

Automatisches Transfer-Lernen mittels Autoencodern

Sebastian Hoch

MASTERARBEIT

zur Erlangung des akademischen Grades Master of Science (M.Sc.)

Studiengang Informatik Master

Fakultät Elektrotechnik, Medizintechnik und Informatik
Hochschule für Technik, Wirtschaft und Medien Offenburg

XX.XX.2020

Durchgeführt bei der PSIORI GmbH

Betreuer

Prof. Dr.-Ing. Janis Keuper, Hochschule Offenburg
Dr. rer. nat. Sascha Lange, PSIORI GmbH

Hoch, Sebastian:

Automatisches Transfer-Lernen mittels Autoencodern / Sebastian Hoch. –
MASTERARBEIT, Offenburg: Hochschule für Technik, Wirtschaft und Medien Offenburg,
2020. 21 Seiten.

Hoch, Sebastian:

Automatic transfer learning using autoencoders / Sebastian Hoch. –
MASTER THESIS, Offenburg: Offenburg University, 2020. 21 pages.

Vorwort

Eidesstattliche Erklärung

Hiermit versichere ich eidesstattlich, dass die vorliegende Thesis) von mir selbstständig und ohne unerlaubte fremde Hilfe angefertigt worden ist, insbesondere, dass ich alle Stellen, die wörtlich oder annähernd wörtlich oder dem Gedanken nach aus Veröffentlichungen, unveröffentlichten Unterlagen und Gesprächen entnommen worden sind, als solche an den entsprechenden Stellen innerhalb der Arbeit durch Zitate kenntlich gemacht habe, wobei in den Zitaten jeweils der Umfang der entnommenen Originalzitate kenntlich gemacht wurde. Die Arbeit lag in gleicher oder ähnlicher Fassung noch keiner Prüfungsbehörde vor und wurde bisher nicht veröffentlicht. Ich bin mir bewusst, dass eine falsche Versicherung rechtliche Folgen haben wird.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Hochschule Offenburg öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Offenburg, XX.XX.2020

Sebastian Hoch

Zusammenfassung

Automatisches Transfer-Lernen mittels Autoencodern

 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Abstract

Automatic transfer learning using autoencoders

Englische Version von Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Inhaltsverzeichnis

1. Einleitung	3
1.1. Motivation und Problemstellung	3
1.2. Zielsetzung	3
1.3. Vorgehen	3
2. Grundlagen	7
2.1. TODO GRUNDALGENDETAIL	7
2.2. Einordnung und bestehende Systeme	7
2.3. Datenverständnis	7
2.4. Datenvorbereitung	10
2.5. Bibliotheken und Werkzeuge	10
2.5.1. Psipy	11
2.5.2. Cnvrge.io	13
2.6. Experimentumgebung	13
3. Experimente	17
3.1. Versuchsaufbau	17
3.1.1. Psipy-Modul	17
3.2. Modellierung	17
3.2.1. todo: Greifer	17
3.2.2. todo: Transferlearning	17
3.2.3. todo: Holz	17
3.3. Evaluierung	17
3.3.1. todo: Greifer	17
3.3.2. todo: Holz	17
4. Fazit	21
4.1. Zusammenfassung	21
4.2. Kritische Reflexion	21
4.3. Ausblick	21
Abkürzungsverzeichnis	i
Tabellenverzeichnis	iii

Inhaltsverzeichnis

Abbildungsverzeichnis	v
Quellcodeverzeichnis	vii
A. Ein Anhang	ix
B. Autocrane Daten	xi

Todo list

notwendige klassische Grundlagen definieren	7
Vorgehen auch so beschreiben / ansonsten Kapitel Einleitung anpassen	7
Im Kapitel Bestehdens System erwähnen / diesen Teil in das andere Kapitel verschieben?	8
siehe Bestehendes System	8
Vorgehen + Metriken detaillierter beschreiben	8
Brightness-Histogramm einfügen	9
Pretrain erläutern	10
TF + Keras nur per Ref	10
Bild nur bestehendes System (Stacked Autoencoder + MixedHyperparam*) .	13
Quelle cnvrg	13
mehr Text; was wurde genutzt (Datasets,...)	13

1. Einleitung

Datenrepräsentation sind maßgeblich für die Leistungsfähigkeit von machine learning Algorithmen verantwortlich. Autoencoder sind neuronale Netze welche komprimierte Datenrepräsentationen finden können. todo Auf komprimierten Repräsentationen können deutlich einfacher Ähnlichkeiten gefunden werden und mit deutlich weniger Rechenleistung ML-Algorithmen ausgeführt werden.

1.1. Motivation und Problemstellung

Geschäftsverständnis In der Phase des Geschäftsverständnisses geht es darum, die konkreten Ziele und Anforderungen für das Data Mining festzulegen. Ergebnis dieser Phase ist die Formulierung der Aufgabenstellung und die Beschreibung der geplanten groben Vorgehensweise.

1.2. Zielsetzung

1.3. Vorgehen

Todo list

2. Grundlagen

2.1. TODO GRUNDALGENDETAIL

notwendige klassische
Grundlagen definieren

2.2. Einordnung und bestehende Systeme

Die Bilddaten und Aufgabenstellungen der neuronalen Netzwerke sind in die Problemstellungen des Autocrane-Projekts von PSIORI einzuordnen. Es sind also echte Datensätze und echte Problemstellungen, wobei die gezeigten Aufgabenstellungen und Modelle nicht zwingend in dem Autocrane-Projekt zum Einsatz kommen. Das Autocrane-Projekt ist ein laufendes Projekt, welches das Ziel hat, einen feststehenden Rundlaufkran vollautomatischen zu steuern. In Abbildung 2.1 ist ein Rundlaufkran abgebildet. Dieser Kran wird in einer holzverarbeitenden Anlage zum Befüllen eines Fülltrichters eingesetzt. Dabei sind insbesondere drei Anwendungsfälle interessant. Die Baumstämme werden mittels LKW angeliefert und müssen nach vorgegebenen Regeln (z. B. Ausrichtung, freier Lagerplatz) als Holzstapel gelagert werden. Der Fülltrichter muss mit Holz aus den Holzstapeln oder vom LKW aus befüllt werden. Es ergeben sich also Aufgabenstellungen wie Greifer-Erkennung, Baumstamm-Erkennung, LKW-Erkennung, Strategien für das entladen und aufbewahren der Baumstämme und vieles mehr. [PSIORIGmbH.2019]

Greifer-Erkennung Model erläutern

2.3. Datenverständnis

Anlehnd dem in Kapitel 1.3 beschriebenen Vorgehen werden in diesem Kapitel

Vorgehen auch so
beschreiben / ansor
Kapitel Einleitung
passen



Abbildung 2.1: Rundlaufkran (Foto: ANDRITZ)

die zur Verfügung stehenden Daten und deren Qualität beschrieben. Dabei ist das Kapitel entsprechend der Beschriftung der Daten in zwei Teilbereiche unterteilt.

Im Rahmen des Autocrane-Projektes 2.2 wurde eine Kamera an einem Rundlaufkran angebracht. Die Kamera ist so ausgerichtet, dass sich Aufhängung des Greifers am mittleren oberen Bildrand befindet. Bei einem Rundlaufkran kann die Auslenkung komplett um das Zentrum des Krans bewegt werden. Der Hintergrund der Bilder kann sich stark ändern. Mittels der Kamera werden kontinuierlich neue unbeschriftete Bilder aufgenommen und bei PSIORITY abgelegt. Zu Beginn dieser Arbeit standen mehr 385.000 nicht beschriftete Bilder zur Verfügung. Die Bilder sind 1024 auf 648 Pixel groß und in Farbe. Sie sind in der Form (1024, 648, 3). Die einzelnen Pixel können dabei Werte zwischen 0 und 255 annehmen. Zum Erreichen der Zielstellung werden zwei Datensätze benötigt.

Greifer Datensatz Der Greifer Datensatz enthält Bilder, in welchen der Greifer mittels Rahmen markiert ist. Abbildung 2.3 zeigt ein beispielhaftes Bild mit markiertem Greifer. Abbildung 2.2 zeigt ein beispielhaftes Bild mit markiertem Greifer. Der Datensatz besteht aus zwei Sammlungen von qualitativ unterschiedlich gut beschrifteten Bildern. Die eine Sammlung besteht aus einem bestehenden Datensatz, welcher 4.684 durch Menschen annotierten Bildern enthält. Für den zweiten Teil der Sammlung wurden mittels der bestehenden Objekterkennung XXXX Bilder annotiert. Das genaue Schema der Datenstrukturen kann in Anhang B nachgelesen werden.

titel Bestehends
erwähnen / die-
l in das andere
verschieben?

estehendes Sys-
ten + Metriken
erter beschreiben



Abbildung 2.2: Greifer mit Rahmen

Baumstamm Datensatz Der Baumstamm Datensatz enthält Bilder, welche die Annotation, ob sich Baumstämme im Greifer befinden haben. Abbildung 2.3 zeigt ein Bild, in welchem der Greifer Baumstämme greift. In Abbildung 2.2 befinden sich keine Baumstämme im Greifer. Der Datensatz wurde mittels Crowd

Brightness-Histo
einfügen

Im Rahmen des Datenverständnisses wird versucht, sich einen ersten Überblick über die zur Verfügung stehenden Daten und deren Qualität zu verschaffen. Es erfolgt eine Analyse und Bewertung der Datenqualität. Probleme mit der Qualität der vorhandenen Daten in Bezug auf die in der vorherigen Phase festgelegten Aufgabenstellung sind zu benennen.

todo? Datenqualität auf helle und dunkle Bilder verweisen und somit zu Datenvorbereitung: Skalierung 0-1 verweisen



Abbildung 2.3: Greifer mit Baumstämmen

2.4. Datenvorbereitung

Die Datenvorbereitung dient dazu, einen finalen Datensatz zu erstellen, der die Basis für die nächste Phase der Modellierung bildet.

In dem Schritt Datenvorbereitung werden die Bilder für die Modellerstellung vorbereitet. In dieser Arbeit wurde für diesen Schritt eine Klasse Preprocessing in einem neuen Modul data_preperation.py erstellt. Wie in Listing

zu sehen werden die Pixel der Bilder zwischen 0 und 1 Skaliert. Die Skalierung erfolgt damit jedes Bild eine ähnliche Gewichtung

Neural networks process inputs using small weight values, and inputs with large integer values can disrupt or slow down the learning process. As such it is good practice to normalize the pixel values so that each pixel value has a value between 0 and 1.

erläutern

Die Bilddaten werden

	Train	Test	Validation	Summe
Greifer	X	X	4.684	x
Baumstämme T/F	X	X	X	18000

Tabelle 2.1.: Datenaufteilung - Train Test Validation

eras nur per Ref

2.5. Bibliotheken und Werkzeuge

Tensorflow**Keras****2.5.1. Psipy**

Psipy ist ein Python-Framework für Maschinelles Lernen welches von PSIORI selbst entwickelte Modelle zusammenfasst und eine einheitliche API zu Verfügung stellt. Diese API ist an die API des verbreiteten Frameworks scikit-learn angelehnt und die Modelle aus scikit-learn können in das von PSIORI entwickelte Framework eingebunden werden. Das Framework ermöglicht außerdem das Einbinden von Modellen basierend auf TensorFlow.“[PSIORI]

PSIORI hat im Rahmen des Projektes ein Python-Framework für Maschinelles Lernen entwickelt, das von PSIORI selbst entwickelte Modelle zusammenfasst und eine einheitliche API zu Verfügung stellt. Diese API ist an die API des verbreiteten Frameworks scikit-learn [1] angelehnt und das PSIORI Framework ist mit scikit-learn insofern kompatibel, dass Modelle aus scikit-learn eingebunden werden können. Das Framework ermöglicht außerdem das Einbinden von Modellen basierend auf TensorFlow. [2]

autoencoder.py ConvolutionalAutoencoder StackedAutoencoder

call_picky

hyperparameter_mixin.py [Lindauer.]

Für den effektiven Umgang mit Hyperparametern wurde das Open-Source-Paket ConfigSpace [Lindauer.] eingebunden. Mithilfe dieses Pakets spannen die n Hyperparameter des Modells einen n-dimensionalen Raum auf, aus dem Modell-Konfigurationen gezogen, auf das Model angewandt und bezüglich eines Gütemaßes evaluiert werden können. Die Hyperparameter können dabei kontinuierlich in einem Intervall, kategorisch oder auch konstant sein. Je nach Art des jeweiligen Hyperparameters ist der Raum in der zugehörigen Dimension entweder kontinuierlich oder diskret und hat feste Grenzen. ConfigSpace erlaubt zudem die Anwendung von bestimmten Bedingungen und Regeln bezüglich der Werte, die ein Hyperparameter - gege-

benenfalls in Abhängigkeit von anderen Hyperparametern - annehmen kann. Der Hyperparameter-Raum eines Modells bildet die Grundlage für alle implementierten Optimierungsalgorithmen, die im Folgenden beschrieben werden.

hyperparameter_optimization.py Von den klassischen Suchverfahren zur Hyperparameteroptimierung wurden Algorithmen für Gittersuche und zufällige Suche implementiert, sodass sie insbesondere auf den Fully Connected Autoencoder angewendet werden können. Die gierige Suche wurde bisher nicht implementiert. Für die Gittersuche wird der von den Hyperparametern aufgespannte Raum entlang aller Dimensionen, d.h. Hyperparametern, diskretisiert, so dass ein Gitter von gleichmäßig verteilten Hyperparameter-Kombinationen (Modell-Konfigurationen) entsteht. Die Gittersuche eignet sich vor allem bei einer kleinen Anzahl von Hyperparametern, da der Rechenaufwand mit der Zahl der Hyperparameter exponentiell ansteigt. Oft soll jedoch nur ein Teil der Hyperparameter optimiert werden, wodurch die Gittersuche sich als ein Standardverfahren in der Hyperparametersuche etabliert hat. [6] Bei der zufälligen Suche werden Konfigurationen aus dem Hyperparameter-Raum zufällig gezogen und evaluiert. Die zufällige Suche ist in der Regel effizienter als die Gittersuche [6], garantiert jedoch keine Abdeckung des gesamten Hyperparameter-Raums. Gitter- als auch zufällige Suche wurden für PsiPy als naive Standardverfahren sowie als mögliche Referenz für die Evaluation von neuen Algorithmen implementiert. Die gierige Suche wurde bisher nicht implementiert, da potentielle Anwendungsbereiche bereits von den anderen Suchverfahren hinreichend gedeckt sind.

Das State-of-the-Art-Verfahren zur Hyperparameteroptimierung, “Bayesian Optimization + Hyperband” (BOHB) [7] wurde in das entwickelte Framework eingebunden und kann mit dem Fully Connected Autoencoder genutzt werden. BOHB verbindet die Methoden Hyperband und Bayessche Optimierung. Hyperband erlaubt es, schnell eine vergleichsweise gute Konfiguration zu finden - ein häufiges Ziel, wenn z.B. verschiedene Modelle verglichen werden sollen. Da hier der Aufwand für die Evaluation einer Konfiguration davon abhängt, wie vielversprechend eine Konfiguration zu sein scheint, kann mit BOHB eine wesentlich effizientere Suche durchgeführt werden. Die Methode liegt als Open-Source-Paket [8] vor und wurde daher nicht neu implementiert. Das Paket bietet zudem die Möglichkeit zur Parallelisierung des Optimierungsprozesses, welche für den Fully Connected Autoencoder jedoch bisher nicht implementiert ist.

[6] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (2012): 281-305. URL: <http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf> [7] Falkner, Stefan, Aaron Klein, and Frank Hutter. "BOHB: Robust and efficient hyperparameter optimization at scale." *arXiv preprint arXiv:1807.01774* (2018). URL: <https://ml.informatik.uni-freiburg.de/papers/18-ICML-BOHB.pdf> [8] <https://github.com/automl/HpBandSter>

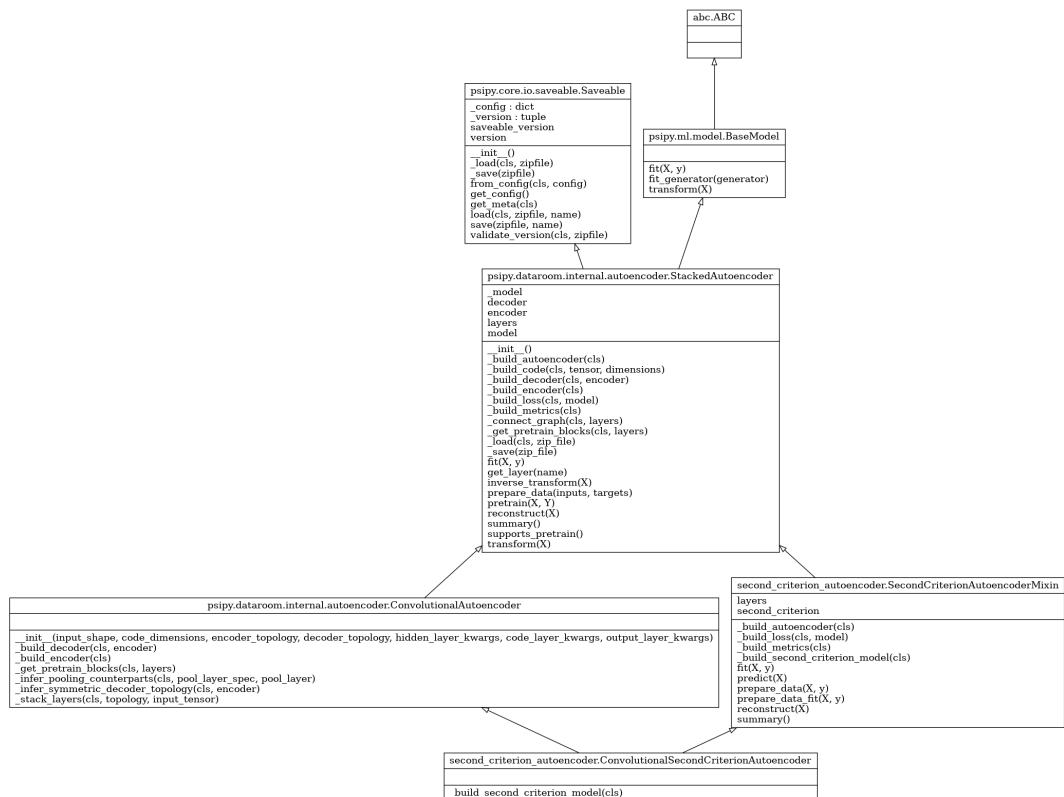


Abbildung 2.4: Klassendiagramm ConvolutionalSecondCriterionAutoencoder

2.5.2. Cnvrg.io

Cnvrge.io ist eine "full-stack Data Science Platform" welche Werkzeuge für die Erstellung, Verwaltung, Bereitstellung und Automatisierung von maschinellem Lernen bereitstellt.

Ouelle cnvrg

mehr Text; was v.
genutzt (Datasets)

2.6. Experimentumgebung

(Hardware + eingesetzte Software)

Todo list

3. Experimente

3.1. Versuchsaufbau

3.1.1. Psipy-Modul

Zweites Kriterium Autoencoder Warum pretrian?: <https://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf> (Warum machen das andere heute nicht mehr? viele cnn haben vanishing gradients problem über relu,... gelösst) (weitere literatur: möglicherweise Geoffrey E. Hinton)

Transfer-Lernen Autoencoder

Auto-Transfer-Lernen Autoencoder

3.2. Modellierung

3.2.1. todo: Greifer

3.2.2. todo: Transferlearning

3.2.3. todo: Holz

3.3. Evaluierung

3.3.1. todo: Greifer

3.3.2. todo: Holz

Todo list

4. Fazit

4.1. Zusammenfassung

4.2. Kritische Reflexion

4.3. Ausblick

insbesondere die möglichen Addons aufführen

Abkürzungsverzeichnis

Tabellenverzeichnis

2.1. Datenaufteilung - Train Test Validation	10
A.1. Tabellenunterschrift	ix

Abbildungsverzeichnis

2.1.	Rund-Kran	8
2.2.	Bsp. Bild: Greifer mit Rahmen	9
2.3.	Bsp. Bild: Greifer mit Baumstämmen	10
2.4.	Klassendiagramm ConvolutionalSecondCriterionAutoencoder	13
A.1.	Beschreibung für Verzeichnis2	x
B.1.	Beschreibung für Verzeichnis2	xii

Listings

B.1. Label	xi
----------------------	----

A. Ein Anhang

Referenz zu Tabelle A.1.

Bezeichnung	Typ	Beschreibung
load.load1	float	The load average over 1 minute.
load.load5	float	The load average over 5 minutes.
load.load15	float	The load average over 15 minutes.
cpu.user	int	The amount of CPU time spent in user space.
cpu.user_p	float	The percentage of CPU time spent in user space. On multi-core systems, you can have percentages that are greater than 100%. For example, if 3 cores are at 60% use, then the cpu.user_p will be 180%.
cpu.system	int	The amount of CPU time spent in kernel space.
cpu.system_p	float	The percentage of CPU time spent in kernel space.
mem.total	int	Total memory.
mem.used	int	Used memory.
mem.free	int	Available memory.
mem.used_p	float	The percentage of used memory.

Tabelle A.1.: Tabellenunterschrift

A. Ein Anhang

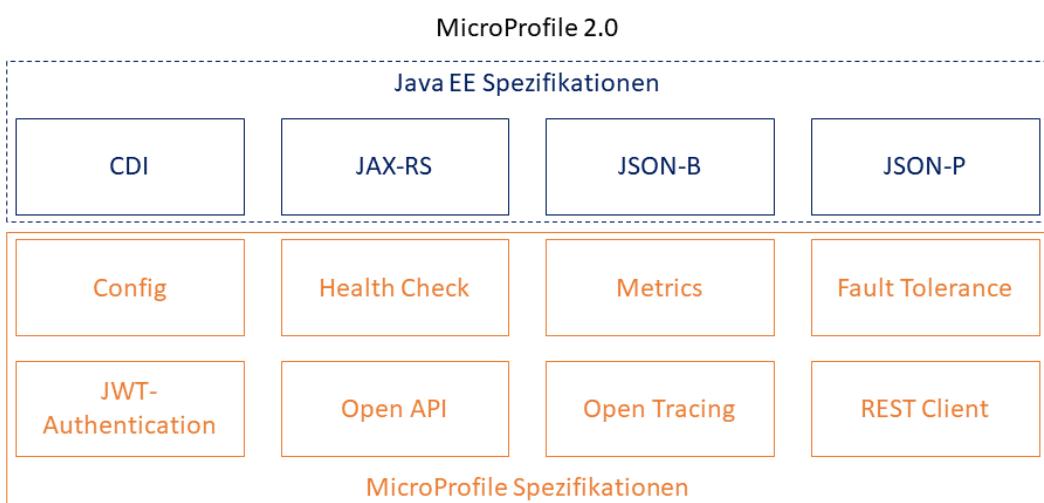


Abbildung A.1: Bildunterschrift2

B. Autocrane Daten

??

```
1 <annotation>
2   <folder>dataset_15_10_2019</folder>
3   <filename>9e26030c-dfbd-4fa7-bd33-5b3a9dcc91ea.png</filename>
4   <path>/Users/jonaskindler/Documents/psiori/second_labels_daniel/
5     dataset_15_10_2019/9e26030c-dfbd-4fa7-bd33-5b3a9dcc91ea.png</path>
6   <source>
7     <database>Unknown</database>
8   </source>
9   <size>
10    <width>648</width>
11    <height>1024</height>
12    <depth>3</depth>
13  </size>
14  <segmented>0</segmented>
15  <object>
16    <name>grapple</name>
17    <pose>Unspecified</pose>
18    <truncated>0</truncated>
19    <difficult>0</difficult>
20    <bndbox>
21      <xmin>256</xmin>
22      <ymin>550</ymin>
23      <xmax>422</xmax>
24      <ymax>679</ymax>
25    </bndbox>
26  </object>
27 </annotation>
```

Listing B.1: Label

B. Autocrane Daten



Abbildung B.1: Bildunterschrift2