Ein Bild, das Text, Schrift, Screenshot, Logo enthält.

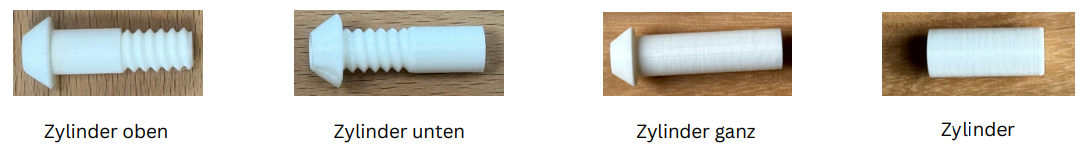
Automatisch generierte Beschreibung

3D-Anomaly

Projektarbeit von Tobias Saur, Niclas Hart und Jakob Kramer im Rahmen der Zweitjahresprojekte im Sommersemester 2024

Ein Bild, das Schraube, Metallwaren enthält.

Automatisch generierte Beschreibung



Inhaltsverzeichnis

[1. Einleitung 3](#_Toc171077849)

[2. Erstellung des Bilddatensatzes 4](#_Toc171077850)

[4. Implementierung und Evaluierung der 3 Modellansätze 5](#_Toc171077851)

[4.1 CNN 5](#_Toc171077852)

[4.1.1 Implementierung 5](#_Toc171077853)

[4.1.2 Auswertung 8](#_Toc171077854)

[5. Fazit und mögliche Erweiterungen 10](#_Toc171077855)

## Einleitung

Unser Projekt "3D-Anomaly" hat das Ziel, die Möglichkeiten von Künstlicher Intelligenz in der industirellen Qualitätssicherung aufzuzeigen. Hierbei werden insbesondere Bildverarbeitungstechniken und maschinelles Lernen eingesetzt, um fehlerhafte Produkte anhand von Kamerabildern in Echtzeit zu identifizieren. Dies kann zu einer Effizienzsteigerung und Kostenreduktion in der Fertigung führen.

In der modernen industriellen Produktion spielt die Qualitätssicherung eine entscheidende Rolle, um den hohen Anforderungen an die Produktqualität gerecht zu werden. Traditionelle Methoden der Fehlererkennung durch visuelle Inspektionen durch Mitarbeiter stoßen jedoch oft an ihre Grenzen. Hier setzen fortschrittliche Techniken wie das Template Matching und Convolutional Neural Networks (CNNs) an, die durch ihre Fähigkeit zur Mustererkennung und -klassifizierung neue Möglichkeiten für die automatisierte Qualitätskontrolle bieten.

Im Rahmen des Projekts wurde ein Bilddatensatz von 3D-gedruckten Bolzen aus verschiedenen Perspektiven (seitlich und von oben) erstellt und mittels Data Augmentation erweitert. Es wurden drei Ansätze zur Anomalieerkennung getestet: YOLO (You Only Look Once), ein vortrainiertes CNN, das sich durch hohe Geschwindigkeit und Genauigkeit auszeichnet und besonders in der Industrie weit verbreitet ist. Template Matching ist ideal für die Erkennung unbekannter Fehler, da es kein vortrainiertes Modell benötigt. Außerdem wurde auch ein CNN selbst aufgebaut und trainiert und hinsichtlich seiner Performance evaluiert.

Die Ergebnisse zeigen, dass alle drei Ansätze wertvolle Beiträge zur kamerabasierten Anomalieerkennung und somit zur Qualitätssicherung in der Industrie leisten können.

## Erstellung des Bilddatensatzes

Zu Beginn des Projekts gab es kein vorgegebenes 3D-Druck Modell, das für die Bearbeitung des Projekts verwendet werden sollte. Nach kurzer Recherche fiel die Wahl auf ein fertiges Modell eines 25x8 Bolzens von Thingiverse (<https://www.thingiverse.com/thing:193647/files>). Aufgrund der potenziellen Nutzbarkeit im industriellen Umfeld wurde dieses Modell gewählt.

Anschließend wurden dieser Bolzen zum ersten Mal 3D-gedruckt. Dabei wurde ersichtlich, dass es bei dieser Bauteilkomplexität nicht automatisch zu Fehlern beim Druck kommen würde. Aus diesem Grund wurden mittels Tinkercad die Modelle künstlich manipuliert, um Fehler zu erzeugen, welche die Nutzbarkeit des Bauteils erheblich einschränken. Dabei wurde zum Beispiel das Gewinde teilweise oder gänzlich entfernt oder der Ansatz für den Sechskant-Schlüssel entfernt.

Um tatsächlich eine vollständige Überprüfung der Bauteile zu gewährleisten, sind Bildaufnahmen aus der seitlichen Ansicht (Prüfung des Gewindes) und von oben (Schraubansatz) notwendig. In diesem Zug wurden folgende Fehlerklassen definiert und Bilder in verschiedenen Positionen, mit verschiedenen Hintergründen und unterschiedlicher Beleuchtung aufgenommen:

1. Standard-Schraube (ohne Fehler): 104 Bilder seitlich / 40 Bilder oben

Ein Bild, das Bandage, Im Haus enthält.

Automatisch generierte BeschreibungEin Bild, das Metallwaren, Schraube enthält.

Automatisch generierte Beschreibung

1. Schraube-Zylinder\_ganz: 63 Bilder seitlich / 18 Bilder oben

Ein Bild, das Boden, Im Haus, Papier enthält.

Automatisch generierte Beschreibung

TO-DO: hier noch gemäß Original-Data in Gitlab fertig machen

## Implementierung und Evaluierung der 3 Modellansätze

Im Folgenden wird die Implementierung der drei ausgewählten Modellansätze (Template Matching, YOLO, CNN) genauer erläutert. Außerdem werden die drei Modelle hinsichtlich ihrer Performance auf dem Datensatz, sowie weiterer Gesichtspunkte evaluiert.

## CNN

Beim ersten Modellansatz handelt es sich um ein Convolutional-Neural-Network, das selbständig aufgebaut wurde (Schichten) und das vollständig auf dem erstellten Datensatz mit den 3D-Druck Bildern trainiert, validiert und getestet wurde.

## Implementierung

In diesem Abschnitt wird die Funktionsweise für das Modell der seitlichen Ansicht genauer erklärt. Das Modell, das die obere Ansicht betrachtet funktioniert analog und wird deshalb hier nicht genauer betrachtet.

Zu Beginn des Skriptes werden zunächst die Hyperparameter eingestellt, die für den Trainingsdurchlauf beziehungsweise den Programmablauf verwendet werden sollen.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Die Bildgröße wurde final auf 224x224 Pixel eingestellt, da dies sowohl eine gute Genauigkeit aber auch akzeptablen Speicher- und Zeitbedarf für das Training bedeutete. Außerdem kann eingestellt werden, ob der Datensatz mittels Data Augmentation künstlich erweitert werden soll und wie viele augmentierte Bilder pro Originalbild erstellt werden sollen. Außerdem müssten die Epochenzahl und die Batch-Size festgelegt werden.

Als Nächstes erfolgt das Laden der Bilder aus dem angegebenen Ordner. Die Labels werden dabei anhand des Ordnernamens erstellt, in dem die Bilder enthalten sind. Außerdem erfolgt die Unterteilung in Testdaten (58%) und Validierungs- beziehungsweise Testdaten (jeweils 21%). Die augmentierten Bilder werden zum Testdatensatz hinzugefügt. Dabei ist hervorzuheben, dass für diesen Ansatz zwischen den tatsächlichen Fehlerklassen und nicht nur zwischen Anomalie-Gutteil unterschieden wurde. Bei einem Versuch, bei dem nur die binäre Unterscheidung trainiert wurde, konvergierte das Netzwerk nicht. Dieser Ansatz wurde für das selbst trainierte CNN deshalb auch nicht weiter fortgeführt.

Jetzt kann das Convolutional-Neural-Network trainiert werden. Dabei wurden zum Vergleich zwei verschiedene Netzwerke angelegt:

Ein Mini-CNN, das nur aus einer einzigen Convolutional-Schicht, besteht und deshalb nur sehr einfache Muster in den Bildern erkennen kann.

Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

Ein weiteres, tieferes CNN mit mehreren sich abwechselnden Convolutional-Schichten und Max-Pooling Schichten. Dieses ist in der Lage auch komplexere Muster in den Bildern zu erkennen.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Als Optimierer wurde bei beiden Netzwerken Adam mit Categorical Crossentropy als Loss-Funktion und Accuracy als Metrik verwendet. Außerdem wurden Callbacks für das Training verwendet. EarlyStopping sorgt dafür, dass das Training vorzeitig gestoppt wird, falls sich die Validation-Accuracy über 15 Epochen nicht verbessert. ReduceLROnPlateau verringert die Learning-Rate, falls sich die Validation-Accuracy über 10 Epochen nicht verbessert und kann helfen tiefer in Minima der Fehlerfunktion abzusteigen.

Zuletzt werden die beiden Modelle auf dem Testdatensatz angewendet und die Vorhersage für eine der Klassen getroffen.

Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung

## Auswertung

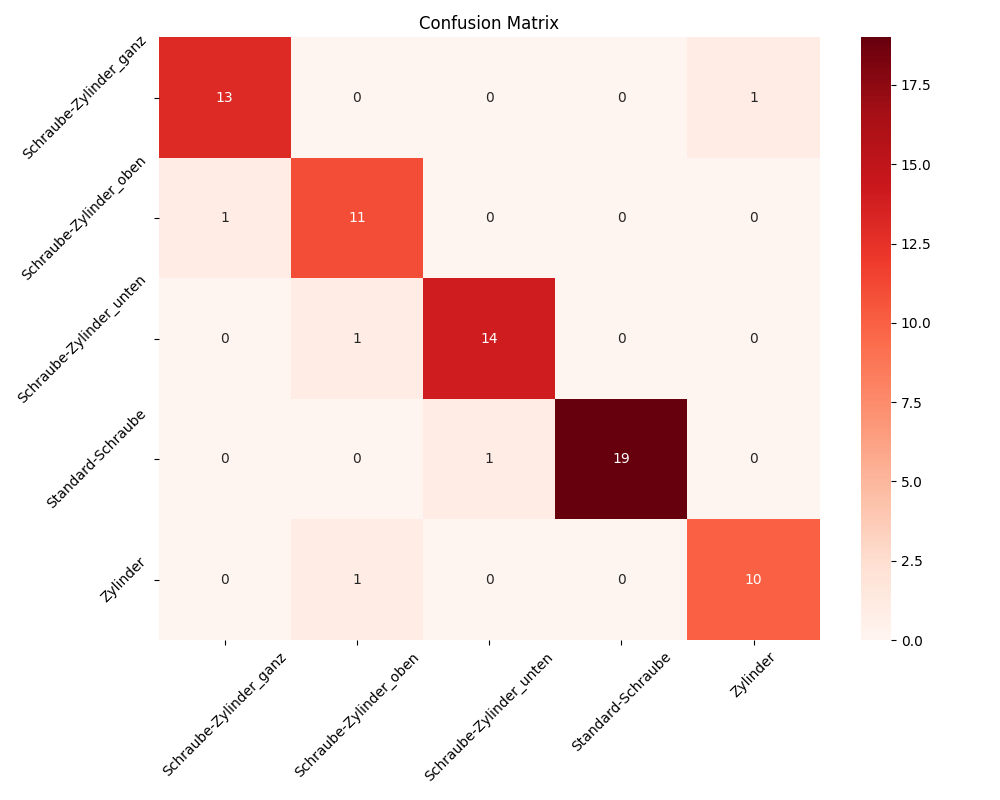
Insgesamt konnten für beide Perspektiven gute Ergebnisse für die Erkennung der Fehlerklassen erreicht werden. Im Rahmen dieses Berichts wird sich erneut auf die seitliche Ansicht beschränkt. Die Ergebnisse der oberen Ansicht können direkt im Gitlab-Projekt angesehen werden.

Ein Bild, das Text, Diagramm, Reihe, Schrift enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Diagramm, Reihe, Schrift enthält.

Automatisch generierte Beschreibung

Insgesamt fällt auf, dass das Training in beiden Fällen sich bereits nach einer sehr kurzen Anzahl an Epochen an die Trainingsdaten angepasst hat. Die Genauigkeit schwankt während des Trainings stark und verbessert sich auch nach deutlich mehr als 30 Epochen nicht. Auffallend ist auch die Entwicklung des Loss auf den Validierungsdaten. Bereits nach ca. drei Epochen steigt der Loss sowohl beim Mini-CNN (stark) als auch beim tieferen CNN (weniger stark), was deutlich auf eine Überanpassung des Netzwerkes (Overfitting) hindeutet die Genauigkeit auf den Test- und Validierungsdaten aber nicht wirklich negativ beeinflusst hat.



Auch die Confusion-Matrix (hier für das tiefere CNN) bestätigt ein gutes Ergebnis. Die auf der y-Achse dargestellten wahren Klassen stimmen bis auf vereinzelte Fehlklassifikationen mit den vorhergesagten Klassen (x-Achse) überein. Positiv ist des Weiteren, dass deutliche Fehlerbilder wie der Zylinder nicht mit der Standard-Schraube verwechselt wurden.

Insgesamt erreicht das Mini-CNN mit nur einer Convolutional-Schicht erreicht eine Genauigkeit von 92% auf die Unterscheidung zwischen den fünf einzelnen Fehlerklassen und eine Genauigkeit von 94% auf die zusammengefasste binäre Entscheidung zwischen Anomalie und Gutteil.

Das tiefere CNN schneidet mit einer Genauigkeit von 93% bei der Unterscheidung der fünf Fehlerklassen und sogar ca. 99% bei der binären Klassifizierung nochmal deutlich besser ab.

## Fazit und mögliche Erweiterungen

TO DO: Bitte noch ergänzen