entspricht etwa dem IoU-Score Durchschnitt der getesteten Bilder. Dabei hat

das Programm ohne Data Augmentation im Schnitt einen IoU-Score von 0,623 erreicht und das Programm mit Data Augmentation einen IoU-Score von 0,804.

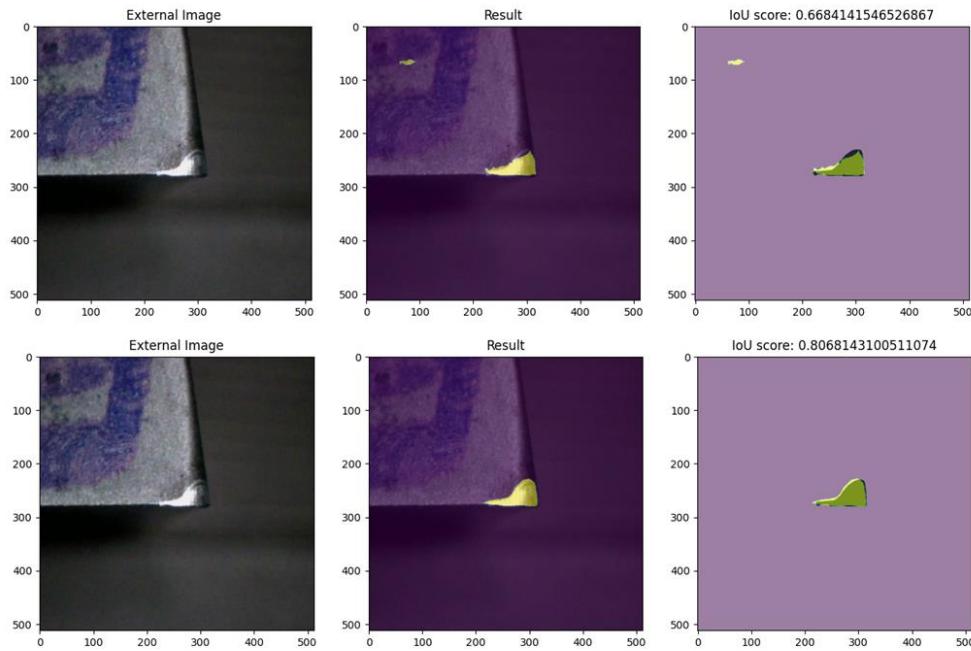


Abbildung 40: Ergebnis ohne Data Augmentation (oben) im Vergleich mit Data Augmentation (unten)

4.2.2 Python-Programmablauf

4.2.2.1 Training

Wie Lin et al. (2021) in ihrer Forschung herausgefunden haben, sind bei der Verwendung eines U-Net Modells die besten Ergebnisse zu erwarten. Deshalb wird zur Durchführung der Verschleißerkennung dieses U-Net verwendet. Für die quantitative Identifizierung von Werkzeugverschleiß wurde als Basis der Programm Code (Bhattiprolu, 2021b) und das U-Net Modell (Bhattiprolu, 2021a) von Dr. Sreenivas Bhattiprolu verwendet, der seine Codes öffentlich auf GitHub zur Verfügung stellt.

In Abbildung 41 ist der Ablauf des Python Programms dargestellt. Zu Beginn eines Trainingsprogramms muss die zuvor erstellte Datenbasis geladen werden. Damit Fehler in der Bildergröße, die möglicherweise beim Erstellen der Datenbasis aufgetreten sind, vermieden werden, werden die Bilder nochmals auf die richtige Größe reduziert. Dabei ist es essentiell, dass immer die passenden Masken zu den Verschleißbildern geladen werden, da das Programm sonst von falschen Informationen lernt. Nachdem das Programm die notwendige Datenbasis geladen hat, muss diese aufgeteilt werden. Hierbei werden jeweils von den Masken und Bildern 10% entfernt. Diese werden zur Überprüfung mithilfe von Backpropagation verwendet. Es werden folglich 90% der Daten zum Training verwendet und 10% zur Validierung. Nachdem dieser Schritt abgeschlossen ist, kann das Modell geladen und kompiliert werden. Als Optimizer wird „Adam“ verwendet. Da bei der Bildauswertung nur entschieden werden muss, ob der betrachtete Pixel zur Verschleißfläche gehört oder nicht, kann als Loss „binary_crossentropy“ verwendet werden, um die Loss Function zu berechnen. Die Metrik „Accuracy“ gibt an, wie viele Pixel in einem Bild richtig klassifiziert worden sind. Da der Großteil der Bilder aus „Hintergrund“ besteht, ist dieser Wert

üblicherweise bei Semantic Segmentation Anwendungen überdurchschnittlich hoch. Mit der Methode `model.fit()` wird das Modell mit bestimmten Parametern trainiert. Die Variable `batch_size` gibt an, wie viele Bilder auf einmal durch das Neuronale Netz laufen. Je größer die `batch_size` ist, desto schneller kann das Programm ausgeführt werden. Mit steigender Größe steigt aber die benötigte Leistung des Computers. Die `batch_size` ist demnach durch die Leistung des verwendeten Computers begrenzt. `Verbose` gibt an, wie die Darstellung der Iterationen während des Programmdurchlaufs sein soll. Dabei gibt es drei Möglichkeiten. 1 steht dafür, dass der Fortschritt mithilfe eines Balkens visualisiert wird. Bei 2 und 3 werden weniger Informationen dargestellt und der Verlauf des Trainings kann nicht genau beobachtet werden. Deshalb wird standardmäßig `verbose=1` verwendet. Die Anzahl an Epochen kann je nach Datensatz und Anwendungsfall variieren. Genauere Erläuterungen sind in [Kapitel 4.2.2.2](#) zu finden. `Shuffle` steht dafür, ob die Trainingsdaten vor jeder Epoche gemischt werden sollen. Dies ist jedoch nicht erwünscht, da die Masken mit den Verschleißbildern zusammenpassen müssen. Nach dem Training kann das trainierte Modell gespeichert werden, damit dieses für die Bildauswertung verwendet werden kann. Das Modell wird mit der Methode `model.evaluate()` ausgewertet. Diese gibt den Loss-Wert und den Wert der Accuracy wieder. Zum Schluss wird der Trainingsverlauf geplottet, um zu sehen, ob das Programm Overfitting aufweist oder nicht. Der IoU-Score beschreibt, wie erfolgreich das Programm trainiert worden ist.

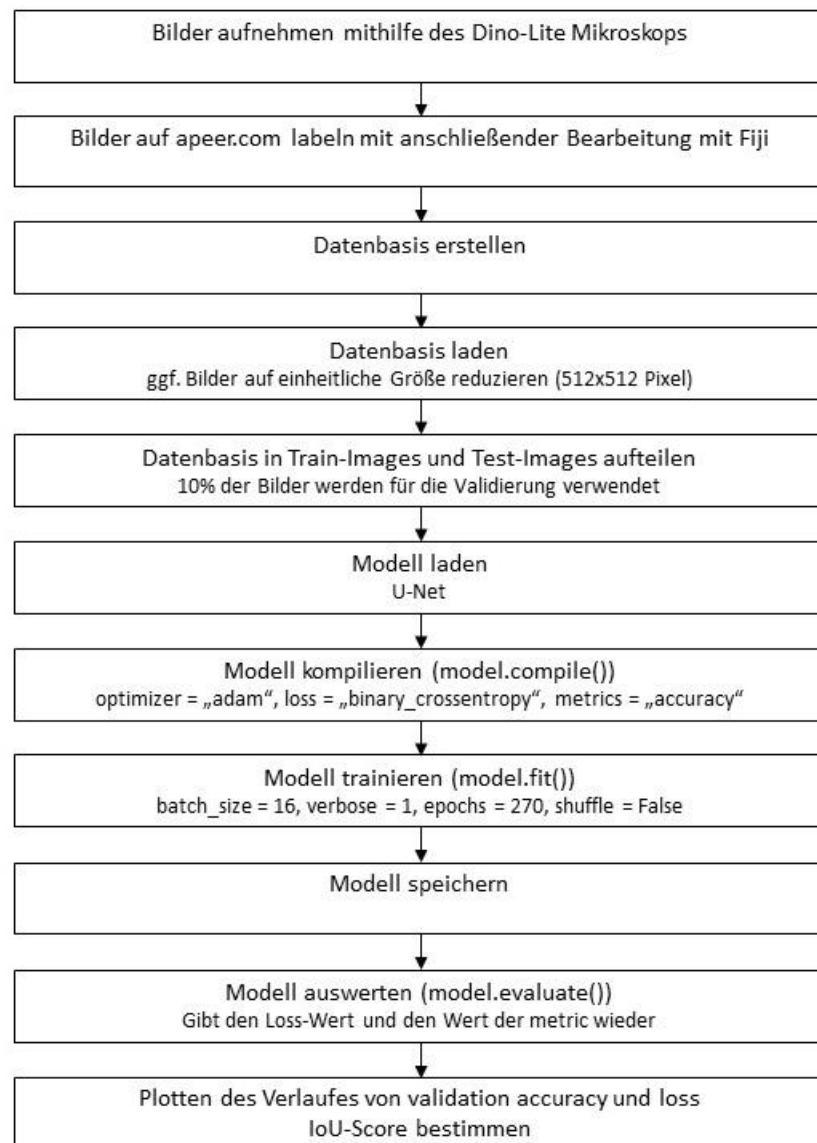


Abbildung 41: Ablauf der Erstellung der Datenbasis und des Trainingsprogramms

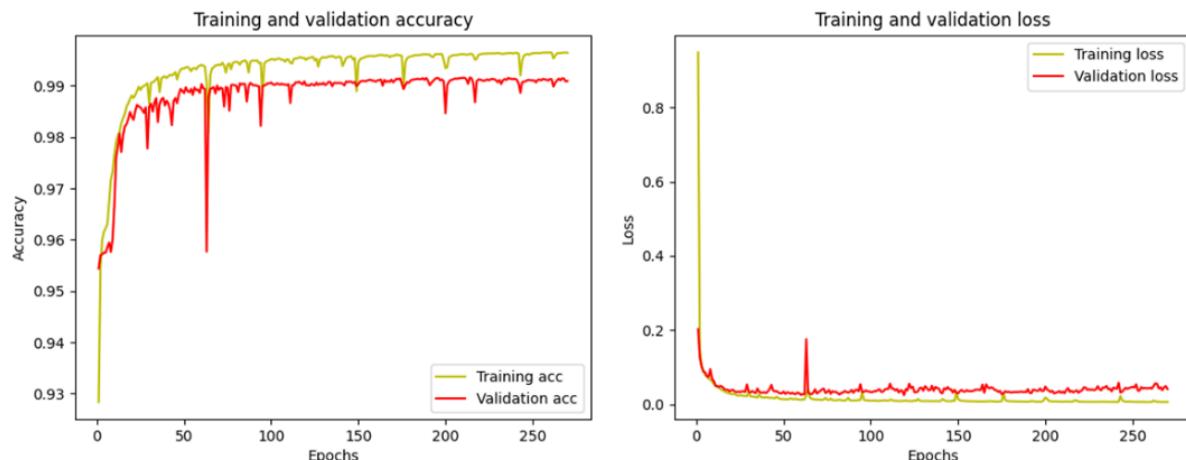


Abbildung 42: Trainingsverlauf mit 270 Epochen

4.2.2.2 Anwendung

Damit bei der Auswertung eines Verschleißbildes das Programm nicht neu trainiert werden muss, kann das trainierte Modell gespeichert und erneut geladen werden. Die Methode `model.load()` lädt das gespeicherte Modell. Anschließend wird das Bild benötigt, bei dem der Verschleiß erkannt werden soll. Dabei kann ein Bild verwendet werden, das mithilfe einer anderen Software aufgenommen wurde oder mittels Kameraimplementierung im Programm. Das verwendete Bild muss auf die Größe der Bilder reduziert werden, mit denen das Modell trainiert wurde. Mit der Methode `model.predict()` werden die „Predictions“ erstellt und können mithilfe der Bibliothek Matplotlib dargestellt werden. Im Anschluss können diese Informationen verwendet werden, um den Verschleiß zu messen.

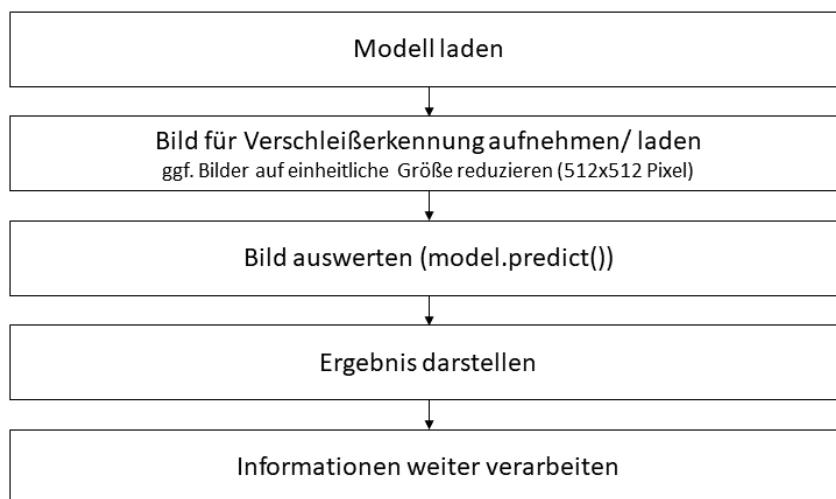


Abbildung 43: Ablauf des Auswertungsprogramms

4.2.2.3 Anzahl der Epochen

Beim Trainieren eines Programms verwendet das Programm den vorgegebenen Datensatz, der einmal durch das Neural Network geschickt wird und anschließend mittels Backpropagation wieder zurück. Da die meisten Computer in ihrer Leistung sehr begrenzt sind, können diese Datensätze in einzelne „Batches“ aufgeteilt werden, damit das Training trotzdem durchgeführt werden kann. Besitzt eine Datenbasis beispielsweise 200 Bilder, dann würde das Programm mit vorgegebener Batch-Size von 20 das Netz zehnmal durchlaufen. Dies entspricht dann einer Epoche. Es wäre auch möglich, den kompletten Datensatz auf einmal durch das Netz laufen zu lassen, jedoch wären dafür enorme Kapazitäten notwendig. Nach einem Durchlauf besitzt das Programm noch nicht die Fähigkeit verwendbare Ergebnisse zu liefern. Deshalb wird eine größere Anzahl an Epochen verwendet, die je nach Datensatz und Anwendung unterschiedlich sein kann. Ist die Aufgabe beispielsweise die Erkennung eines schwarzen Autos vor einem weißen Hintergrund, bedarf es weniger Epochen als bei komplizierteren Aufgaben. Die Anzahl der Epochen kann beliebig variiert werden, um das Optimum herauszufinden. Das Programm wurde mit 30, 90, 130, 180, 230 und 270 Epochen trainiert und deren Ergebnisse in Bezug auf IoU-Score und Bildauswertung verglichen.

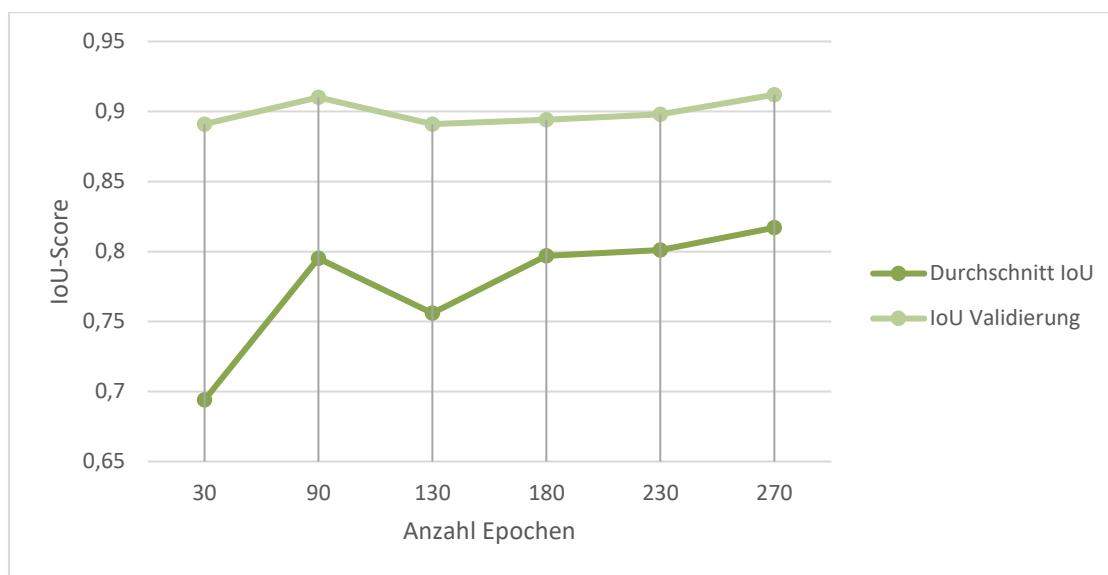


Abbildung 44: IoU-Score in Abhängigkeit von der Anzahl der Epochen

Bei der Betrachtung der Ergebnisse fällt auf, dass mit steigender Anzahl an Epochen der Detaillierungsgrad steigt. Es werden bereits ab 30 Epochen gute Ergebnisse erreicht. Damit können aber bei Bildern mit ungünstigen Voraussetzungen, wie schlechten Lichtverhältnissen keine konsistenten Ergebnisse erzielt werden. Es möglich, auch bei einer höheren Anzahl an Epochen schwächere Ergebnisse zu erhalten. Dies ist abhängig vom Eingabebild. So kann beispielsweise das Ergebnis bei 180 Epochen bei einem Bild perfekte Ergebnisse hervorbringen und bei 270 Epochen fehlerhafte. Dies könnte auch bei einem anderen Bild wieder andersherum sein. Letztendlich überwiegt die Tendenz mit höherer Anzahl an Epochen, bessere Ergebnisse zu erzielen.

Wie in Abbildung 45 zu sehen, hat das Ergebnis mit 270 Epochen den höchsten Detaillierungsgrad und folgt der Struktur des Verschleißes am besten. Dies ist vor allem nützlich bei Verschleißbildern, die eine unregelmäßige Struktur besitzen.

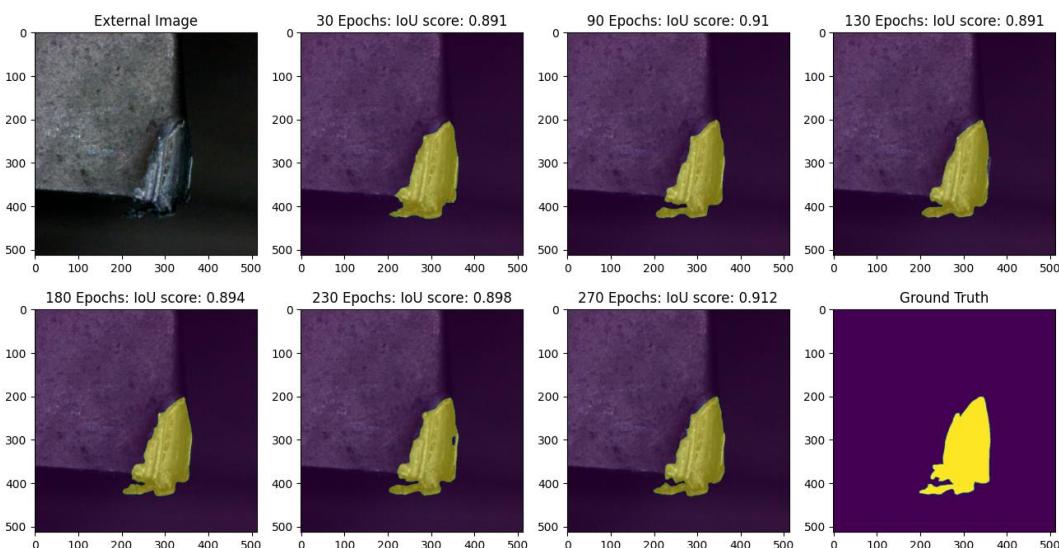


Abbildung 45: Vergleich der Ergebnisse unterschiedlicher Anzahl an Epochen

5 Ergebnisse

Die Resultate der Verschleißerkennung haben eine große Übereinstimmung mit der Fläche des Verschleißes. Um diese Ergebnisse in einer industriellen Umgebung anwenden zu können, wird eine Recheneinheit in einer Werkzeugmaschine benötigt, die die Bildinformationen verarbeiten und auswerten kann, um so die Möglichkeit zu schaffen, diese Methoden während eines Bearbeitungsprozesses zu verwenden.

5.1 Implementierung der Bilderkennung mittels Raspberry Pi in einer Werkzeugmaschine

Für die Anwendung der Bilderkennungsmethoden in einer Werkzeugmaschine unabhängig von einem externen Computer wird eine eigene Recheneinheit benötigt. Dazu bietet sich ein Raspberry Pi an, der aufgrund seiner kompakten Größe und vielfältigen Einsatzmöglichkeiten geeignet ist.

Damit der Raspberry Pi auf die Ressourcen der Programme zugreifen kann, muss Tensorflow auf der Python-IDE (integrated development environment) installiert werden. Als IDE wird Thonny verwendet. Die Implementierung der Programme in den Raspberry Pi wurde mit der Unterstützung der wissenschaftlichen Hilfskraft Jin Yun meines Betreuers durchgeführt.

Die Bilderkennung wird mit einem Mikroskop durchgeführt, das die Verschleißfläche aufnimmt. Das Mikroskop kann entweder direkt in der Maschine eingesetzt werden oder abseits, indem die Wendeschneidplatten jeweils abgeschraubt und separat ausgewertet werden. Eine Implementierung in eine Werkzeugmaschine ist jedoch vorzuziehen, außer dies ist wegen Bauraumbeschränkungen oder anderen Restriktionen nicht möglich. Als Mikroskop für die Implementierung wurde dasselbe verwendet, das auch zur Erstellung der Datenbasis genutzt wurde.

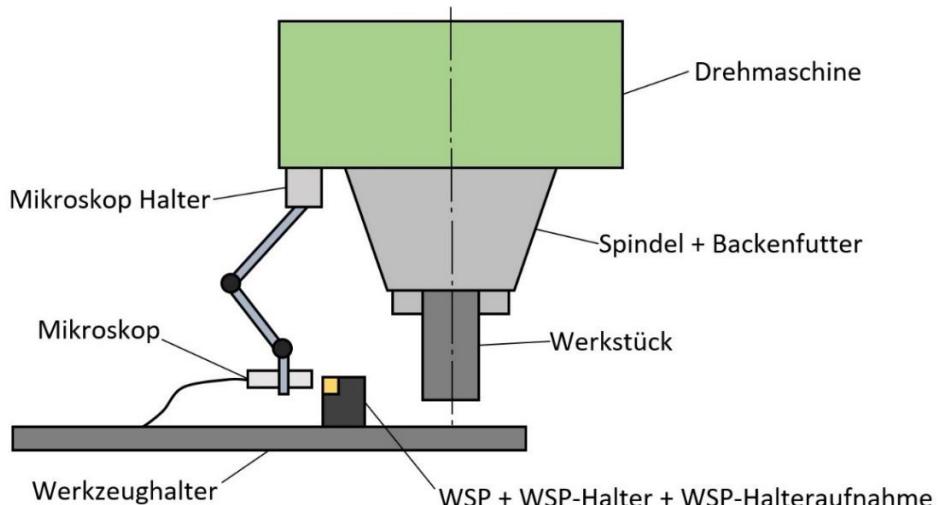


Abbildung 46: Aufbau einer Implementierung in eine Werkzeugmaschine am Beispiel einer Index V100

Bei der Implementierung des Mikroskops in die Werkzeugmaschine muss dieses richtig positioniert werden. Dabei werden je nach Programm unterschiedliche Abstände vom Werkzeug benötigt. Bei einer Werkzeugklassifizierung ist es notwendig, einen großen Teil des Werkzeugs

auf der Bildaufnahme zu sehen. Im Gegensatz dazu ist es bei der Verschleißquantifizierung notwendig, die Verschleißzone vollständig und so groß wie möglich aufzunehmen.

Der Prozessablauf ist in zwei Teile gegliedert. Als Erstes wird das Werkzeug klassifiziert. Mit dieser Information werden je nach Werkzeug unterschiedliche Verschleißerkennungsalgorithmen verwendet. Da sich die quantitative Verschleißerkennung nur auf Wendeschneidplatten bezieht, wird im zweiten Teil der Semantic Segmentation Ansatz zur Verschleißidentifizierung von Wendeschneidplatten verwendet.

5.1.1 Werkzeugklassifizierung

Der Ablauf der Werkzeugklassifizierung ist ähnlich zum Standardverlauf (vgl. [Kapitel 4.1.2](#)) der Werkzeugklassifizierung außerhalb der Maschine. Das liegt daran, dass nur die Form und Farbe des Werkzeugs ausgewertet wird und die Maße keine Rolle spielen.

Für den Zugriff auf das Mikroskop im Python Programm wird die OpenCV-Methode cv2.VideoCapture() verwendet. Mit dieser Methode ist es nicht möglich, die am Mikroskop befindlichen LEDs einzustellen. Damit geht eine bedeutsame Funktion verloren, um die nötigen Grundlagen für ein klares Bild zu schaffen. Bei einer Werkzeugklassifizierung ist dieses Problem jedoch nicht relevant, da vorzugsweise die Form des Werkzeugs ausgewertet wird. Zusätzlich muss die Auflösung des aufgenommenen Bildes den verwendeten Trainingsbildern entsprechen. Dazu wird die Größe des Aufnahmefensters definiert. Das Mikroskop kann jedoch keine quadratischen Bilder mit der Größe 512x512 aufnehmen. Da dies aber notwendig ist, damit eine Auswertung durchgeführt werden kann, muss das Bild beschnitten werden und auf die richtige Pixelgröße skaliert werden.

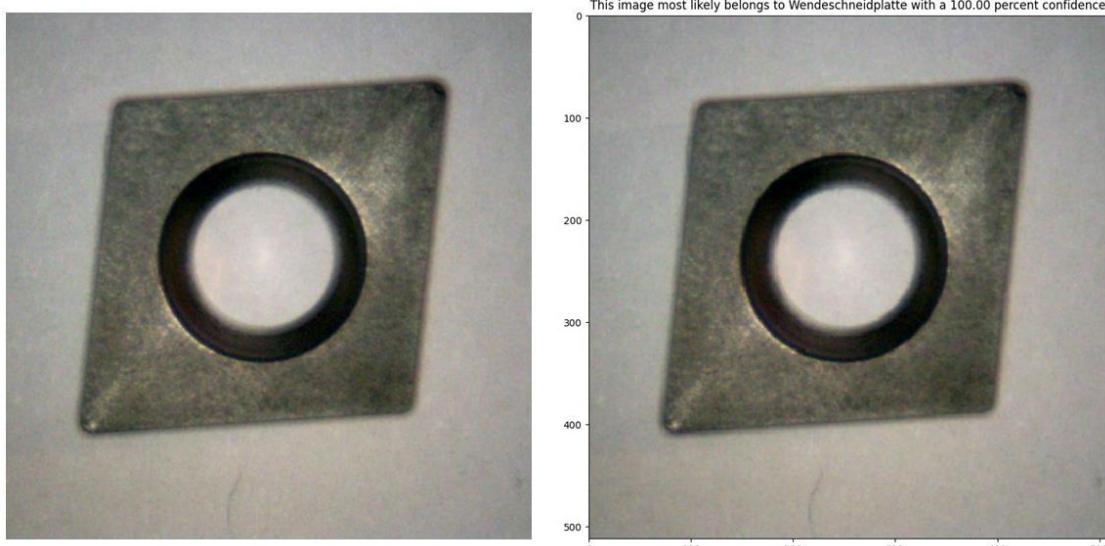


Abbildung 47: Aufnahme mit dem Mikroskop (links), Ergebnis der Auswertung (rechts)

5.1.2 Quantitative Werkzeugverschleißerkennung

Im Vergleich zur Klassifizierung eines Werkzeugs ist die Durchführung dieses Programms deutlich aufwändiger. Es erfordert einer Kalibrierung des Modells und genaues Einstellen der Parameter für die Aufnahme eines Bildes.

5.1.2.1 Abhängigkeit der Ergebnisse von den Lichtverhältnissen

Aufgrund dessen, dass sich die LED-Lichter beim Mikroskop auf dem Raspberry Pi in diesem Programm nicht einstellen lassen, ist es deutlich schwieriger die Verschleißzone adäquat aufzunehmen. Durch einen Iterativen Prozess müssen hierbei die richtigen Lichtverhältnisse gefunden werden, um das beste Ergebnis zu erreichen. Die Gesamthelligkeit am Mikroskop kann manuell mit einem Polarisator eingestellt werden. Wird zum Aufnehmen des Bildes eine Software namens DinoCapture 2.0 verwendet, die vom Hersteller des Mikroskops zur Verfügung gestellt wird, können die sich am Mikroskop befindlichen LEDs in vier Zonen an- und ausgeschaltet werden. Zusätzlich kann die Helligkeitsstufe der LED-Lichter in sechs Stufen gedimmt werden. Dadurch wird es ermöglicht, die Verschleißfläche so aufzunehmen, dass kaum Spiegelungen oder dunklere Flächen auf der Verschleißfläche zu sehen sind, die zu Fehlinterpretationen des Programms führen können.

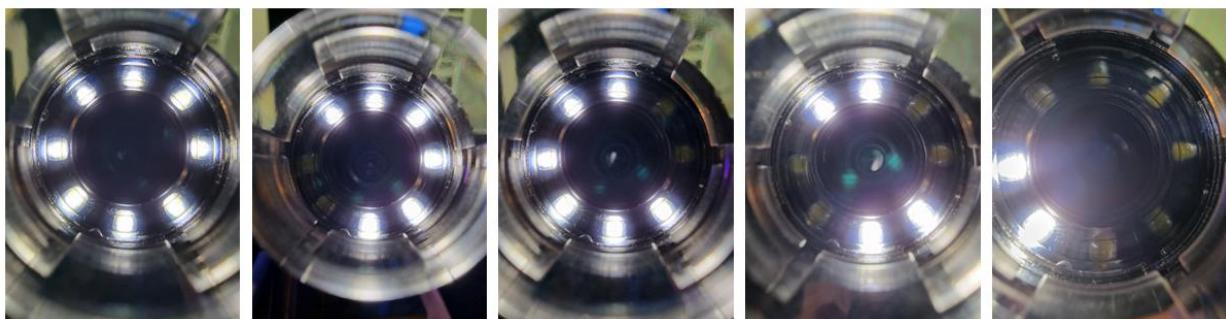


Abbildung 48: Unterschiedliche LED-Einstellungsmöglichkeiten des Dino-Lite Mikroskops

In Abbildung 49 ist zu sehen, dass vor allem dunkle Bilder Probleme bei der genauen Erkennung der Verschleißzone verursachen. Mit steigender Helligkeit werden zwar die erkannten Konturen des Verschleißes genauer, jedoch nur bis zu einem gewissen Punkt. Bei mittlerer Helligkeit sind demnach die besten Ergebnisse zu erwarten. Dieses Ergebnis verglichen mit dem zuvor, mittels DinoCapture 2.0 Software aufgenommenen Bild zeigt, dass die Kontur des Verschleißes aufgrund der besseren Lichtverhältnisse deutlich eindeutiger abgebildet werden konnte. Demnach ist es für eine zuverlässige Anwendung in einer Werkzeugmaschine essenziell, dass diese über ein Mikroskop verfügt, das variable, frei steuerbare LEDs oder ähnliches besitzt, mit dem die Lichtverhältnisse eingestellt werden können.

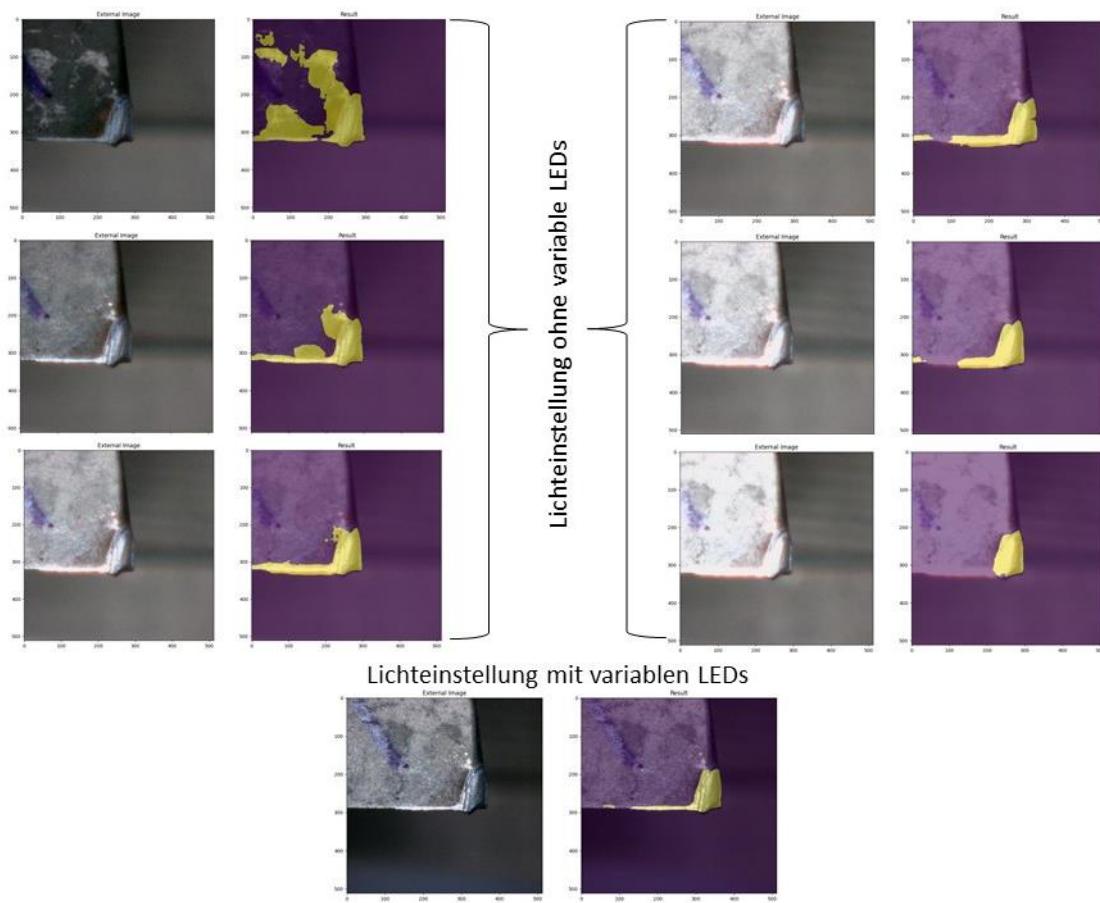


Abbildung 49: Iterative Lichteinstellung für das beste Ergebnis im Vergleich zur variablen Einstellung der LEDs mittels DinoCapture Software (unten)

5.1.2.2 Kalibrierung

Im ersten Schritt des Programms muss das Mikroskop kalibriert werden. Dazu wird das Mikroskop auf den Abstand eingestellt, bei dem die Verschleißzone vollständig und scharf abgebildet wird. Diese Position darf ab diesem Zeitpunkt nicht mehr verändert werden, da sonst eine Neukalibrierung notwendig ist. Bei der Kalibrierung wird ein „Calibration Target“ verwendet, das zum Dino-Lite Mikroskop gehört, auf dem Maßlinien abgebildet sind, die als Referenz verwendet werden können. Mit diesem wird ein Bild aufgenommen und der Abstand zwischen zwei Punkten markiert.



Abbildung 50: Calibration Target eines Dino-Lite Mikroskops

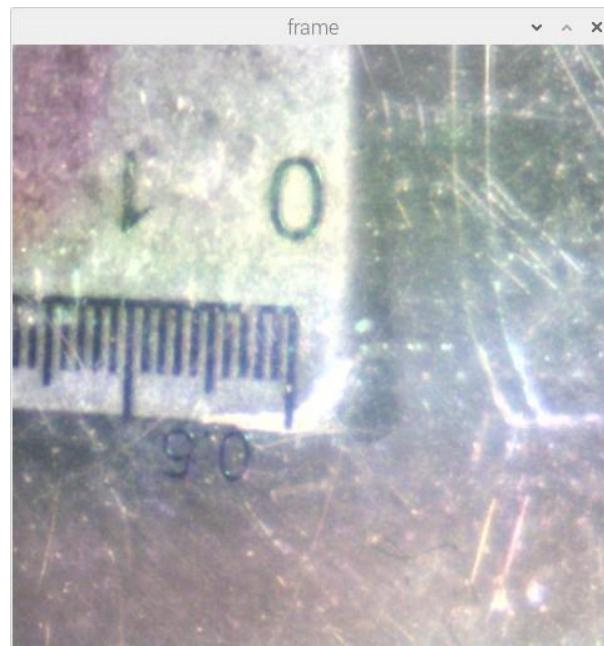


Abbildung 51: Markierung des Abstandes zwischen zwei Werten zur Kalibrierung
(Fenstergröße: 960x960 Pixel)

Die Pixel werden gezählt und die Anzahl ausgegeben. Der in Abbildung 51 zu sehende Abstand zwischen 0 mm und 1 mm entspricht 129,097 Pixeln. Im nächsten Schritt wird die Kantenlänge der einzelnen Pixel berechnet. Dabei wird die Distanz zwischen den zwei Punkten auf dem Calibration Target durch die Anzahl der Pixel geteilt. Im Beispiel sind dies $\frac{1 \text{ mm}}{129,097 \text{ Pixel}} = 0,00775 \frac{\text{mm}}{\text{Pixel}}$. Um zu validieren, ob die Kalibrierung erfolgreich war, wird die unveränderte Mikroskop Einstellung verwendet und der Abstand zwischen zwei Punkten auf dem Calibration Target gemessen.

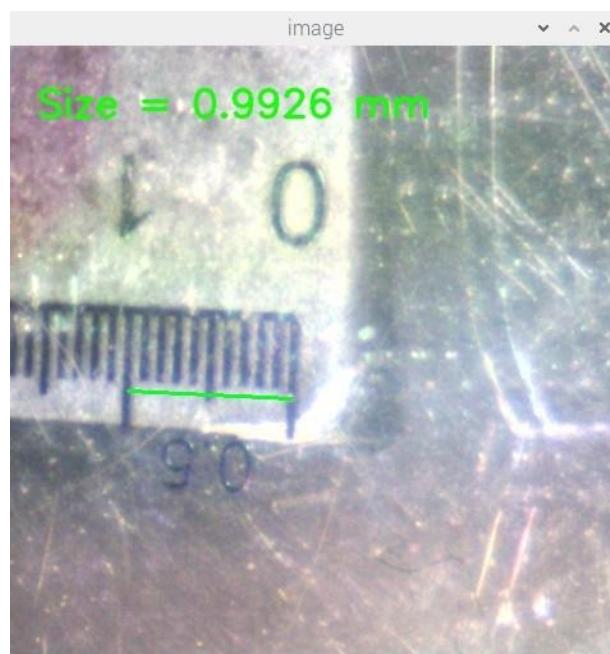


Abbildung 52: Validierung der Kalibrierung (Fenstergröße: 960x960 Pixel)

Nachdem geprüft wurde, ob die Kalibrierung erfolgreich war, kann das quantitative Verschleißidentifizierungsprogramm den Verschleiß erkennen und auswerten. Die Bilder wurden mit einer Auflösung von 960x960 Pixeln aufgenommen. Mithilfe von cv2.CAP_PROP_FRAME_WIDTH und cv2.CAP_PROP_FRAME_HEIGHT kann das Fenster nicht auf 512x512 Pixeln reduziert werden, da automatisch vom Mikroskop die nächstgelegene Auflösung verwendet wird, die das Mikroskop darstellen kann. Diese wäre demnach 640x480 Pixel, wobei beim Beschneiden des Bildes auf eine quadratische Größe die resultierende Kantenlänge 480 Pixel wäre. Das ist aber bei der Skalierung auf 512x512 Pixel mit einem kleinen Informationsverlust verbunden. Tests haben ergeben, dass bei einer nativen Auflösung von 1280x960 Pixeln, mit anschließendem beschneiden auf 960x960 Pixel und skalieren auf 512x512 Pixel, bessere Ergebnisse zu erwarten sind.

Nachdem die Bilder auf die richtige Größe reduziert wurden, können diese mittels Verschleißerkennungsprogramm ausgewertet und die Größe des Verschleißes gemessen werden.

Die Verschleißfläche wird gemessen, indem die Anzahl der gelben Pixel mit der Größe der Pixel, gemessen durch die Kalibrierung, multipliziert wird. Die resultierende Pixelgröße muss zuvor um den Faktor $\frac{960}{512} = 1,875$ skaliert werden, da die Kalibrierung mit einer Auflösung von 960x960 Pixeln durchgeführt wurde. Die Messung der Verschleißfläche erfolgt jedoch mithilfe des Ausgabebilds mit der Größe von 512x512 Pixeln. Die Verkleinerung wurde mittels der Image.resize() Methode ausgeführt, die die Bilder komprimiert aber nicht beschneidet. Dadurch werden die Pixel relativ zur abgebildeten Fläche größer, da mit einem Pixel mehr Fläche dargestellt wird. Demnach ist mit Skalierung die Anzahl der Pixel der Fläche kleiner als bei einer Auflösung von 960x960 Pixeln und das Ergebnis ist verfälscht.

$$\text{Kantenlänge} = \frac{\text{Distanz auf Calibration Target}}{\text{Anzahl Pixel Abstand}} * \frac{960}{512} \quad \text{Formel 5.1.2.2.1}$$

$$\text{Verschleißfläche} = \text{Kantenlänge}^2 * \text{Anzahl gelber Pixel} \quad \text{Formel 5.1.2.2.2}$$

5.1.2.3 Zusammenspiel aller Programme

Ein schematischer Ablauf für das Zusammenspiel aller Programme ist in Abbildung 53 dargestellt. Die Anwendung in einer Maschine kann nach folgendem Ablauf stattfinden:

In Kombination mit der Steuerung der Maschine kann je nach Klassifizierungsaufgabe ein unterschiedlicher Abstand zur Verschleißfläche für die Aufnahme der Bilder programmiert werden. Dabei kann auf Basis der Information des Klassifizierungsprogramms je nach Werkzeug eine unterschiedliche Verschleißmessung durchgeführt werden. Die benötigten Abstände zu Verschleißmessung können je nach Werkzeugart gespeichert werden. Die Kalibrierung fällt weg, solange sich am Abstand nichts ändert. Danach kann je nach Werkzeugart die Verschleißmessung durchgeführt und ausgewertet werden.

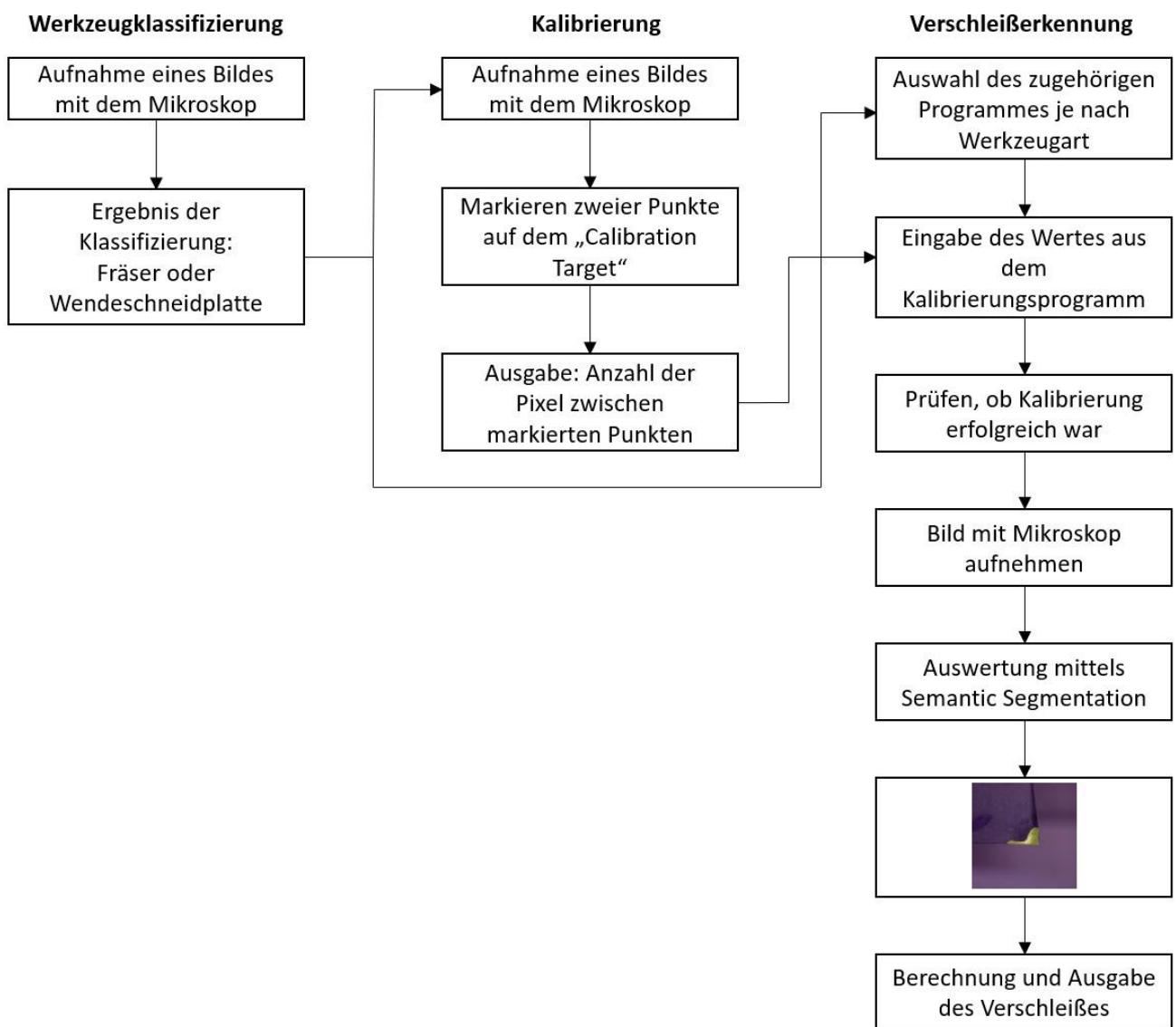


Abbildung 53: Schematischer Ablauf aller drei Programme im Zusammenhang

5.1.2.4 Ergebnisse der Kameraimplementierung im Raspberry Pi

In Tabelle 2 sind Bilder zu sehen, die mittels Verschleißerkennung und vorheriger Kalibrierung ausgewertet wurden. Es fällt auf, dass die Ergebnisse brauchbar sind, trotz nicht einstellbarer LEDs. Mit kleinerer Verschleißfläche auf der Wendeschneidplatte sind bessere Ergebnisse zu erwarten. Mit größerer Verschleißfläche werden auch die Ungenauigkeiten größer. Insgesamt sind diese Ergebnisse aber aufgrund der zu hohen Ungenauigkeit in dieser Form nicht geeignet, um eine genaue Verschleißflächenquantifizierung durchzuführen. Dieses Problem konnte behoben werden, indem weitere Bilder zur Datenbasis hinzugefügt und das Programm neu trainiert wurde. Die zusätzlichen Aufnahmen des Mikroskops enthalten Bilder ohne variable LEDs. Dabei konnte die Genauigkeit verbessert werden, wodurch eine Anwendung zur genauen Verschleißquantifizierung möglich gemacht wurde. Weiterhin spielen einstellbare Lichtverhältnisse eine große Rolle, damit die Auswertung der Bilder schneller und zuverlässiger durchgeführt werden kann.

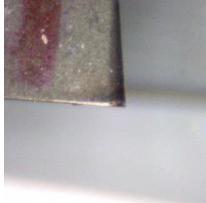
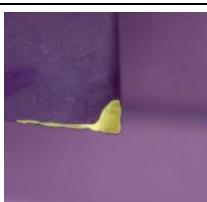
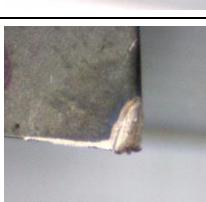
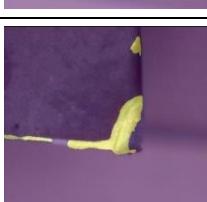
Aufgenommenes Bild	Markierte Verschleißfläche	Verschleißfläche [mm ²]
		0,124
		0,200
		0,295
		0,329
		0,375
		0,419
		0,649

Tabelle 2: Auswertung der Verschleißbilder mit entsprechender Verschleißfläche

5.1.3 Überprüfung der Genauigkeit der Verschleißflächenmessung

Unabhängig von Abstand des Mikroskops von der aufzunehmenden Verschleißfläche sollte das gleiche oder ein ähnliches Ergebnis erreicht werden. Um zu validieren, ob die Verschleißmessung mit dieser Methode breit angewendet werden kann, wurden 13 Messungen vorgenommen, bei denen das Mikroskop jeweils unterschiedliche Abstände zur Wendeschneidplatte hatte. Diese Versuchsreihe wurde bei zwei unterschiedlichen Wendeschneidplatten durchgeführt. Eine der beiden mit einer größeren Verschleißfläche und die andere mit einer kleineren Verschleißfläche. Die Abstände wurden beginnend von 80 mm über der Wendeschneidplatte aufgenommen und jeweils in 5 mm Schritten verringert. Dieser Bereich wurde gewählt, da ein größerer Abstand für die Verschleißmessung zu einem Nachteil führen kann. Ein Abstand kleiner als 20 mm ist nicht möglich, da das Mikroskop ab diesem Wert keine scharfen Bilder mehr erzeugen kann.

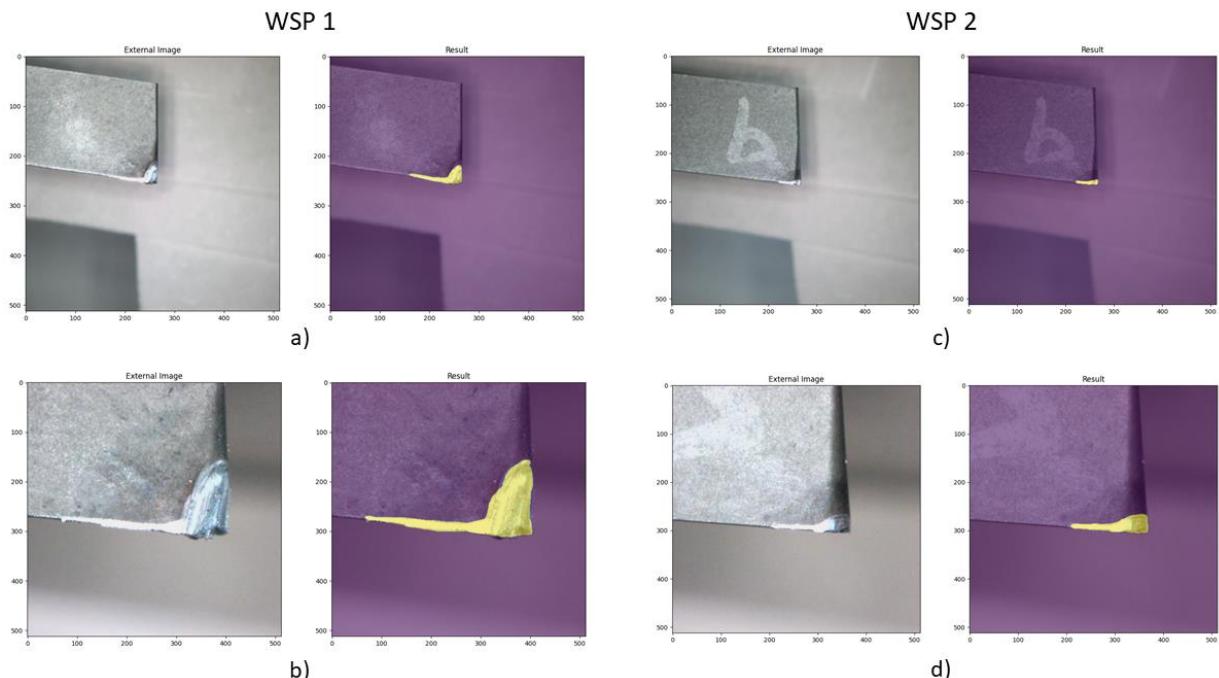


Abbildung 54: Übersicht der Verschleißmessungen. a) und b) zeigen WSP 1 mit 80 mm und 20 mm Abstand, c) und d) zeigen WSP 2 mit 80 mm und 20 mm Abstand

In Abbildung 54 ist zu sehen, dass die aufgenommene Verschleißfläche bei einem geringeren Abstand einen größeren Bereich im Bild einnimmt. Dadurch ist die Auswertung des Verschleißes stabiler und weniger empfindlich gegen eine fehlerhafte Verschleißerkennung. Aus den Diagrammen aus Abbildung 55 und Abbildung 56 ist zu entnehmen, dass bei größerem Abstand die Werte für die Größe der Verschleißfläche große Schwankungen besitzen. Mit kleiner werdendem Abstand werden die Schwankungen zwischen den Werten immer kleiner. Der Mittelwert der Verschleißfläche von WSP 1 beträgt $0,765 \text{ mm}^2$ und der Mittelwert der WSP 2 beträgt $0,163 \text{ mm}^2$. Dabei entspricht die Standardabweichung bei WSP 1 $0,046 \text{ mm}$ und bei WSP 2 $0,0112 \text{ mm}$. Diese Abweichung entspricht ca. einem Anteil von 6,01% des Mittelwerts von WSP 1 und 6,87% des Mittelwerts von WSP 2. Auch wenn der Anteil der Abweichung prozentual bei WSP 2 nur gering ausfällt, ist dennoch zu sehen, dass bei kleinen Verschleißflächen die Streuung der Werte größer ist als bei größeren Verschleißflächen. Insgesamt ist eine Abweichung im Schnitt von ca. 6,5% als hoch zu bewerten. Die Abweichung ist vor allem verursacht durch Fehlauswertungen des Verschleißes bei größeren Abständen. Des Weiteren ist die Genauigkeit

der Kalibrierung bei größeren Abständen schlechter, da der Abstand der Punkte auf dem Calibration Target visuell kleiner wird und es so schwieriger ist, genau die Strichlinien zu markieren. Bei größeren Abständen können selbst kleine Abweichungen zu größeren Fehlern führen. Dies ist bei einer größeren Vergrößerung nicht der Fall. Hinzu kommt, dass das Bilderkennungsprogramm bei kleineren Abständen die Verschleißzone besser abgrenzen kann, wodurch die Vorhersagen genauer sind als bei größeren Abständen. Dies, zusammen mit der ungenauerer Kalibrierung, führt zu einer größeren Streuung der Werte. Da in einem Anwendungsfall jedoch in der Regel immer der gleiche Abstand eingehalten wird und dieser auf eine große Vergrößerung kalibriert ist, ist zu erwarten, dass die Genauigkeit der Vorhersagen für eine quantitative Verschleißauswertung ausreicht, da schon ab einem Abstand von 50 mm stabile Ergebnisse zu erwarten sind.

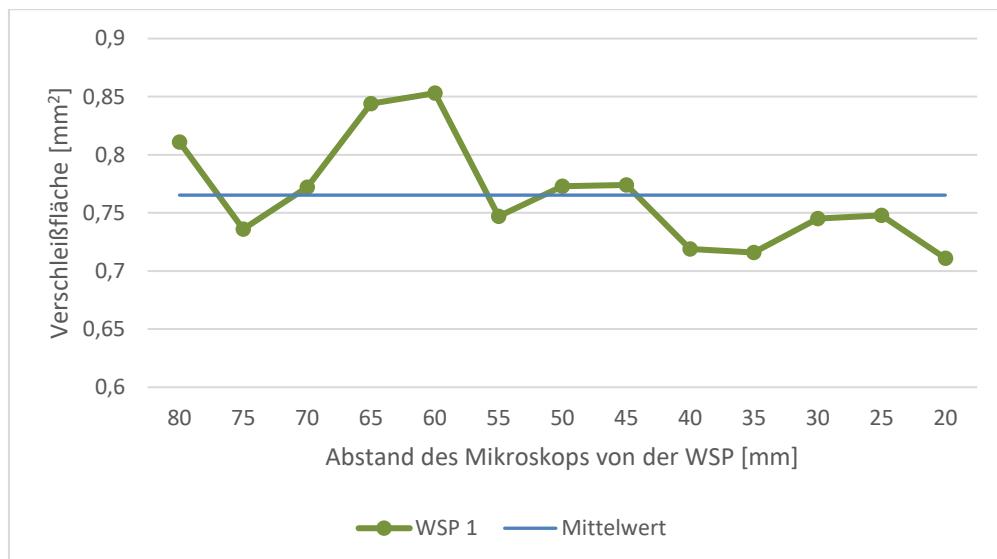


Abbildung 55: Werte der Verschleißfläche von WSP 1 in Abhängigkeit vom Abstand

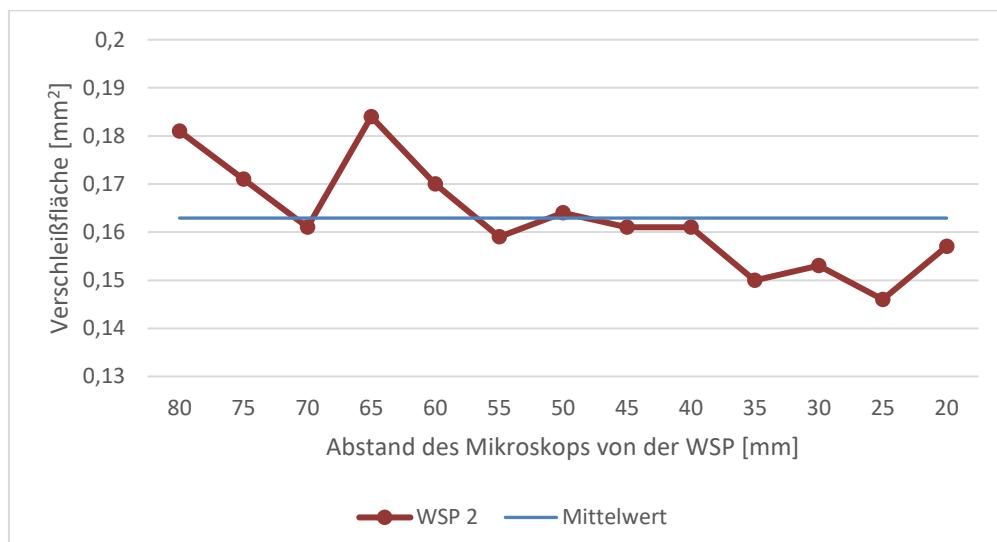


Abbildung 56: Werte der Verschleißfläche von WSP 2 in Abhängigkeit vom Abstand

Um einen Anwendungsfall zu simulieren, wurde in einem Abstand von 35 mm mehrmals dieselbe Verschleißfläche einer weiteren WSP gemessen. Dabei wurde in einer Versuchsreihe die Position der Verschleißfläche im Bildausschnitt des Mikroskops variiert und in einer anderen Versuchsreihe die Verschleißfläche immer an der gleichen Position gehalten. Die zugehörigen Verschleißwerte wurden notiert und in einer Tabelle zusammengefasst. Es ergibt sich folgendes Diagramm:

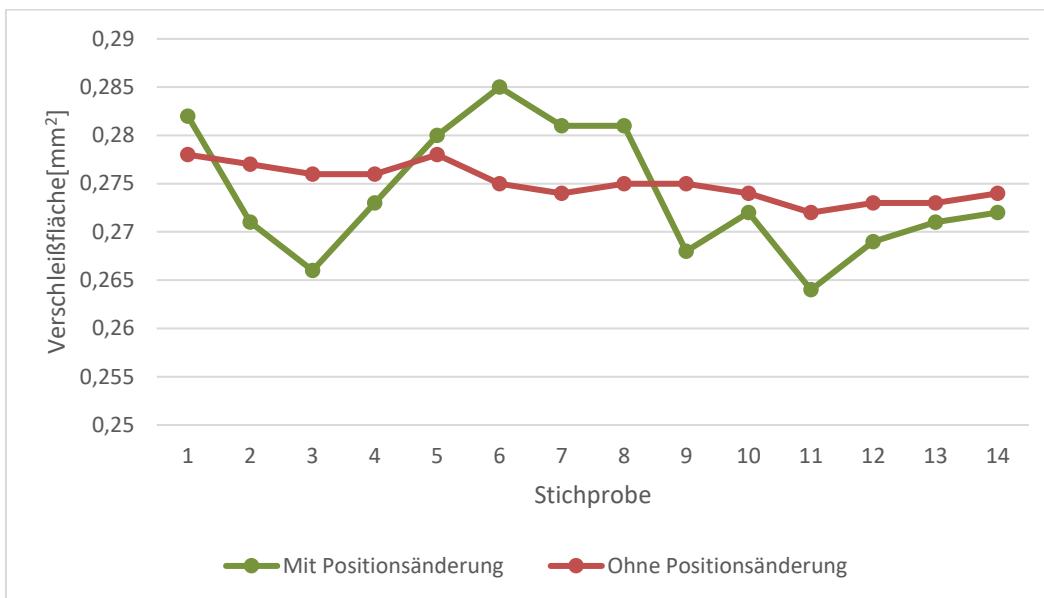


Abbildung 57: Resultate der Verschleißflächenmessung mit Positionsänderung und ohne Positionsänderung

Anhand der Verläufe im Diagramm ist zu sehen, dass die Genauigkeit der Verschleißflächenmessung deutlich höher ist, wenn die Position der zu messenden Fläche nicht variiert wird. Bei einer variablen Positionierung beträgt die maximale Abweichung $0,021 \text{ mm}^2$ und bei einer nicht variablen Positionierung nur $0,006 \text{ mm}^2$. Die Abweichung bei der variablen Positionierung ist um das 3,5-Fache größer. Ein möglicher Grund hierfür ist, dass die Verschleißfläche, je nach Position im Bild, nicht vollständig parallel zu Kamera ausgerichtet ist. Hinzu kommt, dass sich die Lichtverhältnisse mit der Positionsänderung ändern. Es kann vorkommen, dass an bestimmten Stellen die Fläche anders belichtet wird, wodurch Pixel als Verschleiß klassifiziert werden, die nicht zur Verschleißfläche gehören oder umgekehrt.

6 Bewertung

Machine-Learning Bilderkennungsprogramme können in einer Vielzahl von Anwendungsfällen genutzt werden. Es liegt daher nahe, diese vielseitigen Tools auch bei der intelligenten Verschleißerkennung einzusetzen. Die Image Classification und damit verbundene Werkzeugklassifizierung bietet einen ersten Einstieg in die ML-Bilderkennung. Diese Klassifizierung ist in ihrer Komplexität und nötigen Rechenleistung gering. Auch mit einer kleinen Datenbasis können durch Data Augmentation bereits sehr gute Ergebnisse erzielt werden. Auf Basis dieses Klassifizierungsprogramms können Programme, wie die quantitative Verschleißerkennung gezielt eingesetzt werden, um die Ergebnisse so weiter zu verbessern. Beispielsweise kann aufgrund der Information der Werkzeugklassifizierung ein bestimmter Abstand oder eine bestimmte Position des Mikroskops eingestellt werden, um optimale Voraussetzungen für die Aufnahme der Verschleißfläche zu schaffen.

Durch die quantitative Verschleißerkennung kann garantiert werden, dass ein Werkzeug wirtschaftlich eingesetzt wird und gleichzeitig die Qualität der bearbeiteten Oberflächen garantiert ist. Mittels einer Verschleißüberwachung im Prozess kann früh erkannt werden, ob der Verschleiß eines Werkzeugs zu groß ist, um die geforderte Qualität einhalten zu können oder ob das Werkzeug weiter eingesetzt werden kann. Dadurch ist es möglich, dass die Standzeit eines Werkzeugs voll ausgeschöpft werden kann, da mithilfe der Überwachung ein Werkzeug nicht zu früh gewechselt wird. Zusätzlich werden Maschinenstillstandzeiten verringert, da die Werkzeuge seltener gewechselt werden müssen. Es ergibt sich ein hohes Kosteneinsparpotential. Dies wird dadurch ermöglicht, dass die Verschleißerkennung in einer Werkzeugmaschine hohe Genauigkeiten erreicht und den Verschleiß konsistent wiedergeben kann.

Mithilfe von Tensorflow und Keras kann ein ML-Bilderkennungsprogramm unkompliziert erstellt werden. Diese Plattformen bieten eine Möglichkeit derartige Programme zu erstellen, ohne selbst die mathematischen Abläufe programmieren zu müssen. Dies bietet sich besonders im Bereich der Forschung an, da jeder die Möglichkeit besitzt, ML-basierte Anwendungen erstellen zu können, ohne tiefergehende Programmierkenntnisse haben zu müssen. Die Erstellung der dazu nötigen Datenbasen kann mit zahlreichen frei zugänglichen Tools durchgeführt werden. Dabei ist die größte Herausforderung eine adäquate Datenbasis zu erstellen, die die Variationen in einem Anwendungsfall bewältigen kann, um einsatzfähig zu sein.

Ein großer Vorteil der ML-Bilderkennungsprogramme ist, dass das Grundgerüst der Programme auf verschiedenste Anwendungsfälle angewendet werden kann. Hierfür bedarf es lediglich neuer Datenbasen, die den jeweiligen Anwendungsfall repräsentieren. Ist es beispielsweise notwendig, die quantitative Verschleißerkennung um ein weiteres Werkzeug zu erweitern, kann dasselbe Programm verwendet werden. Es wird nur einen neuen Datenbasis benötigt.

Die ML-Bilderkennungsprogramme tragen dazu bei, die Automatisierung voranzutreiben und besitzen dabei das Potential einer großen Kostensparnis.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Gemäß der Aufgabenstellung wurde ein Überblick geschaffen, wie neue Techniken auf Basis von Image Processing und Machine Learning mit klassischen Bildbearbeitungsprozessen kombiniert werden können, um so zur Steigerung der Prozessleistung und zur Verringerung von Maschinenstillstandzeiten beizutragen.

In dieser Arbeit wurde untersucht, wie Machine-Learning Ansätze zur Werkzeugklassifizierung und quantitativen Werkzeugverschleißerkennung eingesetzt werden können. Mithilfe von Tensorflow und Keras wurden Bilderkennungsprogramme erstellt, die zum einen ein Werkzeug zu einer Wendeschneidplatte oder einem Schaftfräser zuordnen und zum anderen die Verschleißfläche eines Werkzeugs erkennen und markieren. Anschließend wurden diese Bilder ausgewertet und der Verschleiß quantitativ bestimmt. Im Anschluss wurde die Genauigkeit validiert, indem durch unterschiedliche Positionierung und Kameraentfernung die Ergebnisse verglichen wurden.

Mithilfe der erstellten Bilderkennungsprogramme konnten die Werkzeuge richtig klassifiziert werden. Anschließend konnte der Werkzeugverschleiß mit einer hohen Genauigkeit gemessen werden. Selbst bei einer Positionsveränderung könnte die Verschleißflächenmessung in dieser Weise aufgrund der hohen Genauigkeit Anwendung in einer industriellen Umgebung finden. Diese Verschleißmessung wurde in dieser Form noch nicht angewendet und stellt eine Verbesserung im Vergleich zur konventionellen Verschleißmessung dar, bei der die Verschleißmarkenbreite VB manuell gemessen wird. Die Betrachtung der kompletten Verschleißfläche sorgt damit für eine höhere Genauigkeit und hebt sich dadurch von der konventionellen Verschleißmessung ab. Zusätzlich ist durch die autonome Verschleißerkennung eine Verbesserung der Auswertungsgeschwindigkeit und Konsistenz zu erwarten.

7.2 Ausblick

Diese Arbeit stellt das Potential der ML-Bilderkennung in der quantitativen Verschleißerkennung und deren Möglichkeiten zum Einsatz dar. Es bleiben jedoch Probleme, wie zum Beispiel die Empfindlichkeit gegen veränderte Lichtbedingungen und die Beschränkung auf den Freiflächenverschleiß bei Wendeschneidplatten, die die Leistung des Programmes verringern und gelöst werden müssen.

Da das Erstellen einer Datenbasis aufgrund des Labeling-Prozesses sehr viel Zeit in Anspruch nimmt, konnte aufgrund der limitierten Zeit zur Erstellung der Bachelorarbeit nur eine kleine Datenbasis erstellt werden. Damit eine konstante Leistung auch unter schwierigeren Bedingungen erreicht werden kann, müssen mehr Bilder gesammelt und gelabelt werden.

Auf Grundlage der Informationen aus der Werkzeugklassifizierung können jeweils werkzeugspezifische Semantic Segmentation Ansätze verwendet werden. So kann beispielsweise der Ansatz von Lin et al. (2021) verwendet werden, um den Verschleiß bei Fräsern zu quantifizieren, damit jede Schneide des Werkzeugs ausgewertet werden kann. Zudem bietet es sich an, je nach Art des Werkzeugs oder des Verschleißes unterschiedliche Datenbasen zu erstellen und zu verwenden, um Ungenauigkeiten durch Ähnlichkeiten zu verhindern. Diese Methode entspricht dem „One-for-each“-Ansatz, bei dem für jede Werkzeugart ein eigenes Netzwerk und eine eigene Datenbasis verwendet wird.

Des Weiteren kann der Einsatz in einer Werkzeugmaschine mittels Raspberry Pi verbessert werden, indem eine LED-Steuerung des Mikroskops in einem Python-Programm ermöglicht wird.

Dabei kann ein Mikroskop dauerhaft stationär in eine Maschine implementiert werden, das über einen Autofokus verfügt, um ein manuelles Einstellen der Kamera zu umgehen und den Automatisierungsgrad weiter zu erhöhen.

Abbildungsverzeichnis

Abbildung 1: Verschleißmechanismen und -Formen (Denkena & Tönshoff, 2011)	3
Abbildung 2: Verschleißformen beim Drehen (Denkena & Tönshoff, 2011)	4
Abbildung 3: Einordnung Artificial Intelligence, Machine Learning und Deep Learning (IONOS Digital Guide, 2020)	5
Abbildung 4: Schematische Darstellung eines Perzeptrons (Bergs et al., 2020)	6
Abbildung 5: Sigmoid Function	7
Abbildung 6: Tanh Function	7
Abbildung 7: ReLu Function	7
Abbildung 8: Schematische Darstellung eines Deep Neural Networks (Bergs et al., 2020)	8
Abbildung 9: Zwei mögliche Wege des Gradient Descent (Ng, 2021)	10
Abbildung 10: Zusammenhang zwischen Cost Function und Hypothesis Function (Ng, 2021)	10
Abbildung 11: Vergleich zwischen Underfitting, Optimum und Overfitting (IBM Cloud Education, 2021)	11
Abbildung 12: Optimum zwischen Overfitting und Underfitting (IBM Cloud Education, 2021)	11
Abbildung 13: Struktur eines Convolutional Neural Networks (Lämmel & Cleve, 2020)	12
Abbildung 14: Schematische Darstellung einer Convolution Operation (Bergs et al., 2020)	13
Abbildung 15: Max-Pooling mit unterschiedlichen Schrittweiten (Lämmel & Cleve, 2020)	14
Abbildung 16: Netzwerke ohne Dropout (i) und mit Dropout (ii) (Verdhan, 2021)	14
Abbildung 17: Unterschiedliche Aufgaben von Computer Vision (Matcha, 2021)	16
Abbildung 18: U-Net Architektur (Ronneberger et al., 2015)	17
Abbildung 19: Confusion Matrix (Mishra, 2018)	18
Abbildung 20: Intersection over Union (Bergs et al., 2020)	19
Abbildung 21: Vergleich zwischen IoU und OP Accuracy (Bergs et al., 2020)	19
Abbildung 22: Ergebnisse eines CT-Scans eines menschlichen Kiefers mit Amalgamfüllungen. (a) mittels Artificial Intelligence erstellte Maske und das daraus rekonstruiertes Bild, (b) original aufgenommene Projektionsdaten und das rekonstruierte Bild, (c) manuell erstellte Maske und das daraus rekonstruiertes Bild (Stille et al., 2013).	21
Abbildung 23: Ausgewertetes Bild einer Fettleber mittels Bilderkennung (Kuzhipathalil et al., 2021)	21
Abbildung 24: Beispiel eines Semantic Segmentation Ergebnisses basierend auf einem PSPNet (Fujiyoshi et al., 2019)	22
Abbildung 25: Verschleißerkennung mit höherer und niedrigerer Helligkeit. Vergleich zwischen traditionellem CV und einem Deep Learning Ansatz (Bergs et al., 2020)	24
Abbildung 26: Vergleich unterschiedlicher Modelle für die Verschleißerkennung (Lin et al., 2021)	24

Abbildung 27: Beispiele aus der Datenbasis	25
Abbildung 28: Ablauf des Trainingsprogramms	26
Abbildung 29: Trainingsverlauf mit Overfitting	27
Abbildung 30: Beispiel Data Augmentation mit einer Wendeschneidplatte	27
Abbildung 31: Trainingsverlauf ohne Overfitting mit 50 Epochen	28
Abbildung 32: Ablauf des Auswertungsprogramms	28
Abbildung 33: Klassifizierung eines Schafffräzers und einer Wendeschneidplatte	29
Abbildung 34: Vorrichtung zur Aufnahme der Verschleißbilder	30
Abbildung 35: Beispiel eines aufgenommenen Bildes mit dem Dino Lite Digital-Mikroskop	31
Abbildung 36: Aufgenommenes Originalbild mit einer Auflösung von 1280x960 Pixeln (links) und komprimiertes, zugeschnittenes Bild mit einer Auflösung von 512x512 Pixeln (rechts)	31
Abbildung 37: Apeer.com Bearbeitungssoberfläche	32
Abbildung 38: Maske eines Verschleißbildes	32
Abbildung 39: Originalbild (links oben) und sieben erweiterte Bilder	33
Abbildung 40: Ergebnis ohne Data Augmentation (oben) im Vergleich mit Data Augmentation (unten)	34
Abbildung 41: Ablauf der Erstellung der Datenbasis und des Trainingsprogramms	36
Abbildung 42: Trainingsverlauf mit 270 Epochen	36
Abbildung 43: Ablauf des Auswertungsprogramms	37
Abbildung 44: IoU-Score in Abhängigkeit von der Anzahl der Epochen	38
Abbildung 45: Vergleich der Ergebnisse unterschiedlicher Anzahl an Epochen	38
Abbildung 46: Aufbau einer Implementierung in eine Werkzeugmaschine am Beispiel einer Index V100	39
Abbildung 47: Aufnahme mit dem Mikroskop (links), Ergebnis der Auswertung (rechts)	40
Abbildung 48: Unterschiedliche LED-Einstellungsmöglichkeiten des Dino-Lite Mikroskops	41
Abbildung 49: Iterative Lichteinstellung für das bestes Ergebnis im Vergleich zur variablen Einstellung der LEDs mittels DinoCapture Software (unten)	42
Abbildung 50: Calibration Target eines Dino-Lite Mikroskops	42
Abbildung 51: Markierung des Abstandes zwischen zwei Werten zur Kalibrierung (Fenstergröße: 960x960 Pixel)	43
Abbildung 52: Validierung der Kalibrierung (Fenstergröße: 960x960 Pixel)	43
Abbildung 53: Schematischer Ablauf aller drei Programme im Zusammenhang	45
Abbildung 54: Übersicht der Verschleißmessungen. a) und b) zeigen WSP 1 mit 80 mm und 20 mm Abstand, c) und d) zeigen WSP 2 mit 80 mm und 20 mm Abstand	47
Abbildung 55: Werte der Verschleißfläche von WSP 1 in Abhängigkeit vom Abstand	48
Abbildung 56: Werte der Verschleißfläche von WSP 2 in Abhängigkeit vom Abstand	48

Abbildung 57: Resultate der Verschleißflächenmessung mit Positionsänderung und ohne Positionsänderung 49

Tabellenverzeichnis

Tabelle 1: Verwendete Data Augmentation Operationen	33
Tabelle 2: Auswertung der Verschleißbilder mit entsprechender Verschleißfläche	46

Literaturverzeichnis

apeer.com. <https://www.apeer.com/home/>

Bartenschlager, J., Dillinger, J., Escherich Walter, Günter, W., Ignatowitz, E., Oesterle, S., Reißler, L., Stephan, A., Vetter, R. & Wieneke, F. (2013). *Fachkunde Metall* (57. Aufl.). *Europa-Fachbuchreihe für metalltechnische Berufe*. Verl. Europa-Lehrmittel Nourney Vollmer.

Bergs, T., Holst, C., Gupta, P. & Augspurger, T. (2020). Digital image processing with deep learning for automated cutting tool wear detection. *Procedia Manufacturing*, 48, 947–958. <https://doi.org/10.1016/j.promfg.2020.05.134>

Bhalley, R. (2021). *Deep Learning with Swift for TensorFlow: Differentiable Programming with Swift*. Springer eBook Collection. Apress. <https://doi.org/10.1007/978-1-4842-6330-3>

Bhattiprolu, S. (2021a). *u-net model*.

https://github.com/bnsreenu/python_for_microscopists/blob/master/204-207simple_unet_model.py

Bhattiprolu, S. (2021b). *Training and testing for semantic segmentation (Unet) of mitochondria*. https://github.com/bnsreenu/python_for_microscopists/blob/master/204_train_simple_unet_for_mitochondria.py

Brühl, V. (2019). *Big data, data mining, machine learning und predictive analytics – ein konzeptioneller Überblick*. Center for Financial Studies (Frankfurt am Main): CFS working paper series: No. 617. Universitätsbibliothek Johann Christian Senckenberg.

Choo, K., Greplova, E., Fischer, M. H. & Neupert, T. (2020). *Machine Learning kompakt: Ein Einstieg für Studierende der Naturwissenschaften. essentials*. Springer Spektrum. <http://www.springer.com/>

Denkena, B. & Tönshoff, H. K. (2011). Verschleiß Verschleiß. In B. Denkena & H. K. Tönshoff (Hrsg.), *Spanen* (S. 135–166). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19772-7_7

Fujiyoshi, H., Hirakawa, T. & Yamashita, T. (2019). Deep learning-based image recognition for autonomous driving. *IATSS Research*, 43(4), 244–252. <https://doi.org/10.1016/j.iatssr.2019.11.008>

Gottschalk, T. M., Maier, A., Kordon, F. & Kreher, B. W. (2021). Learning-based Patch-wise Metal Segmentation with Consistency Check. In C. Palm, T. M. Deserno, H. Handels, A. Maier, K. H. Maier-Hein & T. Tolxdorff (Hrsg.), *Informatik aktuell. Bildverarbeitung für die Medizin 2021: Proceedings, German Workshop on Medical Image Computing, Regensburg, March 7-9, 2021* (S. 4–9). Springer Vieweg. https://doi.org/10.1007/978-3-658-33198-6_4

Heller, M. (8. Oktober 2020). Was ist Computer Vision? COMPUTERWOCHE. <https://www.computerwoche.de/a/was-ist-computer-vision,3549884>

IBM Cloud Education. (2020). *Neural Networks*. <https://www.ibm.com/cloud/learn/neural-networks>

IBM Cloud Education. (2021). *Overfitting*. <https://www.ibm.com/cloud/learn/overfitting>

IONOS Digital Guide (Hrsg.). (2020). *Deep Learning vs. Machine Learning – was sind die Unterschiede?* <https://www.ionos.de/digitalguide/online-marketing/suchmaschinenmarketing/deep-learning-vs-machine-learning/>

About Keras. (2021). <https://keras.io/about/>

Jason Brownlee. (2019). *What is a Hypothesis in Machine Learning?* <https://machinelearningmastery.com/what-is-a-hypothesis-in-machine-learning/>

- Johanning, V. (4. September 2020). Alles zur Zukunft der Mobilität. *COMPUTERWOCHE*. <https://www.computerwoche.de/a/alles-zur-zukunft-der-mobilitaet,3549675>
- Kleeberger, M. (2021). *Künstliche Intelligenz in der Medizin*. Fraunhofer-Institut für Kognitive Systeme IKS. <https://www.iks.fraunhofer.de/de/themen/kuenstliche-intelligenz/kuenstliche-intelligenz-medizin.html>
- Kuzhipathalil, A., Thomas, A., Chand, K., Gomes Ataide, E. J., Link, A., Niemann, A., Saalfeld, S [Sylvia], Fribe, M. & Ziegler, J. (2021). A Machine Learning Approach Towards Fatty Liver Disease Detection in Liver Ultrasound Images. In C. Palm, T. M. Deserno, H. Handels, A. Maier, K. H. Maier-Hein & T. Tolxdorff (Hrsg.), *Informatik aktuell. Bildverarbeitung für die Medizin 2021: Proceedings, German Workshop on Medical Image Computing, Regensburg, March 7-9, 2021* (S. 86–91). Springer Vieweg. https://doi.org/10.1007/978-3-658-33198-6_21
- Lämmel, U. & Cleve, J. (2020). *Künstliche Intelligenz: Wissensverarbeitung - neuronale Netze* (5., überarbeitete Auflage). Hanser. <https://www.hanser-fachbuch.de/buch/Kuenstliche+Intelligenz/9783446459144>
- Lin, W.-J., Chen, J.-W., Jhuang, J.-P., Tsai, M.-S., Hung, C.-L., Li, K.-M. & Young, H.-T. (2021). Integrating object detection and image segmentation for detecting the tool wear area on stitched image. *Scientific reports*, 11(1), 19938. <https://doi.org/10.1038/s41598-021-97610-y>
- Long, J., Shelhamer, E. & Darrell, T. (2014, 14. November). *Fully Convolutional Networks for Semantic Segmentation*. <https://arxiv.org/pdf/1411.4038>
- Mahony, N. O., Campbell, S., Carvalho, A., Harapanahalli, S., Velasco-Hernandez, G., Kraljova, L., Riordan, D. & Walsh, J. (2020). Deep Learning vs. Traditional Computer Vision, 943. <https://doi.org/10.1007/978-3-030-17795-9>
- Matcha, A. C. N. (2021). *A 2021 guide to Semantic Segmentation*. <https://nanonets.com/blog/semantic-image-segmentation-2020/>
- Messmann, H. (2021). Artificial Intelligence in Endoscopy. In C. Palm, T. M. Deserno, H. Handels, A. Maier, K. H. Maier-Hein & T. Tolxdorff (Hrsg.), *Informatik aktuell. Bildverarbeitung für die Medizin 2021: Proceedings, German Workshop on Medical Image Computing, Regensburg, March 7-9, 2021* (S. 3). Springer Vieweg. https://doi.org/10.1007/978-3-658-33198-6_3
- Mishra, A. (2018). *Metrics to Evaluate your Machine Learning Algorithm*. <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>
- Ng, A. (2021). *Machine Learning*. <https://www.coursera.org/learn/machine-learning/home/welcome>
- Radeck, B. (4. Juni 2018). Computer Vision – Eine Übersicht. *AIM Agile IT Management*. <https://www.agile-im.de/2018/06/04/computer-vision-eine-uebersicht/>
- Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. & Savarese, S. (2019). *Generalized Intersection over Union*. <https://giou.stanford.edu/>
- Ronneberger, O., Fischer, P. & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells & A. F. Frangi (Hrsg.), *Lecture Notes in Computer Science. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (Bd. 9351, S. 234–241). Springer International Publishing. https://doi.org/10.1007/978-3-319-24574-4_28
- Rudolph, S. (2021). *Künstliche Intelligenz (KI) und maschinelles Lernen*. <https://www.iks.fraunhofer.de/de/themen/kuenstliche-intelligenz.html>

- Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S [Stephan], Schmid, B., Tinevez, J.-Y., White, D. J., Hartenstein, V., Eliceiri, K., Tomancak, P. & Cardona, A. (2012). Fiji: an open-source platform for biological-image analysis. *Nature methods*, 9(7), 676–682. <https://doi.org/10.1038/nmeth.2019>
- Schmidt, M. & Steinecke, H. (2020). Heimische Pflanzen mit dem Smartphone bestimmen – ein Praxistest. *Der Palmengarten*, 83(2), 138–140. <https://doi.org/10.21248/palmengarten.517>
- Shorten, C. & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1). <https://doi.org/10.1186/s40537-019-0197-0>
- Informationen zu Fotos und zur Umgebung abrufen - Android - Google Fotos-Hilfe*. (2022, 1. Januar).
<https://support.google.com/photos/answer/7539151?hl=de&co=GENIE.Platform%3DAndroid>
- Stahl, T., Koch, C. & Wersig, S. (2019). *Künstliche Intelligenz in der Praxis: AUTOENCODER – EINE VIELSEITIG EINSETZBARE ARCHITEKTUR*.
https://www.bminformatik.de/fileadmin/download/Eigene_Veroeffentlichungen/AI_Spektrum_2019-06_Autoencoder_eine_Vielseitig_einsetzbare_Architektur.pdf
- Stille, M., Kratz, B., Müller, J., Maass, N., Schasiepen, I., Elter, M., Weyers, I. & Buzug, T. M. (2013). Influence of metal segmentation on the quality of metal artifact reduction methods. In R. M. Nishikawa & B. R. Whiting (Hrsg.), *SPIE Proceedings, Medical Imaging 2013: Physics of Medical Imaging* (86683C). SPIE. <https://doi.org/10.1117/12.2006810>
- TensorFlow. (2018). *Image classification*.
<https://github.com/tensorflow/docs/blob/master/site/en/tutorials/images/classification.ipynb>
- Verdhan, V. (2021). *Computer Vision Using Deep Learning: Neural Network Architectures with Python and Keras*. Springer eBook Collection. Apress. <https://doi.org/10.1007/978-1-4842-6616-8>
- Wu, X., Liu, Y [Yahui], Zhou, X. & Mou, A. (2019). Automatic Identification of Tool Wear Based on Convolutional Neural Network in Face Milling Process. *Sensors (Basel, Switzerland)*, 19(18). <https://doi.org/10.3390/s19183817>
- Wuttke, L. (10. September 2021). Bilderkennung: Definition, Funktionsweise und Anwendungsbereiche. *datasolut GmbH*. <https://datasolut.com/bilderkennung/>
- Zaccone, G. (2016). *Getting started with tensorflow: Get up and running with the latest numerical computing library by Google and dive deeper into your data! Community experience distilled*. Packt Publishing.
<https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1295353>
- Zhu, X., Liu, Y [Yan], Liu, X. & Li, C. (2019). Convolutional Neural Networks for Finance Image Classification. In W. K. Wong (Hrsg.), *Advances in Intelligent Systems and Computing. Artificial Intelligence on Fashion and Textiles* (Bd. 849, S. 237–245). Springer International Publishing. https://doi.org/10.1007/978-3-319-99695-0_29