

Automatisches Transferlernen 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 Transferlernen 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 Transferlernen mittels Autoencodern

Im Rahmen dieser Arbeit wurden drei Werkzeuge erstellt um Merkmalsextraktion, Transferlernen und AutoML zu kombinieren. Das erste Werkzeug gleicht einen Schwäche eines Autoencoders aus. Beim Training eines Autoencoders wird die Rekonstruktion, also der Output des Modelles und nicht direkt die Einbettung als Bewertungskriterium herangezogen. Um diese Schwäche zu kompensieren wurde der SCAE erstellt. Dieses Werkzeug ist ein Autoencoder mit weiterem Ausgang. Die Datenrepräsentation wird durch ein zweites Kriterium gestärkt. Das zweite Werkzeug nutzt die Datenrepäsentation um einen Transferlearning Task durchzuführen. Das zweite Kriterium wird durch ein neues Kriterium ersetzt. Als drittes Werkzeug wurde der TCSCAE um funktionen des AutoML erweitert. Die besten Hyperparameter werden automatisch gefunden. Die Werkzeuge wurden anhand von echten Datensätzen getestet und validiert. Dabei hat sich gezeigt, dass mit den Werkzeugen eine ähnlich gute Leistung wie auf dem herkömmlichen Weg erreicht werden kann und das durch das Transferlern sogar aufwand reduziert werden kann.

Beschreibugn der Tools weniger konkret?

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. Zielsetzung	3
1.2. Vorgehen	4
2. Grundlagen	7
2.1. TODO GRUNDALGENDETAIL	7
2.2. Bibliotheken und Werkzeuge	7
2.3. Einordnung und bestehende Systeme	9
2.4. Datenverständnis	11
2.5. Datenvorbereitung	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
Abbildungsverzeichnis	v
Quellcodeverzeichnis	vii

Literatur	ix
A. Ein Anhang	xi
B. Autocrane Daten	xiii

Todo list

notwendige klassische Grundlagen definieren	7
Quelle Autoencoder pretrain	9
Quelle fit Keras	9
Vorgehen auch so beschreiben / ansonsten Kapitel Einleitung anpassen	11
Im Kapitel Bestehdens System erwähnen / diesen Teil in das andere Kapitel verschieben?	11
finale Zahl setzen	12
stimmt das am Ende noch?	12
Pretrain erläutern	13
Geheimhaltung: PSIPy-Code nicht zeigen Zeichnungen sind ok, Bilder rücksprache halten, Ansosnten offen.	

1. Einleitung

Das finden geeigneter Repäsentationen von Daten ist ein bekanntes Problem im Feld der DataScience. Dabei ist bekannt, dass Datenrepräsentation maßgeblich für die Leistungsfähigkeit von Maschinellem Lernen verantwortlich ist. Insbesondere Hochdimensionale Daten, wie Bilder, haben mit dem Fluch der Dimensionalität[] zu kämpfen. Der Einsatz von Autoencoder[] erlaubt es komprimierte Datenrepräsentationen für Bilder zu finden. Sind dabei ein unüberwachtes Lernverfahren, brauchen also keine beschrifteten Daten. In vielen Anwendungsfällen ist es teuer oder schwierig beschriftete Daten für neue Anwendungsfälle zu beschaffen. Transferlernen adressiert diese Problematik. Es hat das Ziel, gute Modelle in einer neuen Domäne basierend von Wissen in einer anderen Domäne zu erstellen.

Die PSIORI GmbH [PS20] ist ein Unternehmen, welches Projekte im Bereich der künstlichen Intelligenz für Kunden durchführt. PSIORI kann dabei auf einen mehrjährigen Erfahrungsschatz im Bereich des Maschinellem Lernen zurückgreifen und setzt dabei häufig Autoencoder und das Transferlernen zum Lösen von Aufgabenstellungen ein. Oft ist der Ansatz des Transferlernenes kostengünstiger um ein neues Modell in einer bestehenden Domäne zu erstellen.

(AutoMI auch noch Motivieren)

1.1. Zielsetzung

Ziel dieser Arbeit ist es, Werkzeuge zur Kombination der Ansatzes des Autoencoders und des Transferlernenes zu erstellen und an Hand von einem echten Datensatz zu evaluieren.

1. Einleitung

SecondCriterionAutoencoder Der SecondCriterionAutoencoder (SCAE) ist ein Werkzeug welches beim Erstellen eines Autoencoder mit weiterer Verlustfunktion unterstützt.

TransferSecondCriterionAutoencoder Der TransferSecondCriterionAutoencoder (TSCAE) ersetzt die Verlustfunktion eines SCAE.

AutoTransferSecondCriterionAutoencoder Der AutoTransferSecondCriterionAutoencoder (AutoTSCAE) erweitert den TSCAE um eine automatische Hyperparametersuche.

1.2. Vorgehen

WERKEZUGE erstellen Mnist + Emnist 1. 2.

Modelle Iterativ verbessern 1. fit 2. fitgenerator 3. 80.000 3. AutoML

[MNIST] [EMNIST]

Todo list

34_40 Tonnen 7m 16 pro tag

notwendige klassische
Grundlagen definieren

2. Grundlagen

2.1. TODO GRUNDALGENDETAIL

Stacked Convolutional Auto-Encoders
Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction
Jonathan Masci, Ueli Meier, Dan Ciresan, and Jürgen Schmidhuber
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Lugano, Switzerland jonathan,ueli,dan,juergen@idsia.ch

- layerwise pretrain Greedy Layer-Wise Training of Deep Networks - representation - transferlearning

2.2. Bibliotheken und Werkzeuge

Für den Praktischen Teil der Abschlussarbeit wurde als Entwicklungsumgebung Cnvrge [cn] genutzt. Cnvrge.io ist eine "full-stack Data Science Platform" welche Werkzeuge für die Erstellung, Verwaltung, Bereitstellung und Automatisierung von maschinellem Lernen bereitstellt. Cnvrge erlaubt es Arbeitsbereiche mittels Containern zu erstellen. Die Container können dabei auf Maschinen in Azure[] zugreifen. Für die Experimente wurde ein vorgefertiger Container mit einer tesla-k80 [Nv20], fünf CPUs und 49 GB Arbeitsspeicher genutzt.

Für die Entwicklung wurden insbesondere Python [Py20], Jupyter Notebooks [**ProjectJupyter**] und das Framework Tensorflow [Ma15] genutzt. Die wichtigsten Bibliotheken für die Arbeit sind Keras [Ch15] , Numpy [Ol06] , Matplotlib [Hu07] , scikit-learn [Pe11] , ConfigSpace [**Lindauer.8162019**] , Bayesian Optimization and Hyperband [SAF18] .

Insbesondere für die Visualisierung von Embeddings wurde das Werkzeug PSI ORI Visualizer erweitert und eingesetzt. Der Visualizer erlaubt es Daten in 3D darzustellen, von verschiedenen Blickwinkel und Zoomstufen zu betrachten, zu Filtern und

2. Grundlagen

mit zusätzlichen Informationen zu versehen. Die Abbildung 2.1 zeigt einen Screens-

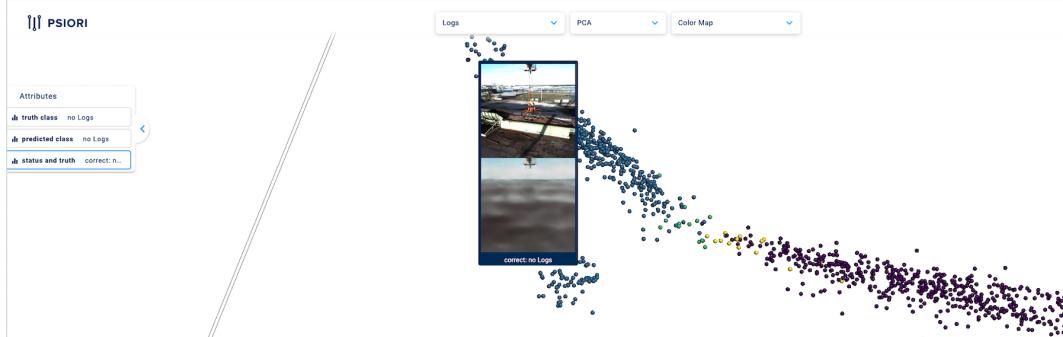


Abbildung 2.1: Beispiel PSIORI Visualizer

hot einer Visualisierung eines Embeddings. Als zusätzliche Information sind die Datenpunkte entsprechend einer Klassifikation in True-Positiv, True-Negativ, False-Negativ und False-Positiv eingefärbt. Über einen Datenpunkt kann mit der Maus geschwebt werden um ein Bild anzuzeigen. Diese Funktion wurde insbesondere zum Anzeigen eines Orginalbildes und ihrer Rekonstruktion mittels Autoencoder genutzt.

Kern des praktischen Teils der Arbeit ist das Framework Psipy [PS19]. 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 Framework eingebunden werden. Das Framework ermöglicht außerdem das Einbinden von Modellen basierend auf TensorFlow.

In den nachfolgenden Abschnitten werden die wichtigsten Module des Frameworks vorgestellt. Diese Module wurden beim erstellen der neuen Module genutzt.

saveable.py Das Modul Saveable bietet Kernfunktionalität zum Speichern und Laden von Klassen. Dabei werden verschiedene Arten von Modellen welche diverse Bibliotheken nutzen können auf eine einheitliche Art und Weise gespeichert.

autoencoder.py Das Modul Autoencoder enthält die drei Klassen StackedAutoencoder, FullyConnectedAutoencoder und ConvolutionalAutoencoder. Der StackedAutoencoder wird als Vaterklasse für die anderen beiden Klassen genutzt. Im Konstruktor werden Methoden aufgerufen welche in den Kindklassen ausprogrammiert sind. Dabei wird ein Keras-Modell für einen Encoder und Decoder entsprechend

von Parametern erstellt. Als weitere wichtige Methoden gibt es die Methode pretrain und fit. Mittels pretrain werden die Schichten eines symmetrischen Autoencoder von aussen nach innen wie in [Greedy Layer-Wise Training of Deep Networks](#) vortrainiert. Die Auswahl der Schichten erfolgt wieder in den Kindklassen. In der fit-Methode wird nach einigen Prüfungen die Methode fit() [fit Keras] des Kerasmodells aufgerufen. In Abbildung 2.2 ist das Klassendiagramm mit den öffentlichen Methoden des ConvolutionalAutoencoder dargestellt.

Quelle Autoencoder
pretrain

Quelle fit Keras

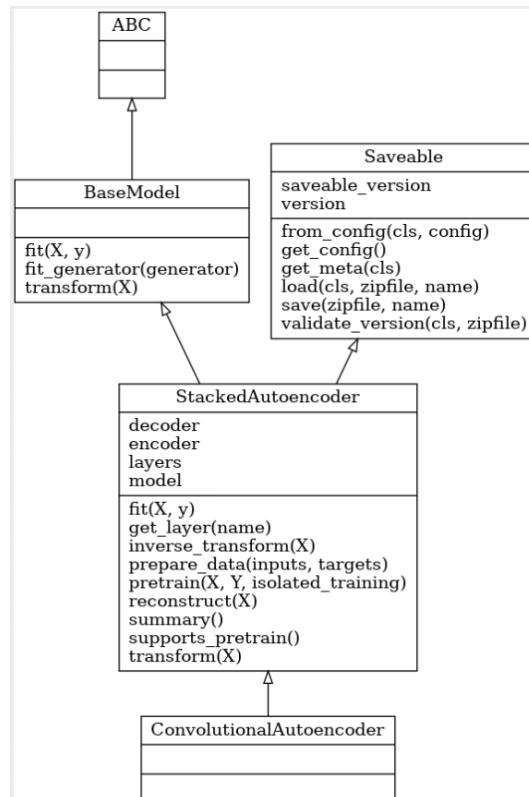


Abbildung 2.2: Klassendiagramm ConvolutionalAutoencoder

hyperparameter_mixin.py Hyperparameter_mixin wird zum Standardisierten ablegen von Hyperparametern für AutoML Klassen genutzt. Auf die Hyperparameter kann anschließend einheitlich zugegriffen werden.

2.3. Einordnung und bestehende Systeme

Die Bilddaten und Aufgabenstellungen der neuronalen Netzwerke sind in die Problemstellungen des Autocrane-Projekts von PSIORI einzurordnen. Es sind also echte

2. Grundlagen

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 2.3 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. [PS20]



Abbildung 2.3: Rundlaufkran (Foto: ANDRITZ)

Greifer-Erkennung Zur Lösung der Aufgaben wird unter anderem ein neuronales Netzwerk zur Positionserkennung des Greifers eingesetzt. Für diese Arbeit werden Vorhersagen des Netzes als Vergleichswert für die Versuche genutzt. Das Netz liegt als frozen_inference_graph.pb vor. Protocoll Buffer [<https://developers.google.com/protocol-buffers/>] ist ein sprachneutraler, plattformneutraler, erweiterbarer Mechanismus zur Serialisierung strukturierter Daten. In diesem Fall enthält die Datei den eingefrorenen Graph und die Model Gewichte. Vorhersagen können mittels einer Tensorflowsession getroffen werden. Dabei liefert das Model, Rahmen in welchem sich der Greifer befindet [<https://leimao.github.io/blog/Save-Load-Inference-From-TF-Frozen-Graph/>]

Baumstamm-Klassifikation Ein Klassifikator, für die Frage ob sich Baumstämme in dem Greifer befinden liegt als meta_graph vor. Die Vorhersage des Modells liefert die vorhergesagte Klasse mit einer Wahrscheinlichkeit zurück.

2.4. Datenverständnis

Anlehnend dem in Kapitel 1.2 beschriebenen Vorgehen werden in diesem Kapitel 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 2.3 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 PSIORI 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.

Vorgehen auch
schreiben / anson
Kapitel Einleitung
passen

Im Kapitel Beste
System erwähne
sen Teil in das a
Kapitel verschie

Greifer Datensatz Der Greifer Datensatz enthält Bilder, in welchen der Greifer mittels Rahmen markiert ist. Abbildung 2.5 zeigt ein beispielhaftes Bild mit markiertem Greifer. Abbildung 2.4 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 14.018 Bilder annotiert.

Baumstamm Datensatz Der Baumstamm Datensatz enthält Bilder, welche die Annotation, ob sich Baumstämme im Greifer befinden haben. Abbildung 2.5 zeigt ein Bild, in welchem der Greifer Baumstämme greift. In Abbildung 2.4 befinden sich keine Baumstämme im Greifer. Der Datensatz wurde im Zusammenarbeit mit quality-match[<https://www.quality-match.com/imprint>] und Crowdworkern erstellt.



Abbildung 2.4: Greifer mit Rahmen

Weiterer Datensatz Im Laufe der Arbeit wurden in Zusammenarbeit mit quality-match ein weiterer Datensatz erstellt. Dieser Datensatz enthält 80.000 beschriftete Bilder. Es wurden sowohl die Beschriftung "Logs ja / nein" als auch die Beschriftung Rahmen des Greifers erstellt. Zusätzlich wurden weitere Beschriftungen wie Helligkeit, Winkel des Greifers, ... erstellt. Diese weiteren Beschriftungen wurden in der Arbeit nicht genutzt.

zahl setzen

das am Ende



Abbildung 2.5: Greifer mit Baumstämmen

2.5. 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_preparation.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.

Die Bilddaten werden

Pretrain erläutern

Greifer

Baumstämme T/F

logs 5671 not loaded 6542 train 9.771 1 4.537 nl 5.233 80 test 1.224 1 568 nl 655 10 val 1.225 1 568 nl

Tabelle 2.1.: Datenaufteilung - Train Test Validation

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	13
A.1. Tabellenunterschrift	xi

Abbildungsverzeichnis

2.1. Beispiel PSIORI Visualizer	8
2.2. Klassendiagramm ConvolutionalAutoencoder	9
2.3. Rund-Kran	10
2.4. Bsp. Bild: Greifer mit Rahmen	12
2.5. Bsp. Bild: Greifer mit Baumstämmen	13
A.1. Beschreibung für Verzeichnis2	xii
B.1. Beschreibung für Verzeichnis2	xiv

Listings

B.1. Label	xiii
----------------------	------

Literatur

- [Ch15] Chollet, F. et al.: Keras, 2015.
- [cn] cnvrg.io: cnvrg.io, URL: <https://cnvrg.io/>.
- [Hu07] Hunter: Matplotlib: A 2D graphics environment, 2007.
- [Ma15] Martin Abadi; Ashish Agarwal; Paul Barham; Eugene Brevdo; Zhifeng Chen; Craig Citro; Corrado, G. S.; Andy Davis; Jeffrey Dean; Matthieu Devin; Sanjay Ghemawat; Ian Goodfellow; Andrew Harp; Geoffrey Irving; Michael Isard; Jia, Y.; Rafal Jozefowicz; Lukasz Kaiser; Manjunath Kudlur; Josh Levenberg; Dandelion Mané; Rajat Monga; Sherry Moore; Derek Murray; Chris Olah; Mike Schuster; Jonathon Shlens; Benoit Steiner; Ilya Sutskever; Kunal Talwar; Paul Tucker; Vincent Vanhoucke; Vijay Vasudevan; Fernanda Viégas; Oriol Vinyals; Pete Warden; Martin Wattenberg; Martin Wicke; Yuan Yu; Xiaoqiang Zheng: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015, URL: <https://www.tensorflow.org/>.
- [Nv20] Nvidia: Tesla K80, 2020, URL: <https://www.nvidia.com/de-de/data-center/tesla-k80/>.
- [Ol06] Oliphant, T.: NumPy: A guide to NumPy, hrsg. von Trelgol Publishing USA, 2006, URL: <http://www.numpy.org/>.
- [Pe11] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12/, S. 2825–2830, 2011.
- [PS19] PSIORI GmbH: psipy documentation, hrsg. von PSIORI GmbH, 2019, URL: <https://psipy.azurewebsites.net/source/psipy.html>.
- [PS20] PSIORI GmbH: PSIORI, 2020, URL: <https://www.psiori.com/de>.

- [Py20] Python Software Foundation: The Python Language Reference, 2020,
URL: <https://docs.python.org/3.7/reference/>.
- [SAF18] Stefan Falkner; Aaron Klein; Frank Hutter: BOHB: Robust and
Efficient Hyperparameter Optimization at Scale. arXiv:1807.01774/,
2018, URL:
<https://ml.informatik.uni-freiburg.de/papers/18-ICML-BOHB.pdf>.

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

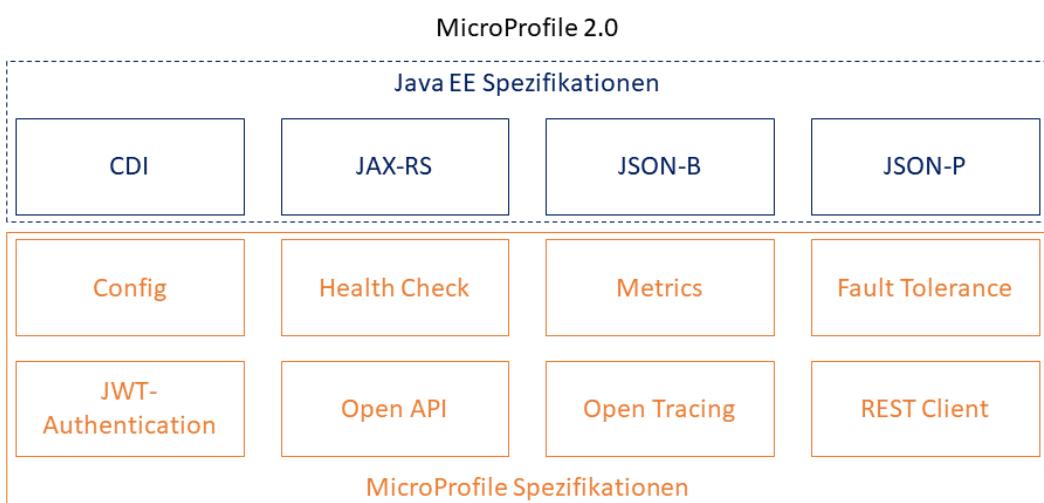


Abbildung A.1: Bildunterschrift2

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