



FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

**Application of Deep Learning for
Inspection in Industrial Overhaul
Processes**

Benjamin Taheri, M.Sc.



FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Forschungs- und Lehrinheit 1
Angewandte Softwaretechnik

Application of Deep Learning for Inspection in Industrial Overhaul Processes

Benjamin Taheri, M.Sc.

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr. Matthias Althoff

Prüfer der Dissertation: 1. Prof. Dr. Bernd Brügge
2. Prof. Dr.-Ing. Birgit Vogel-Heuser

Die Dissertation wurde am 05.03.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 08.07.2020 angenommen.

Abstract

Industrial machineries wear down by nature after certain years of operation, which could lead to critical break-down and a costly repair. However, a regular overhauling process can stop the system from wearing down and malfunctioning. Overhauling consists of disassembling, inspecting and reassembling the components to ensure that each part is in serviceable condition. The main workflow of the overhaul process can be generalized to removing, disassembly, cleaning, inspection, replace/repair, reassembly, installing and testing.

Inspection is the core of the overhauling process, in which all components must be inspected for damages and anomalies, and repaired or replaced if needed. The components consist of parts of various sizes such as rotors and covers, as well as small parts and fasteners such as bolts, screws, washers, and nuts that hold different parts together. Therefore, the inspection involves identifying damages, sorting and classifying the fasteners, and placing them into compartments for further reuse, which are performed manually in overhaul plants by technicians. The technicians are more likely to be exposed to a broad range of occupational hazards, noise, vibration and different kinds of radiation, and chemicals. Some inspection tasks can be tedious and time-consuming, which increases the errors, especially under time pressure. Moreover, the inspection tasks have their specific challenges: 1) the number and the similarity of the fasteners and small parts that must be manually sorted, classified and packaged, 2) characteristics and nature of metallic parts and surfaces, and 3) different range of damages which need to be inspected. These challenges introduce opportunities to use machine learning, computer vision and automation for inspection tasks in overhaul processes.

To assist technicians in overhauling processes, we conducted an experimental research on developing supporting systems using deep learning and computer vision, including damage detection, sorting, classification, and packaging of the fasteners used in the machineries. We used a deep learning based approach to sort, classify and place the components inside compartments. Deep learning based methods are characteristically data-centric algorithms and their performance depends on the availability of a dataset of images. We collected 7 experimental different datasets for fastener damage detection, surface damage detection, sorting, single-view classification, multi-view classification, classification with mixed real and synthetic images, and bin picking. Having an accuracy of over 99% in damage detection and classification, over 99% in sorting, over 99% in classification and over 70% in bin picking, our approaches imply a possible usage in overhaul processes. Using these applications reduces the the time and human error of the overhauling process.

Zusammenfassung

Industriemaschinen nutzen sich nach bestimmten Betriebsjahren von Natur aus ab, was zu kritischen Ausfällen und kostspieligen Reparaturen führen kann. Regelmäßige Wartungsprozesse können jedoch Abnutzungen reduzieren und dadurch auch Ausfallzeiten minimieren. Der Überholungsprozess besteht in der Regel aus der Demontage, Inspektion und Montage der Komponenten, um sicherzustellen, dass sich jedes Teil in einem betriebsbereiten Zustand befindet. Der Hauptarbeitsablauf des Überholungsprozesses kann auf das Entfernen, Zerlegen, Reinigen, Prüfen, Ersetzen / Reparieren, Zusammenbauen, Installieren und Testen verallgemeinert werden.

Die Inspektion ist der Kern des Überholungsprozesses, bei dem alle Komponenten auf Beschädigungen und Anomalien überprüft und bei Bedarf repariert oder ersetzt werden müssen. Die Komponenten bestehen aus Teilen verschiedener Größen wie Rotoren und Abdeckungen sowie kleinen Teilen und Befestigungselementen wie Bolzen, Schrauben, Unterlegscheiben und Muttern, die verschiedene Teile zusammenhalten. Daher umfasst die Inspektion das Erkennen von Schäden, das Sortieren und Klassifizieren von Befestigungselementen sowie dessen Einordnung in verschiedene Behälter zur weiteren Wiederverwendung. Diese Tätigkeiten werden heutzutage meist noch manuell von Technikern in Wartungszentren durchgeführt. Techniker kommen hierdurch leichter mit einer Vielzahl von Gefahren am Arbeitsplatz in Berührung, wie z.B. Lärm, Vibrationen, Chemikalien und verschiedene Arten von Strahlung. Einige Inspektionsaufgaben können langwierig und zeitaufwändig sein, was die Fehlerraten von Instandhaltungen insbesondere unter Zeitdruck erhöhen kann. Darüber hinaus haben Inspektionsaufgaben ihre jeweiligen spezifischen Herausforderungen: 1) Anzahl und Ähnlichkeit der Verbindungselemente und Kleinteile, die manuell sortiert, klassifiziert und verpackt werden müssen, 2) Eigenschaften und Art der Metallteile und -oberflächen und 3) unterschiedliche Bereiche von Schäden, die überprüft werden müssen. Diese Herausforderungen bieten die Möglichkeit, maschinelles Lernen, Computer Vision und Automatisierung für Inspektionsaufgaben in Wartungsprozessen einzusetzen. Um Techniker bei der Durchführung von Wartungsprozessen zu unterstützen, wurden Experimente zur Entwicklung unterstützender Systeme unter Verwendung von Deep Learning und Computer Vision durchgeführt.

Diese Experimente beinhalteten die Schadenserkennung, Sortierung, Klassifizierung und Verpackung der in den Maschinen zu verwendeten Befestigungselemente. Unter Verwendung eines Deep Learning-basierten Ansatzes wurden Komponenten sortiert, klassifiziert und verschiedenen Behältern zugeordnet. Deep Learning-basierte Ansätze sind charakteristisch datenzentrierte Algorithmen und ihre Leistung hängt von der Verfügbarkeit eines Datensatzes von Bildern ab. Für unsere Experimente wurden Datensätze von den folgenden 7 Aufgabenbereichen generiert: Die Erkennung von Befestigungsschäden, die Erkennung von Oberflächenschäden, die Sortierung von Einzelteilen, die Einzelansichtsklassifizierung, die Mehrfachansichtsklassifizierung, die Klassifizierung mit gemischten realen und synthetischen Bildern sowie für die Kommissionierung von Behältern. Als Resultate konnten die folgenden Genauigkeiten erzielt werden: Schadenserkennung und -klassifizierung (über 99%), bei der Sortierung und Klassifizierung von Einzelteilen (über 99%), bei der Kommissionierung in Behälter (über 70%). Diese Ergebnisse implizieren, dass sich unsere Ansätze gezielt bei anstehenden Wartungsprozessen einsetzen lassen. Die Verwendung unserer Ansätze reduziert darüber hinaus die Wartungsdauer, sowie das menschliche Versagen während des Wartungsprozesses.

Acknowledgments

I would like first to express my deepest sincere gratitude to my advisor professor Bernd Brügge, for his trust and support, and the freedom he gave me during the years of my doctoral studies. I never forget the first time I talked to him after his lecture and the willingness and kindness he showed to help. I would also like to thank professor Birgit Vogel-Heuser, for accepting to be my second supervisor. I first met her at CASE 2019 in Vancouver and we had a wonderful discussion during the lunch with leaders, in which she implicitly showed her interest to be my supervisor.

I am also thankful to all my colleagues and friends at the chair of applied software engineering. In particular, I would like to thank Jan Knobloch for being a nice office-mate to me, Juan Haladjian for the discussions we had and all the comments he gave me on my papers and research, and Zardosht Hodaie for being there whenever I needed to chat and for his input and comments on my approaches, Rana Alkadhi for all her encouragements, and Jan Ole Johanßen and Nadine von Frankenberg und Ludwigsdorff for their help in reviewing parts of my dissertation. I also thank Monika Markl for managing and organizing all the bureaucratic processes, Helma Schneider for resolving whatever technical issue I had, and Uta Weber for helping with the financial aspects of my studies and conference travels.

During the years of my doctoral studies, I had the opportunity to work with motivated and talented students. I would like to thank Amr Abdelraouf, Ralf Schönfeld, René Birkeland, Valon Xhafa, Paul Rangger, Haonan Yu, Leo Vinzenz, and Marcel Schmitt. Specifically, I would like to thank Ralf Schönfeld and Haonan Yu for their help, their working attitude, and the discussions with them that enriched my research.

My research would not have been possible without the support with hardware and mechanical setups. I would like to thank Mr. Michael Neumaier from NeuPro Solutions for supporting me in my research. I am also grateful to Mr. Christoph Haas and Mr. Karsten Bunde at MTU Maintenance (Hannover) for their kind invitation to visit the company and their comments on my first paper manuscript.

Last but not least, I thank my wife, family and friends. I thank Hannah for all her patience and love. Without her support I would have not been able to make it through the doctoral studies. I also thank my parents for their support before and during my doctoral studies.

Contents

abstract	i
Zusammenfassung	i
Acknowledgments	iv
1 Introduction	1
1.1 Hypotheses	12
1.2 Rating and Scope	13
1.3 Outline	15
2 Background	17
2.1 Basics of Computer Vision	17
2.1.1 Contour Detection	17
2.1.2 Image Moment	18
2.1.3 Dimensionality Reduction	18
2.2 Basics of Deep Neural Networks	20
2.2.1 Artificial Neurons	20
2.2.2 Multilayer Perceptron	21
2.2.3 Activation Functions	22
2.2.4 Loss Functions	24
2.2.5 Layers	24
2.2.6 Training a Network	26
2.3 Convolutional Neural Networks	31
2.3.1 Classification	32
2.3.2 Multi-view Convolutional Neural Networks	34
2.3.3 Rotation Net	37
2.3.4 Siamese Networks	38
2.3.5 Object Detection	38
2.3.6 Object Segmentation	39
2.4 Unsupervised Techniques for Anomaly Detection	41

2.4.1	Autoencoder	41
2.4.2	One-class Support Vector Machine	42
2.4.3	Isolation Forest	43
2.4.4	Local Outlier Factor	44
2.5	Evaluation Metrics	44
3	PPIC: Platform for Parts Image Capturing	46
3.1	Industrial Parts Characteristics and Challenges	46
3.2	PPIC	48
3.2.1	PPIC Configurations	49
4	Damage Identification	56
4.1	Damage Detection of Fasteners	57
4.1.1	Dataset Collection	58
4.1.2	Methods	60
4.1.3	Results and Discussion	60
4.2	Surface Damage Detection	67
4.2.1	Data Collection	68
4.2.2	Methods	69
4.2.3	Evaluation and Results	71
4.2.4	Discussion	71
5	Sorting the Fasteners based on their Similarity	76
5.1	Data Collection	77
5.2	Training and Evaluation	77
5.3	Results and Discussion	80
6	Fine-grained Visual Categorization of Fasteners	84
6.1	Single-view Classification	84
6.1.1	Dataset Creation	85
6.1.2	Training and Evaluation	86
6.1.3	Results and Discussion	90
6.2	Multi-view Classification	91
6.2.1	Dataset Creation	92
6.2.2	Training and Evaluation	94
6.2.3	Results and Discussion	96
6.3	Using Synthetic Data for Classification	98
6.3.1	Dataset Creation	100
6.3.2	Training	101
6.3.3	Results and Discussion	104

7	Bin Picking	106
7.1	Bin Picking on Uniform Background	108
7.1.1	Data Collection	108
7.1.2	Method	109
7.1.3	Evaluation	112
7.1.4	Results and Discussion	112
7.2	Bin Picking on Non-uniform Background	113
7.2.1	Dataset Creation	113
7.2.2	Model Training	113
7.2.3	Method	115
7.2.4	Evaluation	118
7.2.5	Results and Discussion	119
8	Validation	122
8.1	SPARCS	122
8.1.1	Scenarios	124
8.1.2	SPARCS Platform	127
8.1.3	Results	132
8.2	SUMA	134
8.2.1	Scenarios	134
8.2.2	SUMA Platform	136
8.2.3	Evaluation and Results	139
8.3	Benchmark Preparation	139
8.4	Comparison and Discussions	141
8.5	Threats To Validity	143
9	Contributions and Future Work	146
9.1	Contributions	146
9.2	Future Work	148
	Appendix A Acronyms	153
	Appendix B Copyrights	155
	Appendix C Datasets	160
	List of Figures	169
	List of Tables	172
	Bibliography	173

Chapter 1

Introduction

Industrial machineries are made of mechanical components that use power to apply forces to perform an intended action, such as gearbox, carburetor, and valve. These mechanical components are used widely in different areas, such as agriculture, aviation, assembly lines, packaging and labeling. Some areas have defined the worthiness conditions of the machines which refers to their ability to meet the standards and be in a suitable operating condition. For instance, airworthiness of an engine in aviation industry, roadworthiness of a car in automobile industry, and railworthiness of a locomotive in railway industry. There are multiple approaches to ensure the worthiness of a machine and system reliability, such as increasing system capacity, reinforcing redundancy and employing more reliable components [Endrenyi et al., 2001]. However, these approaches can be heavily constrained by the machine specifications, such as size, and cost. Therefore, performing maintenance checks is becoming the most practical approach to ensure the machinery's worthiness and extend its lifespan [Mobley, 2002].

Maintenance programs range from the very simple to the quite complex. European standard EN 13306¹ formulated maintenance as a "combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function". This formulation implies a difference between corrective, and preventive maintenance processes. Corrective maintenance is performed after the occurrence of a failure to fix the machine. It can be carried out immediately after the failure occurs (remedial) or delayed to limit the interference of the production process (deferred) [Moubray, 2001]. It is the simplest maintenance program that can adopt a rigid maintenance schedule where predefined activities are carried out at fixed time intervals [Endrenyi et al., 2001]. Whenever the component fails, it must be repaired or replaced (run-to-failure). There has been a shift since the

¹<https://standards.cen.eu/dyn/www/f?p=204:110:0>

60s to preventive time-driven maintenance programs to ensure higher availability of the machine and lower costs (planned) [Mobley, 2002]. Planned preventive maintenance is performed at predetermined intervals to reduce the probability of failure. Improvement maintenance is aimed at reducing the need of maintenance by making changes in the process or by engineering solutions. Maintenance must be carried out in a shutdown condition, when the whole machine is stopped. Since early 2000s, adapting new technologies and using them in maintenance processes resulted in reducing the manual tasks, and therefore, reducing the costs and time for the maintenance. These new technologies enable monitoring the machines and perform the maintenance whenever there is an early signal for any malfunction. Condition-based maintenance can be performed in normal working conditions, when the system is running. However, depending on the circumstances, shutdown conditions are considered safer than maintenance during working production.

There are also other maintenance methods, such as total productive maintenance (TPM), reliability centered maintenance (RCM), or improvement maintenance. TPM is a Japanese approach that aims to address planned maintenance shortcomings with some best business practices [Ahuja and Khamba, 2008]. RCM can determine which maintenance method is the most appropriate for the specific component in machine [Moubray, 2001]. Improvement maintenance aims to reduce the need of maintenance or enhance the machine with new features by making changes in the process or by engineering solutions. This dissertation uses the aforementioned categorization, shown in Figure 1.1, which is adapted from [Mobley, 2002]. Figure 1.2 illustrates maintenance evolution through the last 70 years.

Industrial overhauling is a maintenance that consists of disassembling, inspecting and reassembling the components to ensure the worthiness condition [Lampe et al., 2004]. Overhauling is carried out for variety of reasons, such as a regular maintenance check or changes to the machine to introduce new features.

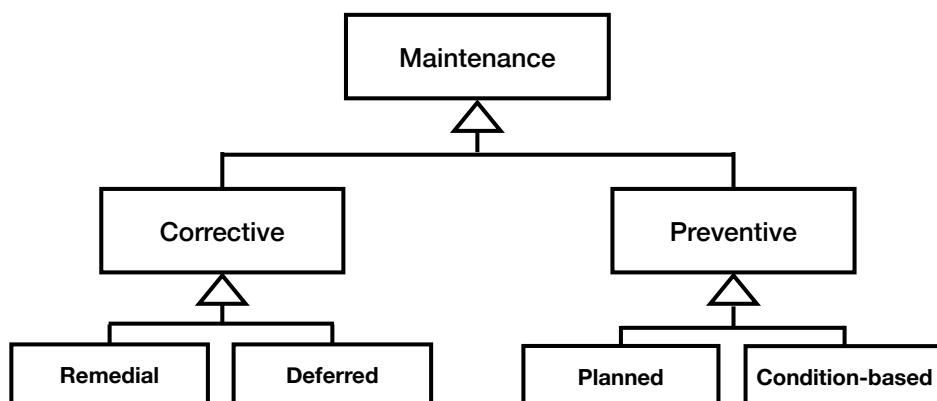


Figure 1.1: Overview of maintenance types.

Corrective: <ul style="list-style-type: none"> • If it ain't broke, don't fix it • High overtime labor cost • High machine down time • High cost 	Preventive (Planned): <ul style="list-style-type: none"> • Time-driven (mean-time-to-failure) • Total shut down • Unnecessary repairs • Catastrophic failure 	Preventive (Predictive): <ul style="list-style-type: none"> • Condition-driven (regular monitoring) • During running mode • Minimize unscheduled breakdowns
1960	2000	Now

Figure 1.2: Evolution of maintenance through the years.

Due to the fact that different industries have different criteria for their machines worthiness and different engineering processes, there is no single standard process for overhaul. Nevertheless, we can generalize how the main workflow of the overhaul process is conducted. Overhauling in planned preventive maintenance begins with component removal, following with disassembly, inspection, repair and replace, reassembly, installation and testing. Overhauling in condition-based preventive maintenance begins with inspection (regular monitoring the machine) and follows with the same activities, but only for the specific machine component, for which the monitoring data show an early signal of failure or malfunction. The main focus of this dissertation is on the machines with high safety requirements, such as airplane engines. The condition-based preventive maintenance is performed on most of these machines, using different sensors to monitor the performance. However, due to higher safety requirements, these machines are required for a total overhaul after every specified amount of usage. In Figures 1.3, and 1.4, different activities in corrective, planned and condition-based overhauling of industrial machines' components are shown.

- **Remove:** while it is possible to perform certain repairs on a mounted components, it is needed to disconnect and remove the entire component for a general overhaul. The steps in preparation for removal consists of disconnecting wires, draining the fuel and oil, remove any auxiliary drive and detaching any throttle. After the preparation, the component must be free of all its attachments except for its mounting. Removing some fasteners allows the component to be lifted and moved out.
- **Disassemble:** this phase consists of disassembling the accessories, such as tanks, flippers, etc., and removing different sub-components, such as carburetor, muffler, valves, cylinders, crankshaft, etc. These sub-components might vary depending on different type of main component.
- **Inspect:** at this stage, each sub-component must be cleaned and inspected

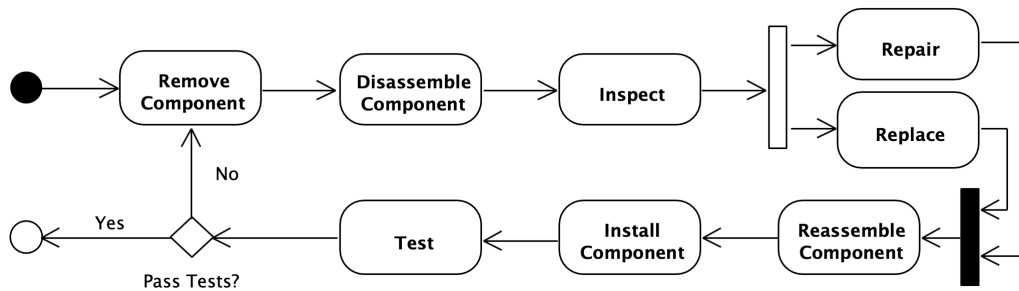


Figure 1.3: Activities in overhauling of industrial machines for both corrective and planned preventive maintenance (UML activity diagram).

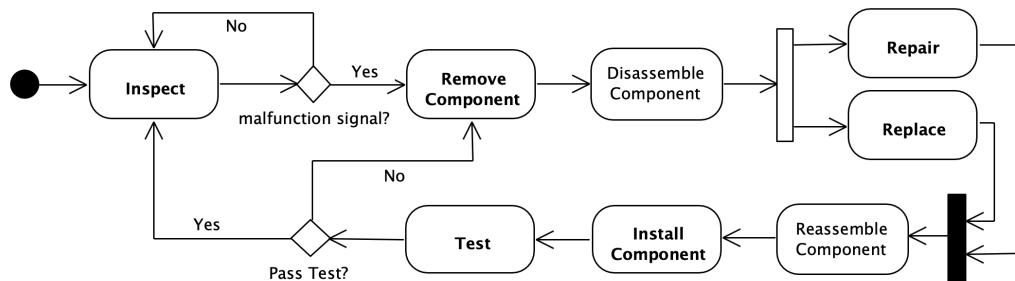


Figure 1.4: Activities in overhauling of industrial machines for condition-based preventive maintenance (UML activity diagram).

properly. All component parts must be compared with the permissible limits and assessed for their reusability or reparability. Each part is observed relative to the legal requirements and the manufacturing specifications.

- **Repair/Replace:** depending on the inspection results, some parts must undergo a replacing or repairing process.
- **Reassemble:** once the required inspection and Repair/Replace tasks are performed, the pile of parts and sub-components must be assemble back into a working component.
- **Install:** after reassembly of the component, it must be installed back into the machine. All parts which were detached in the removal step, must be attached again to the component.
- **Test:** in this step, an integration test must be carried. Therefore, the component is tested for any abnormalities, such as overspeeds, unusual knocking or banging, excessive smoke from exhaust overheating, oil or gas leakage. If the component fails the test, overhauling must continue with removing the

component and disassembling it; this time, the component is not repaired and it must be replaced.

The inspection is the core of the overhaul process. In this phase, all the disassembled machine components must be inspected for possible damages and prepared for the next step (namely Reassembly). This preparation includes sorting and classifying the components, and packaging them into compartments. These components consist of parts of various sizes such as panels from the wing or a propeller, as well as fasteners such as screws, bolts, washers, and nuts that hold different parts together. Figure 1.5 shows a model of machine component and Figure 1.6 presents the workflow of inspection activities in overhaul processes.

As inspection is carried out, technicians are more likely to be exposed to a broad range of occupational hazards. Data from the Spanish working conditions survey² indicate a high exposure of inspection technicians to noise, vibration and different kinds of radiation, and chemicals. Some inspection tasks involve routine and tedious processes in abnormal operating conditions requiring the use of equipment. Exposure to the hazards and abnormal working conditions results in increasing the errors, especially under time pressure and when high safety expectations are required. For example, technicians may decide to replace a fastener of an airplane engine, only because of a simple damage. However, it may turn out that the fastener is in normal condition and can be reused. These sort of false-negative decisions increases the cost of inspection and overhaul in general.

Overhauling inspection technicians often work alongside a running process and in close contact with machines. As overhaul is an activity in which technicians

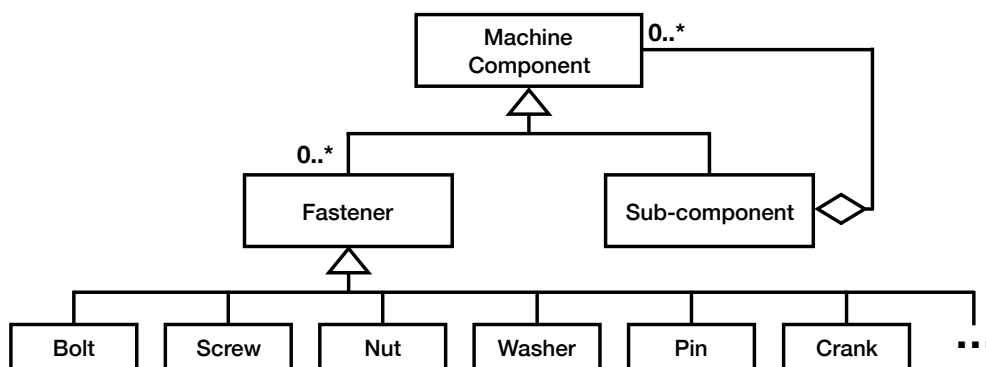


Figure 1.5: Composition of machine component and taxonomy of fasteners.

²<https://www.observatoriovascosobreacoso.com/wp-content/uploads/2015/12/VI-ENCUESTA-NACIONAL-CONDICIONES-TRABAJO-INSHT.compressed.pdf>

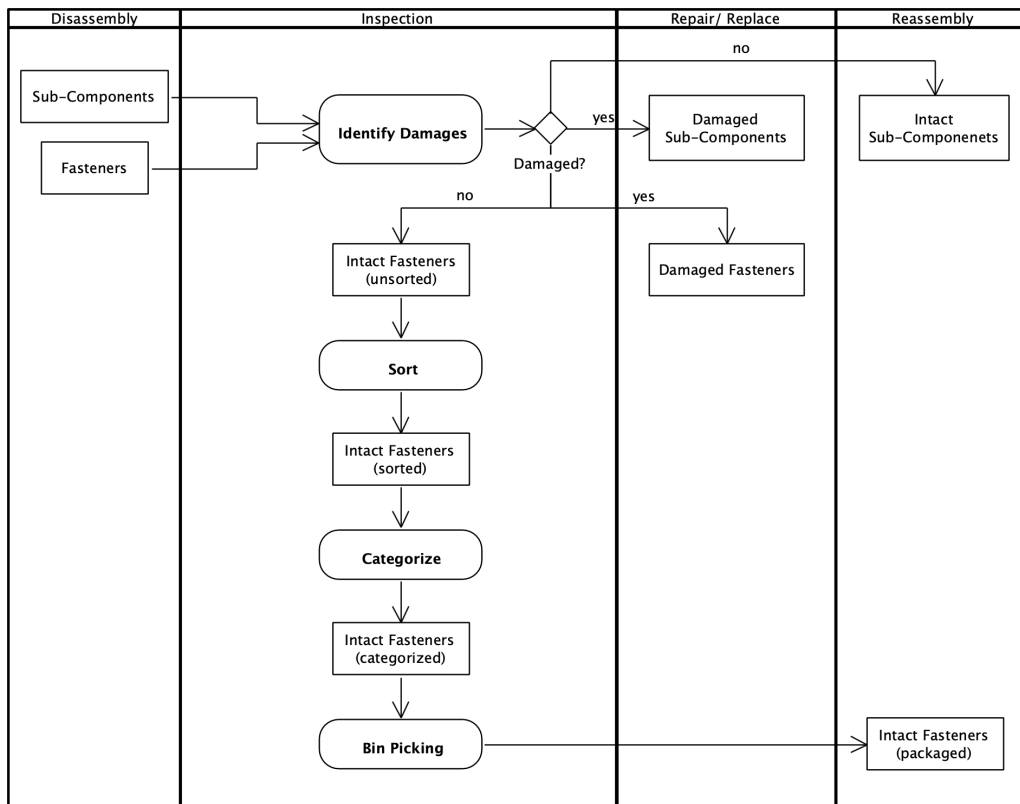


Figure 1.6: Simplified activities with focus on Inspection in overhaul processes (UML activity diagram).

need to be in close contact with processes (in contrast to production modes), direct contact between technicians and machines cannot always be reduced substantially. Therefore, inspection processes are yet depending on manual workforce. This dissertation studies each of the inspection sub-processes in industrial overhauling and defines the typical challenges in each. The following examples present the possible challenges during different inspection processes in overhauling:

- Identify Damages:** after disassembling all fasteners from a component and washing off dirt and oil, they are inspected for anomalies and damages. Therefore, they are placed in a container and brought to special workstations in maintenance checking plants with access to magnifiers and illumination with which the technicians inspect them for all kind of damages, including oxidation, scratches, missing or flaked off coating and erosion. Figure 1.7 shows examples of the damages on two sample bolts. There are no comprehensive defined rules to identify fasteners as damaged. Although non-destructive methods like fluorescent penetrant testing or magnetic par-

ticle testing are used to identify cracks and voids in bigger components³, applying them for all the fasteners could be expensive and unrealistic, considering their smaller size. In addition, in some industries, such as aero-engines, most of the fasteners are made of resistant alloys, like steel or titanium, and are manufactured under specific considerations [Teer and Salem, 1977], they can be expensive and technicians must check each of them by looking and touching, in which they may decide to keep or discard them. Typically, the technicians might need up to 15 seconds to inspect a fastener properly in their workstations. Furthermore, the nature of aero-engine fasteners introduces problems like reflection and shadow [Taheritanjani et al., 2019a] which makes the task of damage detection challenging. Therefore, an automatic damage detection process would save time and reduce costs during overhaul processes.



Figure 1.7: Examples of intact and damaged bolts from a side view (a) and top view (b). The annotated damages show either scratches or dirt which could not be cleaned by washing [Taheritanjani et al., 2019b].

³<https://power.mtu.de/engineering-and-manufacturing/aero-solutions/special-processes/>

- **Sort:** the technicians are provided with a box full of fasteners that are removed from a specific component, and must categorize them (Figure 1.8). Categorizing the fasteners one by one is time consuming. Therefore, the technicians start with sorting them based on their similarity. Sorting can be in different stages: first start with placing the different type of fasteners in different groups (bolts, nuts, washers, etc.), then each group to multiple groups based on the size of the fasteners. This process continues until all the fasteners are sorted based on their similarity. Typically, the technicians know the number of different fasteners in the component (n). Therefore, their task is to sort the fasteners in the box, in n groups. The similarity of the fasteners makes sorting still time consuming and error prone. Figure 1.8 shows an example of the sorted fasteners on the workspace.



Figure 1.8: Examples of industrial parts on a workstation that are grouped based on their similarity.

- **Categorize:** one challenge during manual inspection in overhaul processes is number of fasteners and small parts that must be classified and placed in their respective containers [Jardine and Tsang, 2005] [Marx and Graeber, 1994]. The disassembled fasteners in overhaul plants, as shown in Figure 1.9, are brought in compartments to workstations with access to catalogs, digital manuals, pictures of each part, magnifiers, and measurement devices with which the technicians can classify them [Taheritanjani et al., 2019a].



Figure 1.9: Fasteners that were taken from part of engine and must be classified during inspection in overhaul process.

After grouping the fasteners based on their similarity, the technicians pick at least one fastener from each group and classify it with its part number, using the magnifiers. Typically, bolts, screws and nuts have a part number carved on them, and can be classified using their part number. However, some fasteners do not have any part number, such as washers or cranks. In addition, on some fasteners the part number might not be recognizable due to scratches or erosion. In such cases, the technicians must compare fasteners with their comparison-sketch in a catalog in an 1:1 scale relative to the fastener structure (see Figure 1.10) to classify them correctly and place them in the correct container. The technicians may also compare the fasteners with the fastener samples in the catalog. Most of the aero-engine fasteners are made of resistant alloys, like steel or titanium, and are manufactured under specific considerations [Teer and Salem, 1977]. Therefore, they can be expensive and the process of 1:1 scale comparison must be repeated for all fasteners without part number or with unrecognizable part number, which makes the classification time consuming and costly [Marx and Graeber, 1994]. As an example, to classify a bolt with unrecognizable part number, the technicians might need up to 30 seconds to find possible similar sketches in the catalog and compare them [Taheritanjani et al., 2019a].

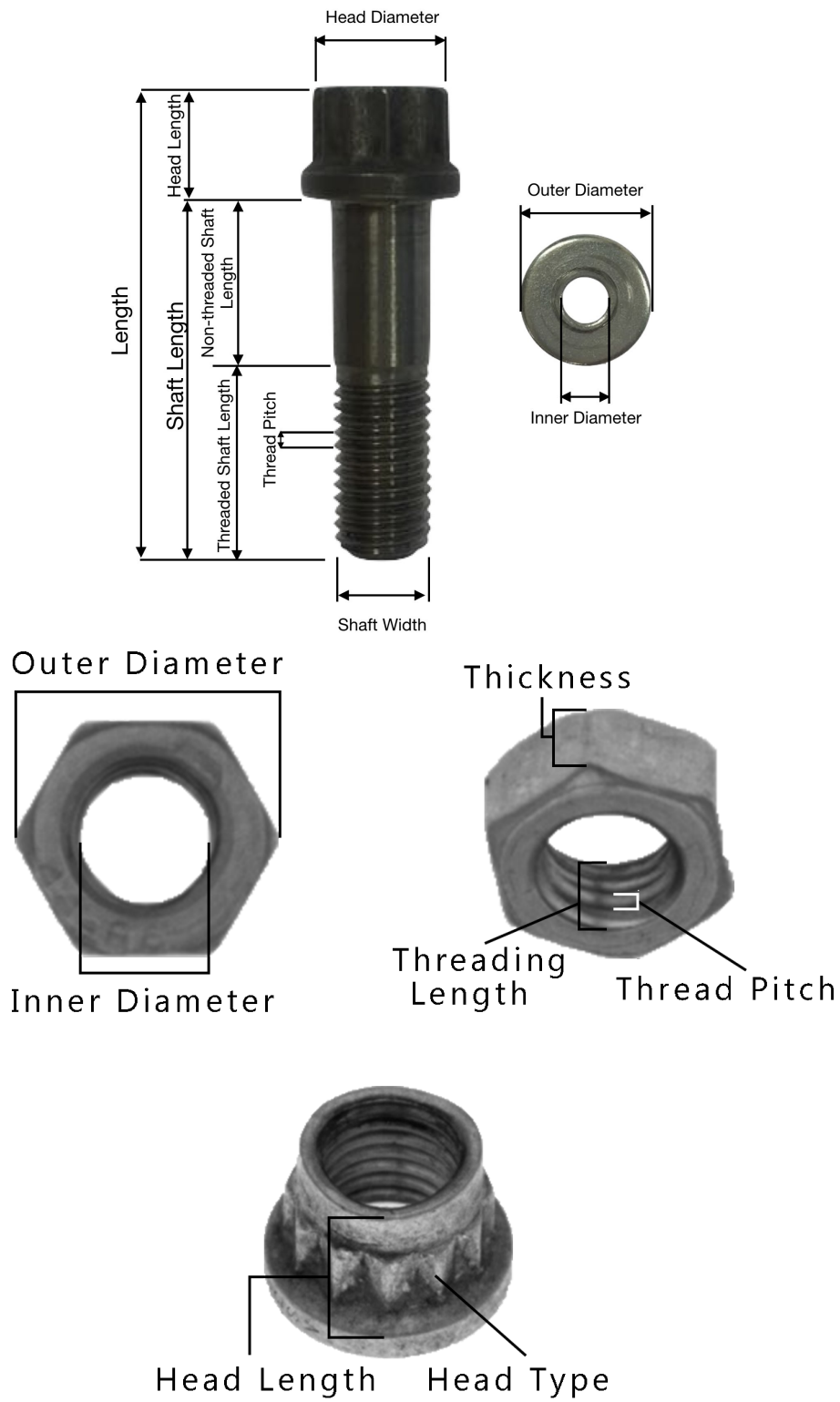


Figure 1.10: The structure of a sample bolt, washer and nut, displaying the length, width, shaft, pitch and diameters ([Taheritanjani et al., 2019a] and [Birkeland, 2018]).

- **Bin Picking:** after sorting and categorization, and prior to refitting, fasteners must be placed into different containers, which are sent to assembly stations for refitting. Figure 1.11 shows examples of compartments, where sorted and inspected parts must be placed into them.



Figure 1.11: Examples of compartments which contain the small parts after packaging.

In large overhauling plants, placing small parts and fasteners into different containers is performed manually by a human or technician. These technicians are provided with guide sheets that define the type and number of parts that must be placed into different containers. Occasionally, some containers must contain different types of parts, e.g., bolts and corresponding washers. Typically, the technicians must check the parts' number, count and pick the required amount, and place them into a designated container. Typically, the position of the part inside its container is unimportant; however, some parts must be placed in a specific position relative to their orientation inside a container.

Filling all containers is tedious and time consuming: checking one part's number, counting, picking and placing them into a container for every part as specified in the guide sheet. One challenge is to memorize the location of each part on the workstation. Due to the numbers and different types of small parts to package, their similarities, and mixed and different orders on the guide sheet, technicians must frequently double check part numbers to ensure they select the correct parts. Therefore, a picking process to automatically place parts into containers could reduce time and costs during packaging in overhaul processes.

1.1 Hypotheses

This dissertation investigates three hypotheses to address the aforementioned challenges during inspection in overhaul processes:

- **H₁: the damages on the fasteners and components' surfaces can be identified by training a model on intact fasteners, i.e. without knowledge of damages.**

Although most of the applications of Machine Learning today are based on super-vised learning, the vast majority of the available data is actually unlabeled. There are images of fasteners and surfaces, but we do not have their status (labels as damaged or intact). Without the knowledge of damages, detecting anomalies on surfaces and fasteners can be performed using unsupervised learning methods. The distinction between damaged and undamaged samples is often delicate and a notion of normality must be determined from undamaged instances.

- **H₂: an automatic inspection process, including damage detection, sorting, classification and bin picking, saves time in overhaul processes.**

This dissertation concentrates on inspection sub-process in industrial overhaul and investigate a set of deep learning based approaches to reduce the costly false-negatives. Sorting, categorizing and bin picking require various measurements. For example, sorting and categorizing involve measuring the length or counting threads of a fastener, while bin picking requires computing the orientation and grasp point of parts to be picked by a robotic arm. Identifying damages requires observations to ensure the component is in a worthy condition. These measurements and observations can be automated using vast variety of methods, such as laser, fluorescent, ultrasonic, scale, RFID, NFC, and camera.

- **H₃: the error (in particular false-negatives) in manual inspection tasks in overhaul processes is reduced with machine learning and computer vision.**

Some methods, such as scaling the components for classification, cannot be employed alone for automatic inspection. They must be used along with combination of other sensors, which can make the design of automatic inspection systems more complex. Non-destructive methods, such as fluorescent penetrant testing or magnetic particle testing, have been employed for damage inspection, particularly to identify cracks and voids in bigger

components⁴. However, applying them for all smart components and fasteners could be expensive and unrealistic, considering their smaller size. In addition, some modern technologies like smart dusts⁵ are not practical, and to the best of our knowledge, they have not been used yet for building the mechanical machines. On the other hand, computer vision approaches using deep learning methods achieved state of the art results in image classification [Chan et al., 2015] [Wang et al., 2017] [Akata et al., 2015], image segmentation [Girshick, 2015] [Ren et al., 2015] [He et al., 2017], and anomaly detection [Schlegl et al., 2017] [Cha et al., 2017], which can be employed for automatic inspection tasks in overhaul processes.

Parts of this dissertation have been previously published [Taheritanjani et al., 2019a] [Taheritanjani et al., 2019b] [Taheritanjani et al., 2020].

1.2 Rating and Scope

To compare the functional and non-functional requirements used in the studies in this dissertation with other studies described in the literature, we use the Table 1.1. Each requirement is listed as fulfilled (+), partially fulfilled (\pm) or not fulfilled (-).

⁴<https://power.mtu.de/engineering-and-manufacturing/aero-solutions/special-processes/>

⁵Smart dust refers to collection of tiny wireless microelectromechanical sensors, which are able to detect conditions such as light, vibration, temperature and noise, and autonomously communicate this information back to a receiver.

Table 1.1: Rating schema used to compare the results and state of the art in this dissertation.

Requirement	Explanation
R ₁ : Support for Industrial Small Parts and Fasteners	+ : used small parts and fasteners in the study ± : can be used for industrial small parts - : is not applicable to be used for industrial small parts
R ₂ : Support for Real-time	+ : method can be performed in less than 1 second ± : method requires less than 3 seconds - : method requires more than 3 seconds
R ₃ : Explainability and Interpretability (using Visualizations)	+ : yes ± : third party solutions can be used with the study - : no
R ₄ : Generalizability	+ : generalizable to variety of different settings ± : partially generalizable with a set of constraints - : only applicable using the stated constraints
R ₅ : Need for Labeled Data	+ : no need for labeled data ± : less than 20% labeled data are needed - : more than 20% labeled data are needed
R ₆ : Accuracy on test datasets for a multi-class classification	+ : above 95% accuracy ± : above 80% accuracy - : less than 80% accuracy

The scope of this dissertation is limited as follows:

1. **Context.** This dissertation focuses on the inspection process of components. Components consist of fasteners and other sub-components (Figure 1.5). The inspection process is the damage identification for both sub-components and fasteners, and sorting, categorizing and bin picking for fasteners.
2. **Damage Type.** Various components can expose different type of damages. For example, the components with electrical circuits and wires may have semiconductor failures after certain amount of time. This dissertation focuses on physical damages such as scratches, oxidation, dents and missing coats.
3. **Camera-based Inspection.** This dissertation uses camera images to apply computer vision and deep learning methods for automatic inspection process. These methods might not be applicable to other sensor data.

1.3 Outline

The dissertation is structured as follows:

Chapter 2 provides background information on deep neural networks, including foundations on image classification, image segmentation, and anomaly detection. In addition, the background information on basic computer vision techniques, such as contour detection and dimensionality reduction, is presented.

Chapter 3 explains the characteristics and challenges of capturing images from industrial fasteners and surfaces. Moreover, it describes *PPIC*, the image capturing platform that we use in the dissertation, to capture images for damage detection, sorting, categorization and bin picking.

Chapter 4 focuses on the validation of H_1 using deep learning methods, in particular supervised learning using Mask R-CNN [He et al., 2017] and unsupervised learning using autoencoders [Schlegl et al., 2017] to identify unknown damages.

Chapter 5 describes a similarity detection approach using Siamese Networks for sorting the fasteners based on their similarity [Bertinetto et al., 2016]. The result of the sorting will be grouping the similar small parts together, using only an image of the mixed parts. This sorting helps to speed up the process, before we categorize the parts in a fine-grained manner. Furthermore, if all the parts can be sorted, they can be used directly for packaging, without any fine-grained classification.

Chapter 6 illustrates convolutional neural network based approaches for automatic classification the small parts in a fine-grained level using one or multiple views [Chan et al., 2015]. As the result, the serial number of each part used earlier during labeling is returned.

Chapter 7 describes different image segmentation methods, such as pixel thresholding and Mask R-CNN, for automatic bin picking of the parts in different containers for further reuse. After segmentation, small parts orientation and grasp point are calculated, using only a 2D image of the sorted parts on the surface. We used principal component analysis (PCA) and Image Moment to find the orientation and grasp point of fasteners to have a baseline for comparisons [Jolliffe, 2011] [Karakasis et al., 2015].

Chapter 8 describes two industrial demonstrators (*SPARCS* and *SUMA*) to validate the Hypotheses H_2 and H_3 . *SPARCS* focuses on identification of small parts in the area of jet-engine maintenance for large aircraft. It uses the automatic categorization and automatic bin picking approaches described in Chapters 6 and 7, to classify the jet-engine fasteners and place them in containers. Using metrics such as the execution time of the categorization and bin picking tasks, the *SPARCS* demonstrator shows that the automatization saves time when compared to manual categorization and manual bin picking. In addition, using precision and recall

metrics, we show the error ratio in *SPARCS* is reduced. *SUMA* investigates problems in the area of surface maintenance. It uses the automatic damage detection methods described in Chapter 4 to identify scratches and dents on boat surfaces. This is followed by automatic grinding and polishing the identified damages with a robotic arm. Using precision, recall, IoU (intersection of units) and execution time for damage identification, we show that the automization increases the accuracy of the damage identification and reduces overall maintenance time when compared with manual maintenance.

Chapter 9 summarizes the contributions of this work, and identifies future work directions.

Appendices include the list of acronyms, the copyrights to use the published articles, and the links to access the datasets.

Chapter 2

Background

This chapter presents the basic concepts of computer vision methods used in this dissertation. In addition, it provides the foundations and principles of neural networks and deep learning, and explains how neural networks can be used for image recognition, similarity detection, image segmentation, and anomaly detection. Section 2.1 describes the basic computer vision methods, such as contour detection and dimensionality reduction. Section 2.2 describes the fundamentals of deep neural networks and how to train them. Section 2.3 explains the convolutional neural networks (CNNs) and different CNN-based approaches, such as multi-view classification, siamese networks, and rotation net, used in this dissertation. Section 2.4 describes unsupervised (recently referred as self-supervised) approaches, such as autoencoders used in anomaly detection. Finally, Section 2.5 presents the evaluation metrics that are used in different studies in this dissertation.

2.1 Basics of Computer Vision

This section presents an overview of basic computer vision methods that are used in different chapters of this work.

2.1.1 Contour Detection

Contour detection detects the borders of the subjects shapes in an image. There are a number of operations that are used to identify contours, namely computing bounding polygons, approximating shapes, and generally calculating regions of interest. Calculating the regions of interest simplifies interaction with image data due to working with a rectangular region, which is easily defined with an array slice, using fast computing libraries, such as NumPy.

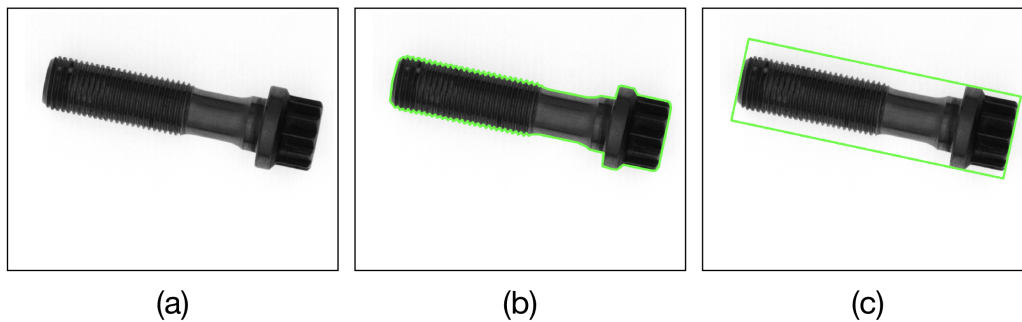


Figure 2.1: Original Image (a). The result of contour detection (b). Minimum enclosing rectangle around the contour (c).

Bounding Box and Minimum Enclosing Rectangle

In many cases, such as object detection applications, finding the contours of an object can be an overkill. In some applications, we are interested in determining the bounding box of the subject or its minimum enclosing rectangle (or circle).

To identify the coordinates of the minimum enclosing rectangle, we calculate the minimum rectangle area, and compute the vertexes of the rectangle. Since the calculated vertexes are floats and pixels are accessed with integers, a conversion is required. Figure 2.1 shows an example of contour detection and minimum enclosing rectangle computation.

Using Minimum enclosing area, we performed preprocessing steps in Chapters 4 and 7.

2.1.2 Image Moment

In computer vision, an image moment is a function with a certain particular weighted average (moment) of the image pixels' intensities, with an attractive property or interpretation. Image moments are useful to describe objects after segmentation. Image moments capture information about the shape of a blob in a binary image, such as area (or total intensity), centroid, and information about its orientation. Figure 2.2 shows an example image and its first image moment (the centroid).

2.1.3 Dimensionality Reduction

In machine learning approaches, we try to find a set of features among the features in the input instances (feature engineering) and train a solution on them. However, many problems involve thousands of features, which makes training

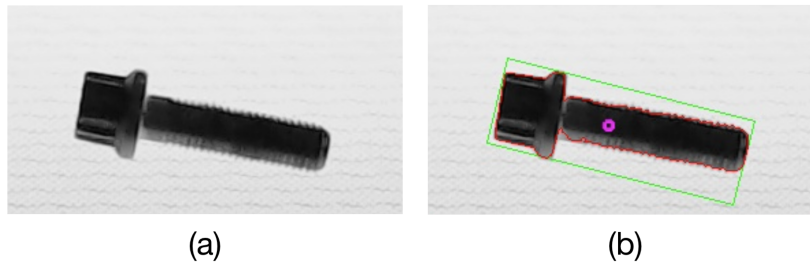


Figure 2.2: Original Image (a) and its first image moment as the centroid in pink color (b). The mass on the bolt head is more than its shaft. Therefore the centroid is not exactly on the center and is towards the head.

extremely slow and can make it harder to find an optimal solution. This problem is often referred to as the curse of dimensionality.

In some problems, it is often possible to reduce the number of features. For example, if the images of the fasteners are center aligned, the pixels on the image borders belong almost always to the background, so we could completely drop these pixels from the training set without losing information. In addition, in a high resolution image, two neighboring pixels are often highly correlated. Therefore, we can merge them into a single pixel by taking the mean of the two pixel intensities [Géron, 2019].

Principal Component Analysis

Principal Component Analysis (PCA) is the most popular dimensionality reduction algorithm [Géron, 2019]. First it identifies the hyperplane that lies closest to the data (in our case, pixel values plotted in 2D plane), and then it projects the data onto it.

Prior to projecting the training set onto a lower-dimensional hyperplane, the right hyperplane must be selected, i.e. the axis that preserves the maximum amount of variance, as it will lose less information than the other projections. Another way to justify this choice is that it is the axis that minimizes the mean squared distance between the original dataset and its projection onto that axis. This is the rather simple idea behind PCA.

PCA can also be used in finding the orientation of an object in an image. By taking the axis that minimizes the mean squared distance between the pixel intensities in the input image, we can compute the orientation of the object. Figure 2.3 shows an example of applying PCA and finding the orientation of the object in the image.

Using PCA, we obtained baselines for comparison purposes in Chapter 7.

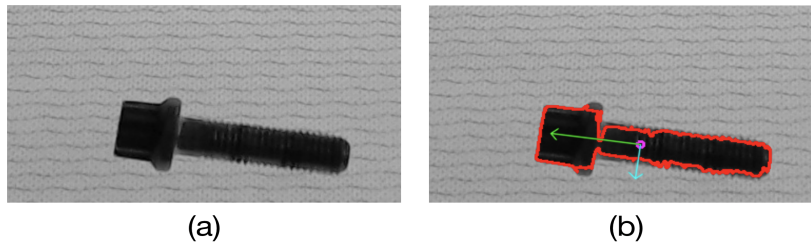


Figure 2.3: Applying PCA to an input image and finding the axes. The bigger axis represents the orientation of the bolt.

2.2 Basics of Deep Neural Networks

Deep learning is a machine learning method that is based on neural networks with multiple layers of artificial neurons. Section 2.2.1 introduces artificial neurons which are the essential parts of neural networks. Section 2.2.2 deals with the basic structure of neural networks. The Sections 2.2.3, 2.2.4, and 2.2.5 focus on different types of layers, which are used to group artificial neurons with the same functionality. Finally, Section 2.2.6 describes in detail how a neural network actually learns.

2.2.1 Artificial Neurons

The human brain is able to interpret sensor data at an immense speed and can learn internal representations without explicit instructions [Hinton, 1992]. The human neuron collects information signals from other neurons to which it is connected with fine structures called dendrites. Spikes of electrical activity are sent out through a long structure (the axon), which splits into branches. At the end of each branch a synapse converts the activity into an electrical effect stimulating connected neurons.

An artificial neuron models the way a neuron in the brain processes information. Artificial neurons modeling digital representations of the neurons are combined into neural networks to achieve brain-like properties. An artificial neuron has one or many inputs (x) with assigned weights (w), as shown in Figure 2.4. These weights model the connection properties of the dendrites and result in a weighted input for the neuron. The bias b in Figure 2.4 is a constant term modifying the output ($output = (inputs * weights) + bias$) allowing scaling/transformation of the activation function. This output is passed to one or multiple outputs (y) being the input of other neurons. For a neural network to know how to perform a specific task the weights modeling the interconnection (how can one neuron influence others) have to be adapted.

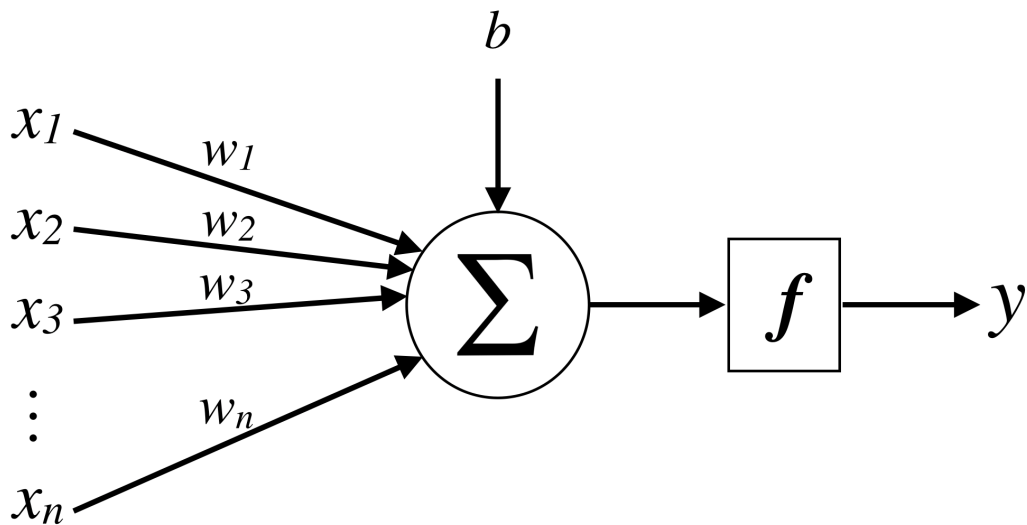


Figure 2.4: Artificial neuron.

2.2.2 Multilayer Perceptron

A Multilayer Perceptron (MLP), also called feedforward neural network is the basis of current deep learning models [Goodfellow et al., 2016]. The goal of a neural network is to approximate an arbitrary function f^* . The neural network is a function matching a specific input to a specific output. A classifier for example takes an input x and has the target to map it to an output y representing the class of the input. The graph in Figure 2.5 shows a perceptron which can for example be used for a binary classification task. The perceptron consists of a network graph with one hidden layer, i input nodes, j hidden nodes and one output node. Each node has an edge connecting it with the previous and following layer.

θ is a set of parameters of the neural network. A mapping from an input to an output in a feedforward neural network is defined as a function $y = f(x, \theta)$ (using the input x and θ to compute the output y). The parameters are learned to result in the best function approximation. The nodes in the graph are artificial neurons introduced in section 2.2.1, also the parameters of the perceptron θ are the weights w of the neuron. Neural networks are typically structured in layers containing the artificial neurons. Usually a layer only has connections to the preceding and the following layer - like the hidden layer in Figure 2.5. A layer is "hidden" when it is not directly connected to the input or output. As the artificial neuron applies a function to inputs and generates outputs, a network of neurons is a nesting of these functions. According to Hornik et al., a feedforward can approximate a Borel measurable [Azoff, 1974] (continuous) function from one finite dimensional space to another, providing enough hidden units (neurons in a hidden layer) [Hornik

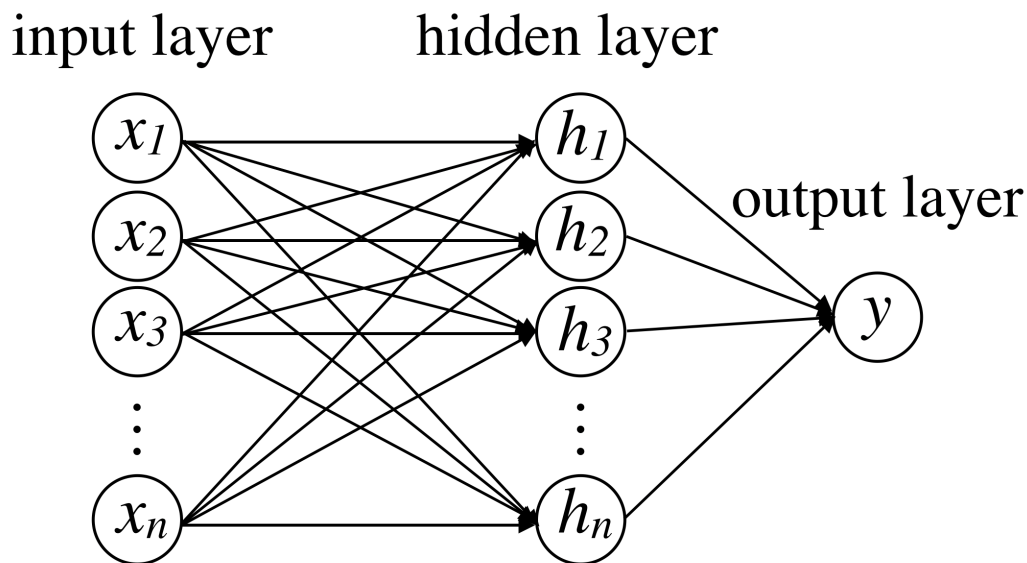


Figure 2.5: An example of a multilayer perceptron.

et al., 1989].

2.2.3 Activation Functions

Artificial neural inputs are summed up and passed to a function which is either linear, threshold-based or nonlinear. Activation functions are nonlinear functions $g(z)$ and are applied element-wise to hidden units (see section 2.2.2). They allow the approximation of complex functions by providing a slope [Géron, 2019]. The sigmoid activation function [Goodfellow et al., 2016] ranges from zero to one, as shown in Figure 2.6. Sigmoid activation can be used for binary classification. The hyperbolic tangent activation function, as shown in Figure 2.6, is a closely related to the sigmoid activation function because $\tanh(z) = 2\sigma(2z) - 1$. Both have the issue of fast saturation (reaching the value one and not further providing a slope for gradient descent). The output stays always the same for large or small values of z . This is one of the reasons why rectified linear units (*ReLU*s) and variations are used in modern neural networks. Figures 2.8 and 2.9 depict rectified linear units where the *ReLU* is $g(z) = \max\{0, z\}$ and the *LeakyReLU* is $g(z) = \max\{0.1z, z\}$. *ReLU*s also have the benefit of simple and therefore fast computation. The standard *ReLU* doesn't saturate for values > 0 and therefore the gradients remain large for active units resulting in faster convergence.

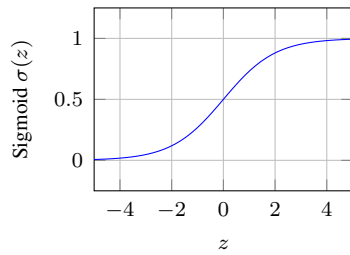


Figure 2.6: Sigmoid activation function.

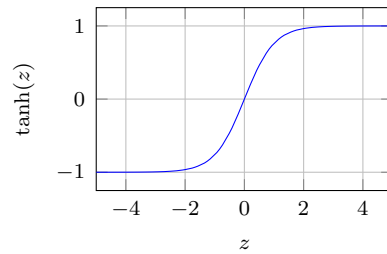


Figure 2.7: Tanh activation function.

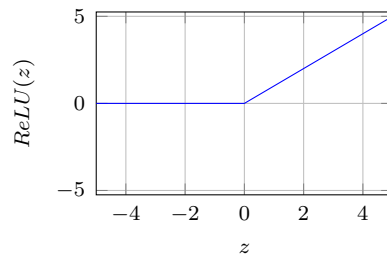


Figure 2.8: ReLU activation function.

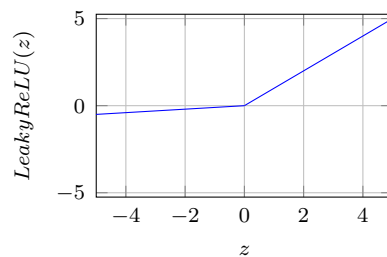


Figure 2.9: Leaky ReLU activation function.

2.2.4 Loss Functions

Loss functions (also known as cost functions) are used to train neural networks. They are computed by using the provided input (x_i) and the expected target (y_i) using the scoring function (applying the model) $s = f(x_i, w)$ with w being the model weights. One basic loss function is the Mean Squared Error (*MSE*) shown in equation 2.1.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i, w))^2 \quad (2.1)$$

For the MSE the sum of differences between model output and ground truth is squared and divided by the dataset size. In recent architectures, such as AlexNet [Szegedy et al., 2013], GoogleNet [Szegedy et al., 2016] and ResNet [He et al., 2016] log loss (also cross entropy loss) is used for classification. Equation 2.2 denotes the cross entropy loss function.

$$CrossEntropyLoss = -\frac{1}{n} \sum_{i=1}^n y_i \cdot \log(f(x_i, w)) \quad (2.2)$$

With the cross entropy loss the difference between predicted probabilities and ground truth probabilities is minimized. It gives the direction in which the weights have to be adapted so that the target value is most likely to occur when applying the weights to the input. The *log* of the output of the scoring function $f(x_i, w)$ using the provided input (x_i) and the model weights (w) is multiplied with the expected target (y_i) and summed up for all inputs. The negative value for the sum divided by the number of samples is the cross entropy loss.

2.2.5 Layers

Neural networks are structured in layers of neurons, all with the same properties, i.e. type. The multi-layer perceptron in Figure 2.5 consists of three fully connected layers. Fully connected layers, also known as linear layers, have connections from every neuron of one layer to all neurons of the previous layer. Every connection is assigned a weight/learnable parameter which works well for small images but is unusable for larger images. For example a 100×100 image with a first layer of only 1000 neurons has already ten million connections resulting in also ten million learnable parameters. Partially connected layers in convolutional neural networks (CNNs) solve this problem [Géron, 2019, p. 356-380].

Convolutional layers are partially connected by only connecting pixels to their receptive fields. In Figure 2.10 depicting a convolutional layer these connections go from the input image or the input feature map to the output feature maps. Neurons on the output layers are only connected to a small patch in the input layer.

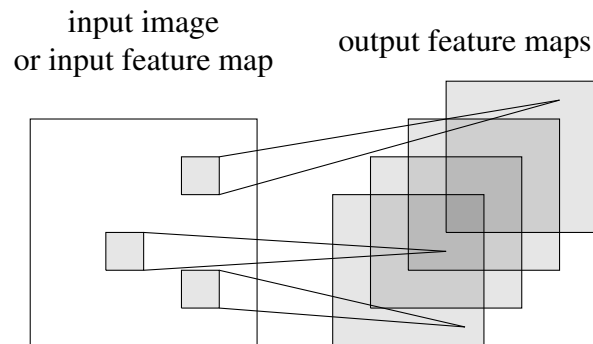


Figure 2.10: Applying convolutional filters to extract features.

This allows to create hierarchical features with low-level features in the first hidden layer and higher-level features in the next hidden layer. Since this hierarchical structure is common in real-world images CNNs work so well for image recognition. However with convolutions some spatial information containing the exact location of a high-level feature is lost. The small patches in Figure 2.10 are called filters or convolutional kernels and share their weights for one output feature map, which is the output of applying a filter to an input image or feature map. Parameters for a convolutional layer are the number of output feature maps (create more or fewer features), filter size, stride and padding. The filter size adjusts the size of the convolutional kernels which are moved with a stride over the whole input. The stride defines how far the filters are applied from the previous application of the filter. The distance to the input border is defined by the padding parameter.

Feature maps in convolutional neural networks capture gradually higher level features (outputs of deeper convolutional layers) that consist of lower level features (outputs of earlier convolutional layers). [Zeiler and Fergus, 2014] introduces a visualization technique which allows to inspect feature maps. In Figure 2.11 feature maps of a convolutional neural network are displayed. The Layer 1 feature map in Figure 2.11a shows low-level features (for example edges) of a small patch of the input image. With deeper layers the patches are combined creating higher-level features like car wheels (see Figure 2.11c) and faces (see Figure 2.11e).

Pooling layers subsample (shrink) the input image or the input feature maps to reduce computational load, the number of parameters as well as reducing the risk of overfitting. Similarly to convolutional layers patches of the input are processed to generate lower dimensional outputs. The pooling layer has the same parameters with respect to filter size, stride and padding. The difference is that a pooling filter has no weights, it just aggregates the input. Aggregations are typically the mean or the maximum of the input. The pooling layer in Figure 2.12 effectively reduces

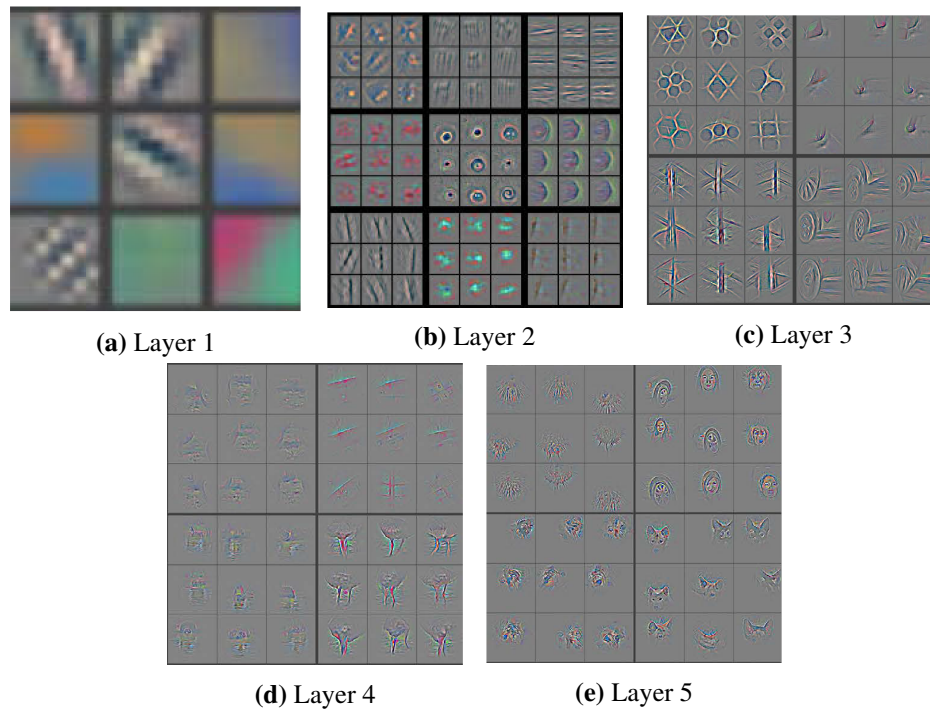


Figure 2.11: An example of feature map visualization [Zeiler and Fergus, 2014].

the size of the input feature maps by aggregation.

2.2.6 Training a Network

The previous sections describe the architecture of a neural network. However, a randomly initialized network is only capable of producing random results. For the network to accurately classify input data, we must first train the network. To do so the network tries to find the optimal weights and bias so that the network for input x approximates the ground truth label y (the result we wish the model to make). This section describes the different techniques used to train a neural network.

Data Splitting and Augmentation

To train a network we require a dataset of training data. The data should be similar to the data we want to classify in the future. A model which should classify cats and dogs, is best trained on pictures of cats and dogs. The first part is to split the dataset into three parts; **train**, **validation** and **test**. A common split is 70%, 20% and 10% respectively. The training set contains the data which is used to train the network. Validation is used to measure the model's performance during

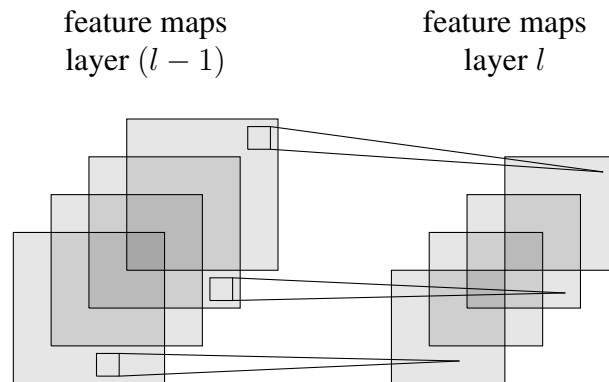


Figure 2.12: Applying pooling filters to subsample feature maps.

training, making sure that the model is not overfitting (see section 2.2.6). The data contained in the test set should never be accessed during training. It should only be used to measure the expected performance of the model after deployment by measuring how well it classifies data the model has never seen before.

Another technique to artificially increase the training dataset is to augment the data. Common augmentations transform the images using rotation, flipping and scaling (see examples in Figure 2.13). Larger datasets usually produce better models, due to better generalization [Cubuk et al., 2018], and augmentations can create multiple slightly different versions of the same image.

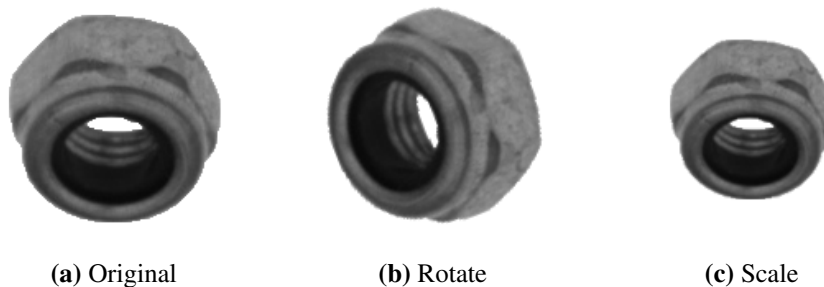


Figure 2.13: Nut augmentations

Initialization

Before training a neural network, the weights and biases must be initialized. Commonly pretrained models are used for initialization. A pretrained model is a neural network that has already been trained before. To implement a CNN for classifying images using the VGG architecture, we can initialize the weights using

a pretrained VGG model trained on ImageNet [Russakovsky et al., 2015b]. ImageNet consists of 1000 different classes and therefore gives a good foundation for any other image classifier. Pretrained models have been shown to converge faster and have better generalization [Taylor and Stone, 2009], by already learning filters that can recognize common structures such as lines, circles etc. After initialization using a pretrained model, it is important to replace the last classification layer with a randomly initialized layer. The last classification layer's output is a vector of the same length as the number of classes. Therefore, it must be replaced with a layer with as many output nodes as classes we wish in our new model. The other common approach is to use random initialization. In this case the neural network's weights and bias is randomly set and will at first produce random predictions.

Forward Propagation and Loss Functions

After the weights are initialized the first step in the training process is sending the data through the network. This process is called **forward propagation**, and produces a prediction vector as described in 2.2.2. The data is forward propagated in batches. The batch contains multiple samples, which are all passed through the network before an optimization step (see section 2.2.6). In the case of random initialization, the model will predict each class the same amount of times, and therefore not be better than random. To compare the output result to the ground truth, we introduce a **loss function**. The loss function produces a loss which tells the model how wrong the prediction is compared to the ground truth. One of the most common loss functions is cross entropy loss.

$$CrossEntropyLoss = -\frac{1}{n} \sum_x (y \ln a + (1 - y) \ln(1 - a))$$

Where x is the input data, y the ground truth and a the model's outputted prediction. Cross entropy loss is a logarithmic loss function, where the loss has a logarithmic growth the more wrong the prediction is. This causes the model to learn quickly when it is very wrong, and then the learning will decrease as the model improves. During training we want to minimize the loss, where a loss of 0 indicates there is no difference between the prediction and the ground truth.

Optimization

The gradient descent algorithm for [Amari et al., 1996, p. 757-763] and variations are used to optimize neural networks to give a specific output to a specific input. Gradient descent is an optimization algorithm which can find the global minimum for any convex function (a convex function has one global minimum, no local minimum and is continuous) [Géron, 2019, p. 111-120]. As mentioned,

the goal is to minimize the loss function. This can be performed by randomly tweaking all the weights and bias and recalculate the loss and use the settings that produce the lowest loss. However, this brute force method quickly becomes impossible as the network grows. To combat the scalability issue *gradient decent* algorithms are commonly used. Gradient decent tries to find the minimum of a function. If the function is convex, it will find the global minimum. However, in most real-life scenarios the functions will not be convex and instead have many local minimas. In other words, we want to find the deepest valley in the function. For each training sample in the dataset, we find the derivative of the loss function (L) with respect to each of the weights and biases (θ) to get a vector of gradients $\frac{\partial L}{\partial \theta}$. This process is explained in more detailed in the section 2.2.6. Finally, all the vectors are summed and averaged. The product is a vector of gradients that all point uphill in the function. Therefore, we need to update the weights and bias in step in the opposite direction of the gradient. This is called an optimization step. To control how far we step we define the hyper-parameter *learning rate* (α). The parameters θ are then updated using the following formula:

$$\theta = \theta - \alpha \frac{\partial L}{\partial \theta}$$

The learning rate is a very important hyper-parameter, as a too high learning rate can cause the step to overshoot the minima, and a too low learning rate can cause the network to require a very long time to converge. Common practices to avoid this is to gradually decrease the learning rate as the network comes closer to a minima or use a different optimization algorithm such as ADAM [Kingma and Ba, 2014].

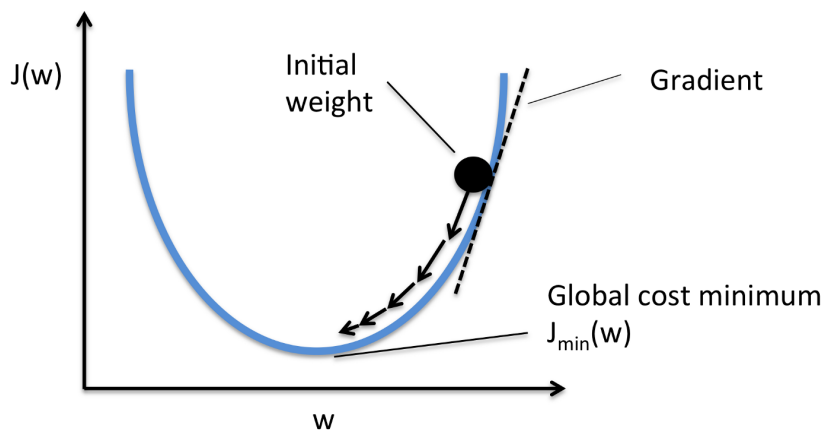


Figure 2.14: Visualization of a gradient descent example.

This optimization process will continue until convergence. The problem with this method is the computational time required to calculate the gradients for all

the parameters for all the training samples. Therefore most optimization methods use *stochastic gradient decent (SGD)* instead. SGD works exactly the same as gradient decent, but instead of calculating the gradient for all training samples, it only uses a single or a few samples before computing the optimization step. The result is much faster computation; however, the gradients will not necessarily point towards the global minima, rather towards the minima of the samples. Nevertheless, the trade-off is worth it as many inaccurate optimization steps converge faster than few accurate steps [Li et al., 2014].

Backpropagation

The neural network learns by examples and targets. The backpropagation algorithm presented in [Rumelhart et al., 1986, p. 533-536] is a method to repeatedly adjust the weights/parameters of the connections in the artificial neural network to achieve the desired input-output matching. The actual output is compared with the desired output and this difference (also called loss/cost see section 2.2.4) is minimized. By adapting their weights the hidden units represent features of the task-domain. In a feedforward network the flow of information from the input x multiplied with the weights w (parameters θ in section Multilayer Perceptron) to the output y is called forward-propagation [Goodfellow et al., 2016, p. 200-220]. The backpropagation allows information from the cost (computed by a function applied to output a desired output) to flow backward through the network to compute the gradient. The backpropagation itself only computes the gradient $\nabla_x f(x, y)$ (the slope of the function). A neural network can be described by a computational graph.

As described in section 2.2.6, we want to calculate the derivative of the loss in respect to each weight. However, the gradients of each layer depend on the gradient of all previous layers. We can view a neural network as a series of nested functions. We define a single convolutional block with three layers as the functions: convolution $c(x)$, activation $a(x)$ and pooling $p(x)$ and the output is the result of $p(a(c(x)))$. We apply the chain rule to compute the partial derivative of the loss with respect to each of the weights. The chain rules states that to calculate the derivative of the composition of two or more functions, for example $f(g(x))$. The derivative with respect to x is:

$$\frac{\partial}{\partial x} f(g(x)) = f'(g(x))g'(x)$$

Since a neural network is just a nested set of functions, by using backpropagation, we can find the gradient of the error with respect to each weight. These gradients are then averaged to produce the final gradient for each weight used by the optimizer.

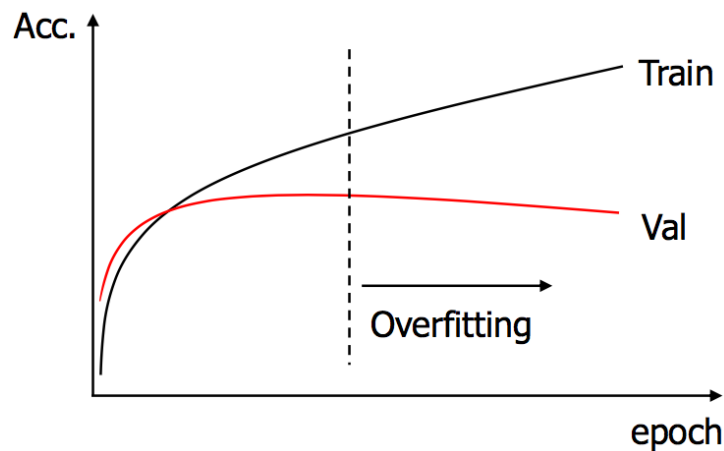


Figure 2.15: Overfitting

Overfitting

During training, a risk is overfitting to the training data. When overfitting occurs, the loss and classification is low because the model has only learned to differentiate the different data in the training set instead of the generalization of the classes. During training the model is seemingly performing very well, even reaching 100% training classification accuracy. To monitor whether the model is overfitting or not during training, validation is used. The validation set contains data which the model has not seen during the current epoch (one full forward propagation). When overfitting occurs, the training accuracy will be high, but the validation low as seen in Figure 2.15. To avoid an overfitted model, we can either deploy the model with the highest validation accuracy or introduce different measures to avoid overfitting the training data. Augmentation as introduced in section 2.2.6 is an effective way by changing how a class looks to force the model to learn the generalization of the class. Dropout is also an effective method, by randomly turning off neurons in the fully connected layers based on a probability. Finally, ensemble methods combine the weights of the same network from different training sessions or completely different networks to better generalize the training data.

2.3 Convolutional Neural Networks

CNNs are used for many different tasks in image recognition. The following sections describe the tasks classification, object detection, semantic segmentation as well as instance segmentation listing possible network architectures. Architec-

tures are described and design choices are analyzed.

2.3.1 Classification

Classification for image recognition tries to match an input image to one specific class. The main idea is that CNNs are extracting lower level (from first layers) and higher level features (later layers) and these features are combined to numeric representations. These numeric representations then are combined to output the class probabilities for the target classes. A high probability means that the model is certain, that the image belongs to the class. CNNs for classification are trained by providing images of the target classes and adapting the model weights to output the target class for the inputs. There are different types of models and architectures using combinations of different layers and activation functions.

The ImageNet challenge [Russakovsky et al., 2015a] is a benchmark for image category classification with hundreds of classes and millions of images. Modern networks tend to report their performance on the ImageNet challenge. For applications not only the performance but also the model size, training and inference (applying the model) times are important. Benchmark results are usually reported for an average ensemble of multiple models of the same type. In an average ensemble for example five models trained on different subsets of the data output their probabilities for the target classes on an input image. The average of the probabilities is then used to determine the final result (with an average probability larger than 0.5 for a class the input is assigned to that class).

Model	Error Rate	Year
Alexnet	15,3%	2012
InceptionV3	3,58%	2015
ResNet152	4,49%	2015
Inception-ResNet V2	3,1%	2016

Table 2.1: Model results on the ImageNet 2012 classification challenge

In table 2.1 the results of various models on the 2012 ImageNet classification challenge is displayed. All results are obtained using ensembles of the architectures. The winner of the 2012 ImageNet Challenge AlexNet [Krizhevsky et al., 2012] achieved an error rate of 15,3% with an ensemble. AlexNet consists of five convolutional layers, max-pooling layers, normalization layers and three linear layers. AlexNet is the first model that uses ReLU as an activation function. Further architectures like ZFNet [Wan et al., 2013] and VGG [Simonyan and Zisserman, 2014] increase the number of layers, filters and the general model size to improve the error rate but don't change the general style of stacking convolutional

layers and max-pooling layers for feature extraction as well as linear layers for mapping the features to the target classes.

In GoogLeNet (InceptionV1) [Szegedy et al., 2015] the idea of inception modules is introduced. A naive inception module on the left of Figure 2.16 extracts multilevel features using different-sized convolution kernels. By adding 1×1 filters (right inception module in Figure 2.16) the model can scale feature maps. Then the output features are concatenated and passed to the following layer. InceptionV3 [Szegedy et al., 2016] slightly improves InceptionV1 by using a different training algorithm (RMSprop, another method for using an adaptive learning rate) and an auxiliary head with batch normalization (scaling of activation functions to output in a range from zero to one) to improve training. With a relatively low model size InceptionV3 compared to VGG an ensemble still is able to produce a low error rate of 3.58% on the ImageNet challenge.

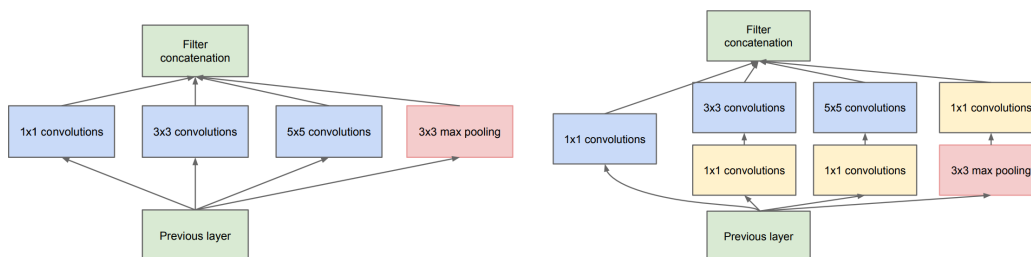


Figure 2.16: Inception modules for multilevel feature extraction [Szegedy et al., 2016]

Residual Neural Networks (Resnets) [He et al., 2016] solve the problem of the vanishing or exploding gradient (deep layers are not reached by the gradient or weights are just added up saturating the network). The network consists of residual blocks depicted in Figure 2.17. The identity function (also skip-connection) of the residual block allows the gradient of a higher layer to be directly passed to a lower layer during backpropagation. The input of a layer x is added to the output of a layer $F(x)$. This makes it easy for the network to learn the identity function for deeper layers, but also easier to find small fluctuations. With residual blocks allows networks can be deeper since even deep layers are reached by the gradient as well as shallower networks can be emulated using the identity function. An ensemble of Resnets with a depth of 152 layers scored an error rate of 4,49% in the 2012 ImageNet challenge, which is lower than InceptionV3, but won the 2015 ImageNet challenge.

Inception ResnetV2 combines the ideas of InceptionV3 and Resnet into one architecture [Szegedy et al., 2017]. An identity function is added to the multi-level convolutional layers of InceptionV3 replacing the Max-Pooling Layers. An ensemble of this architecture achieved an error rate of 3,1% beating InceptionV3 and Resnet.

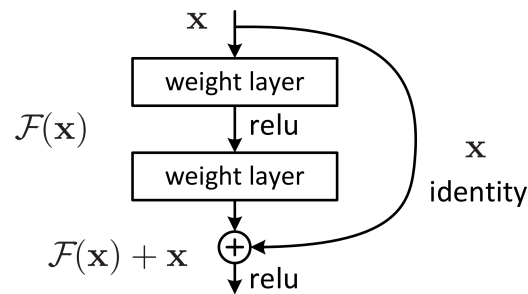


Figure 2.17: Residual block with identity function for free gradient flow [He et al., 2016]

2.3.2 Multi-view Convolutional Neural Networks

Multi-view convolutional neural networks (**MVCNN**) are mostly the same as normal CNNs, with one big difference. Instead of a single input to produce a single prediction, MVCNNs use multiple inputs for a single prediction. MVCNNs were first designed to classify 3D objects. Princeton's ModelNet [Wu et al., 2015] challenge consists of 40 different classes of 3D computer-aided design (**CAD**) models. A CAD model is a vector graphic normally used for digitally designing components for manufacturing. By taking multiple images of an object (or a digital rendering of a CAD model) simultaneously, we can capture more details about the object than with just a single view. MVCNN therefore allows us to capture and process images of an object which contains crucial features that cannot be captured from any single angle.

The Princeton ModelNet¹ project published a 3D CAD model dataset to provide a comprehensive benchmark to develop 3D capable neural networks [Wu

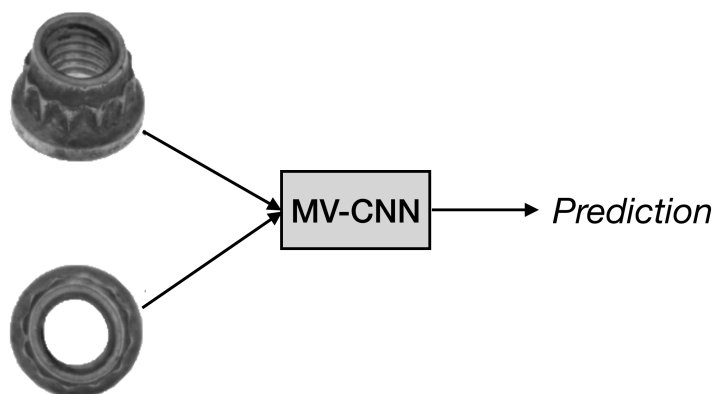


Figure 2.18: Multi view convolutional neural network

¹<http://modelnet.cs.princeton.edu/>

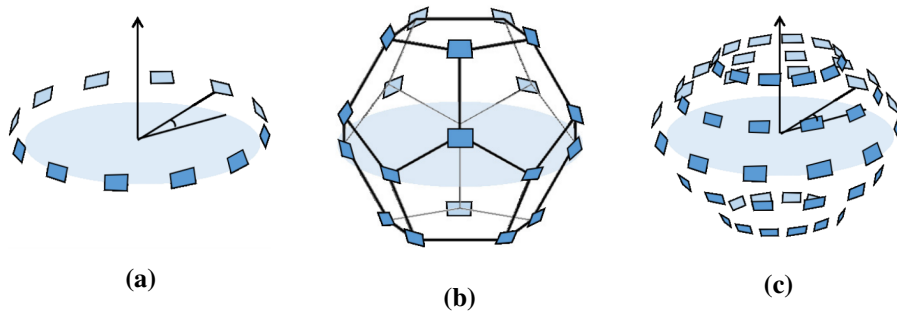


Figure 2.19: Example of artificial camera configurations with 12 (a), 20 (b) and 60 (c) angles [Kanezaki et al., 2018].

et al., 2015]. All the CAD models in ModelNet do not necessarily contain any texture or color. Therefore, the only discriminative features are the object shapes. The total dataset contains 662 object categories, where there are two main subsets for benchmarking that contain 10 and 40 classes, called ModelNet10 and ModelNet40 respectively. The performance of the classification models is measured with classification accuracy.

Multi-view CNNs try to address the problem of 3D points sparsity by projecting 3D objects to 2D representations of the same object. By transforming the 3D data to 2D, we effectively extract the features of the object. In feature extraction, part of the data, which is not considered as important, is discarded. Methods such as principal component analysis (PCA), use the largest eigenvectors of a covariance matrix to project the data into a smaller subspace with fewer dimensions [Holland, 2008]. In the same way, by discarding a dimension from the 3D objects, we reduce space and required computational power, and addressing the *curse of dimensionality* [Bellman, 2013]. ModelNet’s CAD models are rendered using a rendering software, such as Blender², and the 2D images can be artificially captured from any angle. Figure 2.19 shows three potential artificial camera configurations with 12, 20 and 60 angles respectively. By capturing one image per angle, we have n 2D images instead of a 3D object. Multi-view CNN takes all n images as input to produce a single prediction.

One way to implement a form of Multi-view CNN is to use a single network which takes single images as input. Here, we consider each image as an individual image (instead of aggregating the images together). This approach ignores the fact that many of the images can be of the same object at the same time. The benefit is the simplicity of implementation; there is no need for aggregating multiple views. However, the main disadvantage is that the information sharing between views is lost; the network will not be able to confidently classify any angle that does

²<https://www.blender.org/>

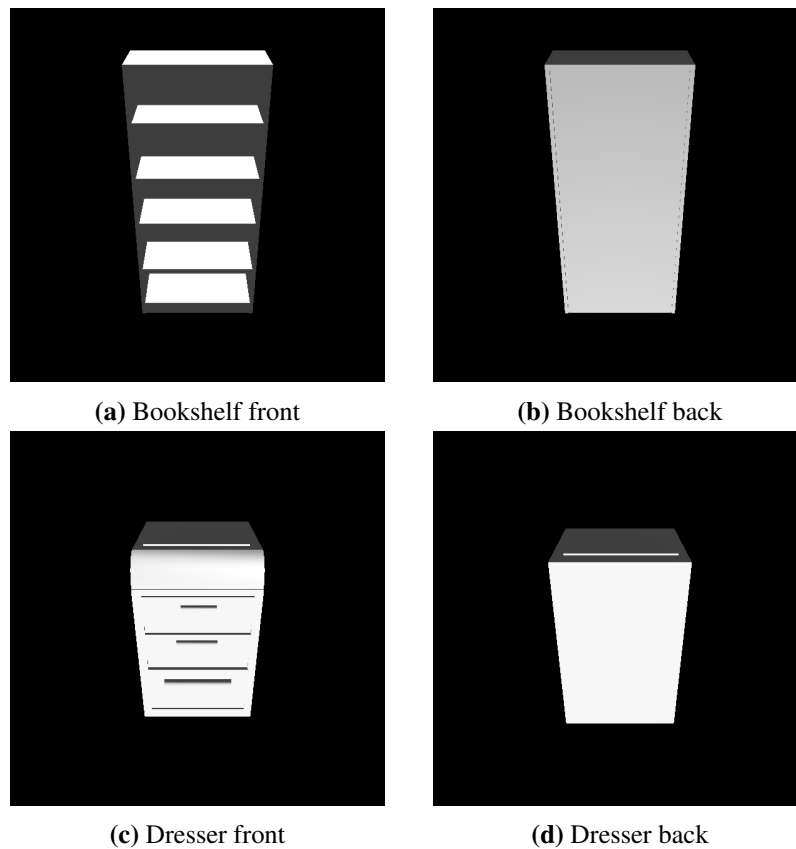


Figure 2.20: Bookshelf and dresser from different angles

not contain discriminative features. Figure 2.20 shows that the classes *Bookshelf* and *Dresser* from ModelNet have no explicit features to distinguish between them from the back view. Either object can be a smaller or taller variant of each other. However, the front view shows that bookshelves have shelves, while dressers have drawers. This example illustrates that it is necessary to share information between views to be able to classify even everyday objects.

Multi-view CNN uses the artificial camera configuration using 12 angles, as shown in Figure 2.19a. Multi-view CNN can be realized as an extension of a normal CNN. For each multi-view image (n images that are all of the same object), each image is passed through the feature extraction layers independently to produce n sets of feature maps. Afterwards, these n sets are reduced through max-pooling layers. The result is a set of feature map of the highest activations between all n angles. This feature map is then squeezed and used as input to the classification layers.

The information sharing between the different views is the advantage compared to a standard CNN. However, pooling multiple feature maps from different

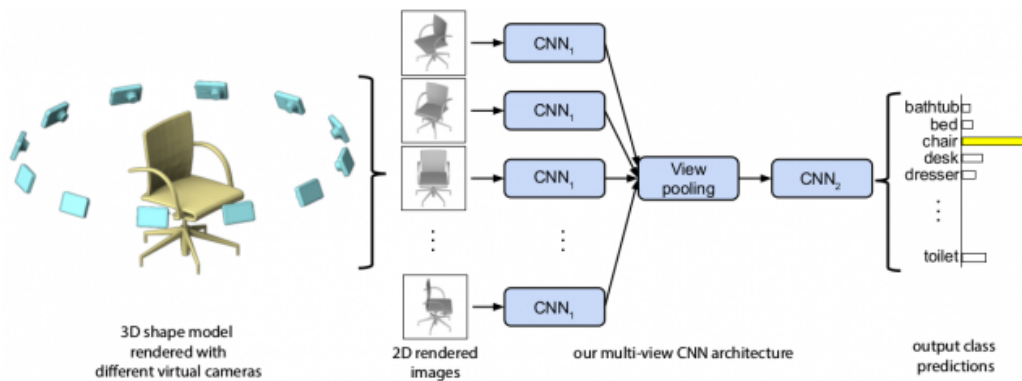


Figure 2.21: Multi-view CNN architecture [Su et al., 2015].

views can cause the result to become unrecognizable. Multi-view CNN achieves 90.1% classification accuracy on the ModelNet40 dataset, outperforming the original volumetric based 3DShapeNets which had an accuracy of 77.3%.

2.3.3 Rotation Net

RotationNet also uses the same idea of Multi-view CNN [Kanezaki et al., 2018]. Similar to Multi-view CNN, it uses multiple views during training. However, during classifying unseen data, RotationNet only requires a subset of the views. RotationNet is able to classify the object even with only a single view instead of the same number used during training. It also differentiates itself by not requiring the objects to be aligned. The objects can lay in any orientation in 3D space. RotationNet achieves this by introducing a latent variable which is optimized during training to learn which viewpoint is the most similar to one the network has seen during training.

As mention, RotationNet only requires a subset of views during inference and is therefore much more practical in real application, where it might not be possible to capture all angles. RotationNet showed its success on a new multi-view dataset *Multi-view Images of Rotated Objects (MIRO)* [Kanezaki et al., 2018], consisting of multi-view images of real objects. In Figure 2.22a we observe that Multi-view CNN is still outperforming RotationNet on ModelNet40, though it is noteworthy the amount of accuracy achieved by RotationNet with fewer views. In Figure 2.22b, using the real object dataset MIRO, RotationNet notably outperforms Multi-view CNN, achieving the same results with only a subset of 2 views during inference.

2.3.4 Siamese Networks

A Siamese network is one of the simplest and most popularly used one-shot learning algorithms. One-shot learning (see also 2.4.2) is a technique in which the network learns from only one training example per class. Siamese network is predominantly used in applications that the classes lack many instances.

Siamese networks consist of two symmetrical neural networks that share the same weights and architecture. The two blocks join together at the end using an energy function, E . The objective of the Siamese network is to learn whether two input values are similar or dissimilar. Figure 2.23 shows the architecture of a sample Siamese network.

The output is flattened into a number in $[0,1]$ with a sigmoid function to make it a probability (1 when the images have the same class and 0 for a different class). Since the siamese network is trained with logistic regression, the loss function should be binary cross entropy between the predictions and targets.

2.3.5 Object Detection

For object detection a classification CNN is typically combined with a bounding box regressor. Neural networks, performing the object detection task, output the probabilities for multiple sliding windows (smaller rectangles on the input image) to find objects. The selection of the sliding windows differs by algorithm, windows of the same class which are close to each other are combined to bounding boxes (marking a rectangular position) for the objects. In Figure 2.24 a sample of an image with probability bounding boxes is shown. CNNs for object detection have to be trained with images including the annotated bounding boxes.

For object detection currently two approaches are used one is first extracting region proposals, then computing features as well as classifying the regions. The classified regions are the bounding boxes output of the network. R-CNN [Girshick et al., 2014] uses selective search [Uijlings et al., 2013] for region propos-

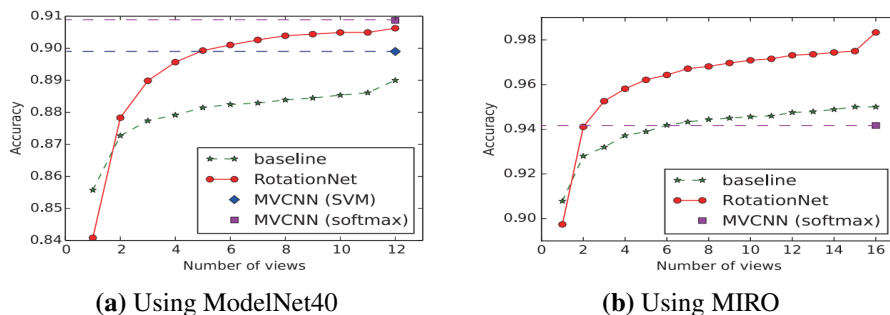


Figure 2.22: RotationNet/Multi-view CNN results [Kanezaki et al., 2018]

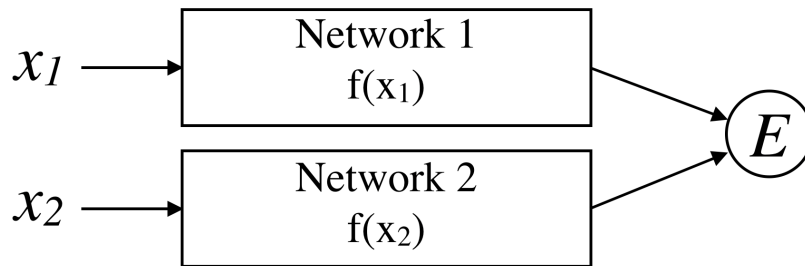


Figure 2.23: Siamese network architecture.

als, Alexnet as a feature extractor and multiple Support Vector Machines (SVMs) trained for each class to identify the region. Fast R-CNN [Girshick, 2015] and Faster R-CNN [Ren et al., 2015] improve detection as well as training by combining region proposals, classification into one network and also using a better performing architecture for classification (VGG).

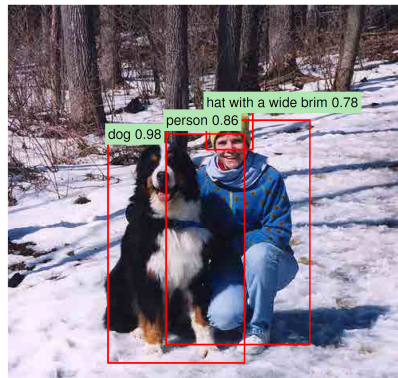


Figure 2.24: Object detection output [Girshick et al., 2014]

The second approach Single Shot MultiBox Detector SSD [Liu et al., 2016], discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. Each box is classified by the network and the default box is adjusted to match the object shapes. Multiple feature maps are combined to handle objects of varying sizes. SSDs are faster than region proposals since the image is just processed once.

2.3.6 Object Segmentation

There are two different levels of object segmentation, namely, semantic segmentation and instance segmentation. Semantic segmentation is a pixel-wise clas-

sification of images. As shown in Figure 2.25 each pixel is assigned a class, which is visualized by a color in the image. In semantic segmentation unlike object detection instances of classes are not differentiated. For the training also pixel-wise annotations are needed.

Fully Convolutional Networks (FCNs) [Long et al., 2015] reduce the dimensionality of an input image extracting features and then upsample again to the input size with on feature map for each class. The FCN in Figure 2.25 is trained end-to-end by providing the pixel-wise class masks. The network learns the specific tasks of pixel-wise segmentation. For semantic segmentation a background class is added representing everything different from the target classes.

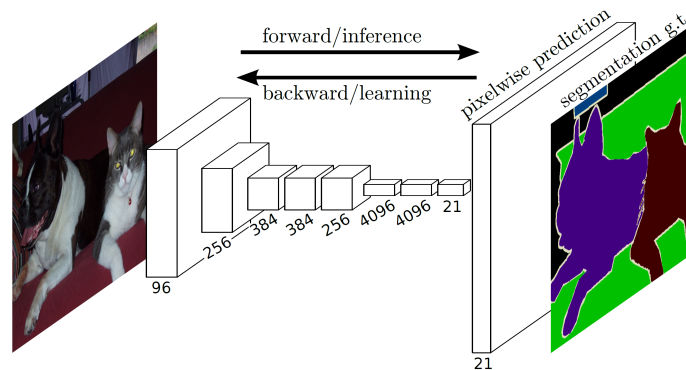


Figure 2.25: Semantic segmentation using a FCN [Long et al., 2015]

Region proposal networks (see section 2.3.5) are also used for semantic segmentation. For each pixel in a proposed region the class is predicted.

Instance segmentation combines object detection and semantic segmentation outputting pixel-wise classification of instances. The output as portrayed in Figure 2.26 is a mask for each detected instance. For training the pixel-wise annotations of the instances have to be provided.

Mask R-CNN [He et al., 2017] extends Faster R-CNN (see section 2.3.5) by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is a general approach which allows exchanging of components. The backbone component for feature extraction can be any CNN like for example Resnet (see section 2.3.1). The network head component is the used for bounding-box recognition (classification and regression) and mask prediction that is applied separately to each region of interest.

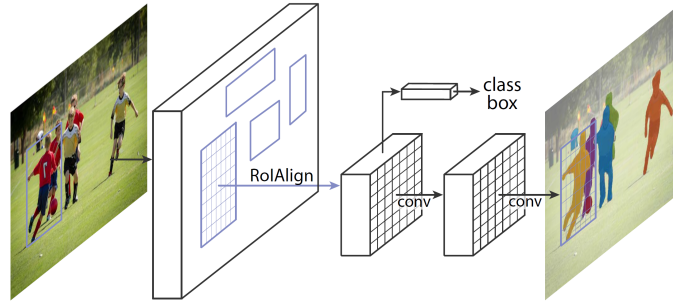


Figure 2.26: Instance segmentation using Mask R-CNN [He et al., 2017]

2.4 Unsupervised Techniques for Anomaly Detection

Classification problems deal with two or more classes. The goal is to distinguish between a number of classes using the training data. Anomaly detection (also outlier detection and one-class classification) deals with the problem of only having one class in the training data and the target to determine if new data is alike the training data. In the following sections methods for outlier detection are described.

2.4.1 Autoencoder

An autoencoder is a method of obtaining a lower dimensional representations of the input dimension. The general idea is that an autoencoder tries to learn how to copy the input to an output [Géron, 2019, p. 415-440]. Since the autoencoder is constrained to pass the input information through a lower dimensional bottleneck (the code) it learns an efficient representation of the input data. Autoencoders are unsupervised since no labels are needed for the training. Autoencoders typically consist of an encoder and a decoder part. The encoder part reduces the dimensionality of the input resulting in the code. The decoder then tries to reconstruct the input from the code. Autoencoders are normally trained with the mean squared error as a loss function (see section 2.2.4).

Linear Autoencoder

A linear autoencoder consists of multiple linear layers for the encoder and the decoder component (see section 2.2.5) each followed by an activation function (see section 2.2.3). The code is a linear layer with a reduced amount of neurons. Since only fully-connected layers are used the amount of parameters for large

images is very large making the linear autoencoder difficult to train and memory intensive.

Convolutional Autoencoder

Convolutional autoencoders consist of multiple convolutional layers (see section 2.2.5) each followed by a activation function (see section 2.2.3) A combination of filter size and stride is used to adapt the image size to extract features and condense the information to the lower dimensional representation. Decoding is achieved by deconvolutional layers which invert the convolutions by applying filters to upsample the input. A convolutional autoencoder like a CNN for detection is able to extract image features and can be built with less parameters.

Variational Autoencoder

Variational autoencoders introduced by [Kingma and Welling, 2013] differ from the previously introduced autoencoder types by being probabilistic (outputs are partly random) and generative. Generative means that they can produce new instances looking like they're sampled from the training set. Variational autoencoders have the same encoder-decoder architecture. The coding is not directly produced by the input, but by summing a random sample from a Gaussian distribution multiplied by the standard deviation of the coding and the mean coding. The loss of the variational autoencoder is composed of the standard mean squared error and the latent loss. The latent loss rewards the autoencoder coding too look like they're sampled from a Gaussian distribution. This latent loss supports the generative properties of the autoencoder as the standard deviation of the coding represents the difference of multiple samples allowing to "morph" from one output to another.

2.4.2 One-class Support Vector Machine

A Support Vector Machine (SVM) is a model for classification and regression problems. It can be used to solve linear and non-linear problems. SVM creates a line or a hyperplane which separates the data into different classes. The non-linear decision boundary is achieved by projecting the data through the non-linear function ϕ . SVM maximizes the distance of the support vector (vector consisting of all features) of each to the separating hyperplane which is constrained by the kernel function. Analogous the one-class SVM [Schölkopf et al., 2000] maximizes the distance of the hyperplane to the origin as shown in Figure 2.27a. The parameters for the SVM classifier determine the trade-off between maximizing the margin

and the number of training data points within that margin. A one-class SVM as depicted in Figure 2.27b can also have a non-linear decision boundary.

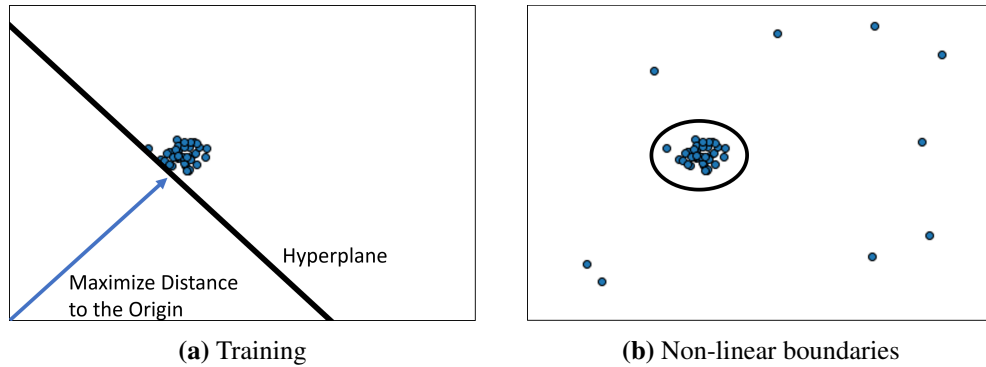


Figure 2.27: One-Class SVM

2.4.3 Isolation Forest

Isolation forests separate observations by randomly selecting a feature and then splitting the observations by a random value between the maximum and minimum value of the feature [Liu et al., 2008]. This "isolation" process is executed multiple times building multiple decision trees which are combined into a forest. The number of splits to isolate a sample is the path length from the root node to the terminating node in a decision tree. With the average path length over a forest of random trees a measure of normality is defined.

Anomalies are easier to separate from the normal data points, with a shorter path length samples are likely to be anomalous. In Figure 2.28a an example of random splits separating a normal data point is shown. With less splits as visual-

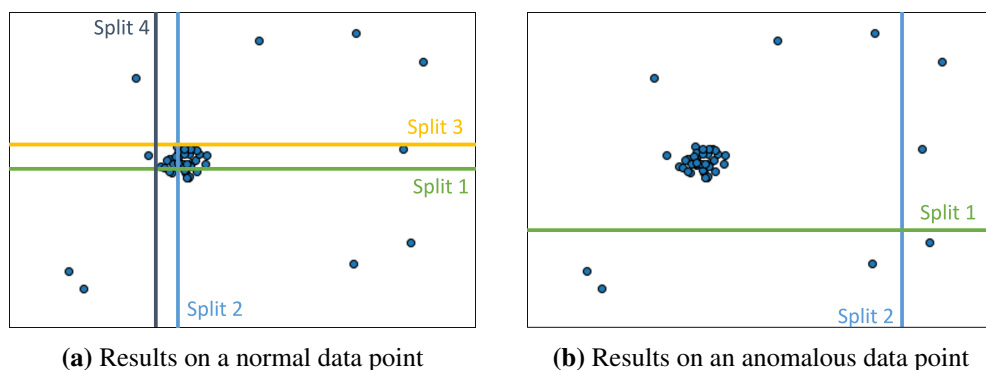


Figure 2.28: Isolation Forest

ized in Figure 2.28b an anomalous data point can be separated. Parameters define the number of trees, features selected and the threshold for the decision function.

2.4.4 Local Outlier Factor

Local outliers are outliers compared to their local neighborhoods, instead of the global data distribution. The local outlier factor is the assigned degree of being an outlier [Breunig et al., 2000]. The local outlier factor is based on a concept of a local density, where locality is given by k nearest neighbors. The distance to the neighbors is used to estimate the density. Comparing the local density of an object to the local densities of its neighbors, outliers can be identified (see Figure 2.29). Outliers have a substantially lower density than their neighbors. Parameters are the number of neighbors, the distance metric and the threshold for the decision function.

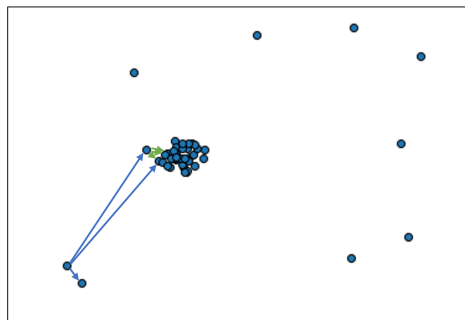


Figure 2.29: Local Outlier Factor determining density with $k=3$ neighbors

2.5 Evaluation Metrics

There are several metrics used by researchers and practitioners to evaluate the performance of machine learning and deep learning models. Each of these metrics measures different aspects of a trained model. Below, we present a brief description of each method.

- **Accuracy:** the proportion of correctly classified instances out of a given set of instances. When the classes are heavily imbalanced, this metric is less useful.

$$Accuracy = \frac{TP + TN}{allinstances}$$

- Precision: the proportion of correct classified items out of all the predictions classified as the given class.

$$Precision = \frac{TP}{TP + FP}$$

- Recall: the proportion of correct classified items out of a given set of the class instances. This metric represents the class detection capability of the model.

$$Recall = \frac{TP}{TP + FN}$$

- True negative rate (Specificity): the proportion of correct false classifications out of a given set of false instances.

$$Specificity = \frac{TN}{TN + FP}$$

- False positive rate: the proportion of instances classified incorrectly as true out of a given set of false instances.

$$FalsePositiveRate = \frac{FP}{TN + FP}$$

- False negative rate: the proportion instances classified incorrectly as false out of a given set of true instances.

$$FalseNegativeRate = \frac{FN}{TP + FN}$$

- F1 score: the harmonic mean of precision and recall. For the problems with imbalanced classes, this is a useful metric to evaluate models.

$$F1 = 2 \frac{Precision \cdot Recall}{Precision + Recall}$$

Chapter 3

PPIC: Platform for Parts Image Capturing

This chapter describes *PPIC*, an image capturing platform that addresses the industrial parts challenges. Parts of the presented work in this chapter have been previously published [Taheritanjani et al., 2019a] [Taheritanjani et al., 2019b]. Section 3.1 describes the characteristics of industrial parts and the challenges of capturing images. Section 3.2 describes the image capturing configuration and its components.

3.1 Industrial Parts Characteristics and Challenges

To obtain a dataset for sorting and classification of industrial parts, the training dataset must contain the main discriminative features [Chai, 2015]. In addition, image segmentation for damage detection and bin picking of fasteners requires quality images. There are three challenges for industrial parts in comparison with most other image datasets [Deng et al., 2009] [Huang et al., 2008] [Khosla et al., 2011]. Firstly, unlike the cases in which the background of the image can contribute to the classification, in inspection processes the background is not informative, can be a source of noise, and results in unnecessary computation overhead. Secondly, many classes can only be separated by subtle details, for example small length differences or difference in the thread pitch for bolts. Figure 3.1 is an example of such cases. Thirdly, capturing images of industrial parts faces challenges due to the nature of the parts themselves:

- **Perspective:** in the graphic arts, perspective is an approximate representation of an image on a flat surface as it is seen by the eye. Perspective is characterized by its correlation between object's size and its distance to

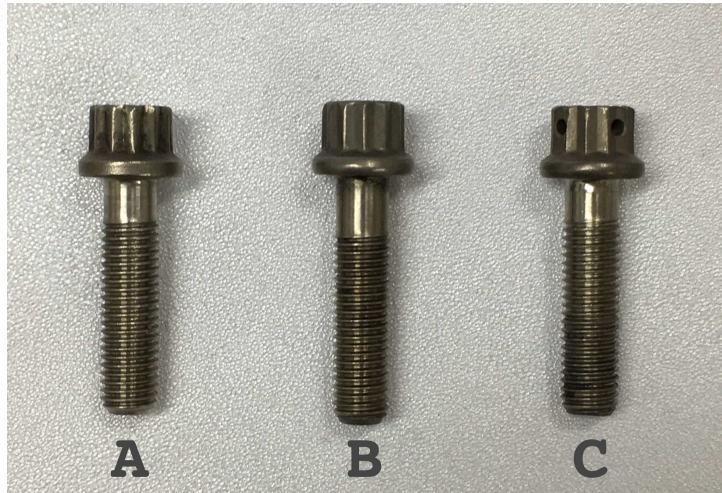


Figure 3.1: An example of similar fasteners. A is identical to C except that its non-threaded shaft is one millimeter shorter. B has a two millimeter longer non-threaded shaft and a one millimeter smaller diameter than C [Taheritanjani et al., 2019a].

the observer. The objects become smaller as their distance from the observer increases [Grau, 2003]. Unlike most of the objects where scaling the size, width or height does not affect their actual category, the size of the fasteners must be preserved in all the instances of the dataset. Any scaling transformation of the image of the fastener can result in misclassification. Figure 3.2 shows an example of two bolts which are scale versions of each other, considering their length and width. Their distance to the camera lens must be known and always scaled to a fixed size in all the data samples.

- **Reflection:** most of the parts used in mechanical machines are metallic and primarily silver color. Their surface reflects the light, which might introduce extra noise and hides important details of the part in the image such as the fastener threads or the damages on the part surface. Therefore, a specific light polarization must be utilized to control and reduce the light reflection.
- **Shadow:** laying the parts on a surface with a normal source of light such as room lighting results in them being surrounded by shadow. Considering the small size of some parts like fasteners and the generated shadows, shadows can be a source of noise which makes the data augmentation process less useful.

Aside from the challenges, the shape, color, and texture have previously been studied as the most common visual image features [ping Tian et al., 2013] [Stanchev

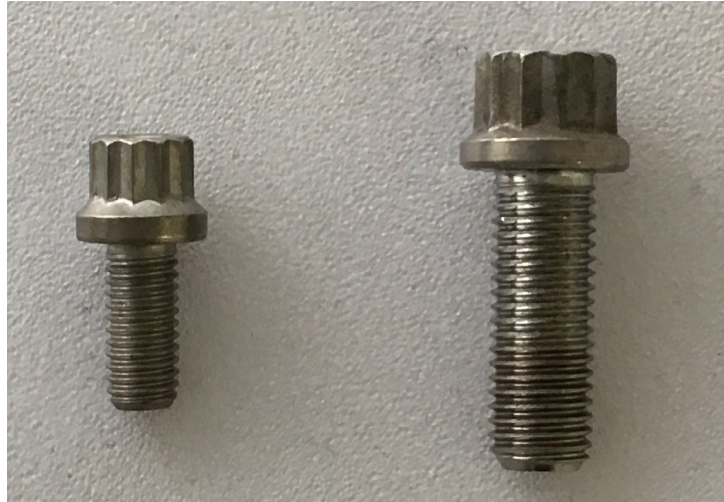


Figure 3.2: An example of two bolts that each one is a scale version of the other one. Scaling one's image is an example of the perspective issue and can result in a misclassification [Taheritanjani et al., 2019a].

et al., 2003] [Zhang and Lu, 2004]. The parts used in the inspection process have typically metallic silver or dark color and a metallic texture, irrespective of their constituent alloy. As the texture and color are identical for most parts, we focus on shape, threads, and head.

The shape of a part can identify whether the part is a bolt, screw, nut, washer or crank and how large, small, narrow or thick it is. Moreover, for threaded parts such as bolts, screws, and nuts, the threads can be used to distinguish between different similar-shaped items. Differences in the thread pitch, size, and the length of the threaded shaft of the part indicate a different fastener. In addition, there are many screws and bolts which have the same shape, length and width and also the same threads—pitch, length and location of threads on the shaft. However, they have different heads such as hex socket, hex cap, flat head, or curved head bolts and screws. The fastener head indicates what kind of wrench or screw driver is required to drive it and different head types are appropriate for different threaded holes.

3.2 PPIC

Considering the challenges described in 3.1 - perspective, reflection, and shadow with normal ambient lighting, as well as the shape, threads, and head features on which we want to focus, we describe a platform for parts image capturing, *PPIC*, a research platform for recording images of the parts for creating the datasets,

training, and prediction.

To control the reflection on parts' surface, *PPIC* utilizes textiles to polarize the lights [Biver et al., 2012]. The textile diffuses the light, softening it so that the lighting is evened out. The even light from all angles ensures that there is no critical reflection. Moreover, to reduce the shadows around parts and have a uniform background, it uses dimmable backlighting. Shining a controlled light behind the object directly at the camera lens darkens the object, which again reduces reflection effects [Biver et al., 2012]. In addition to the backlighting underneath the surface, there are multiple lights in the room to spread light from above and the side. Furthermore, fixed cameras are used to overcome the perspective issue, which are placed inside the polarized space. Using fixed cameras, the distance between fastener and the camera lens is fixed (ignoring minor variation in distance resulting from the different positions of the fasteners on the surface). *PPIC* can also employ a non-fixed camera. However, we must compute the camera distance to the part, (for example, using fiducial markers), and scale the image to the initial fixed size.

To semi-automate the process of recording the images for the dataset, *PPIC* uses a backlight supported textile conveyor belt. To ensure that one fastener arrives under the camera at a time and to reduce the occlusion effect, it employs vibration to separate them. The vibration must be performed before the parts arrive under the camera. Finally, to fully automate the capturing process, *PPIC* uses a mechanical solution with another conveyor belt to move the parts at the end of the main conveyor belt back to its beginning. This can be also beneficial for the parts which do not arrive separately under the camera (occlusion). Using the second conveyor belt, *PPIC* can fully automate data creation, even if there are few samples from a part. Figure 3.3 shows a sketch of *PPIC*.

3.2.1 PPIC Configurations

PPIC, as shown in Figure 3.3, is a generic research platform, that can be configured for different use cases. Depending on the requirements and specifications of a study, we can use parts of *PPIC* for the hardware configuration. The light green parts of *PPIC* depicted in Figure 3.3 support semi-automatic dataset creation, as well as automatic collection and feeding back the leftover fasteners¹. While these light green parts help to automate the process, the inspection processes are performed under the cameras, as shown in yellow in Figure 3.3. Therefore, this section focuses on the under camera part of *PPIC* and describes four configurations, which are used in the rest of this dissertation.

¹Leftover fasteners refer to the ones that the automatic damage identification, sorting, classification or bin picking could not be applied to them, mainly due to their position relative to other nearby fasteners.

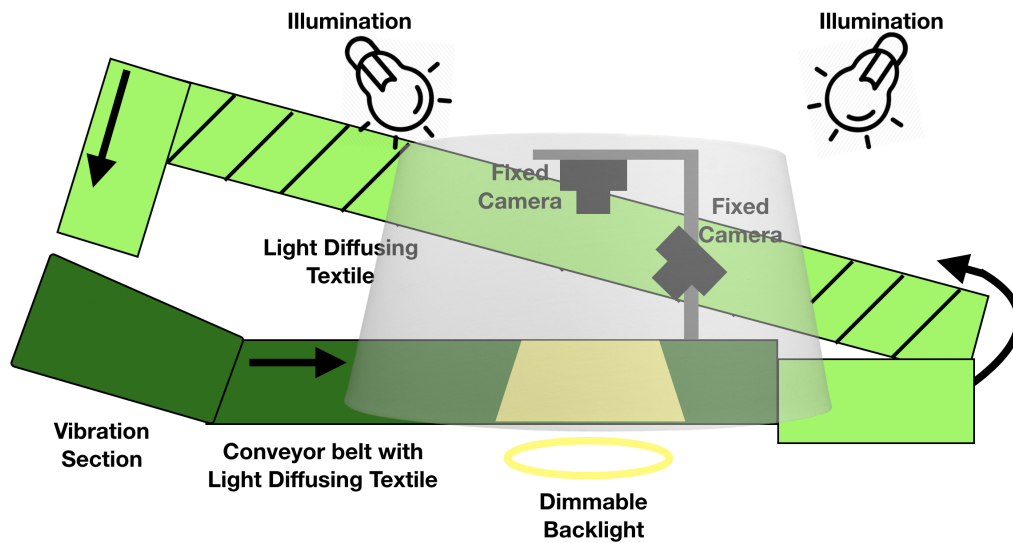


Figure 3.3: The sketch of *PPIC* with a vibration conveyor belt. The backlighting comes from below the conveyor belt. The light green section can move the parts back to the beginning of the conveyor belt and the vibration section.

***PPIC-A*: Configuration without Ambient Light and with Single Camera**

PPIC-A uses only a single camera on top of the conveyor belt. The camera lens angle with the conveyor belt surface must be 90° . The backlighting and light diffusing textile are required to reduce the reflection on parts and shadows around them. In addition, *PPIC-A* necessitates to turn off sources of illumination on top. Therefore, it does not require any light diffusing textile for polarization (Chapter 6 shows how this configuration setup can be used for single-view classification of fasteners). Figure 3.4 shows a sketch of *PPIC-A*. Figure 3.5 shows an image of two bolts and one washer obtained with normal ambient lighting (3.5a) and with *PPIC-A* (3.5b).

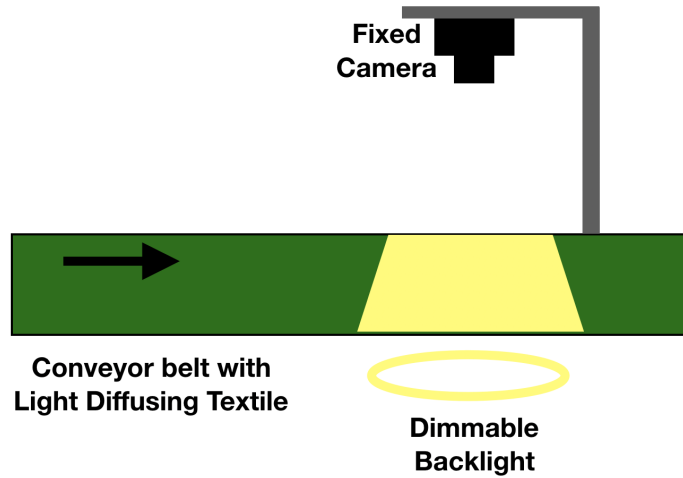


Figure 3.4: The sketch of *PPIC-A* with a light diffusing conveyor belt. The backlighting comes from below the conveyor belt and a single camera with 90° captures the images.

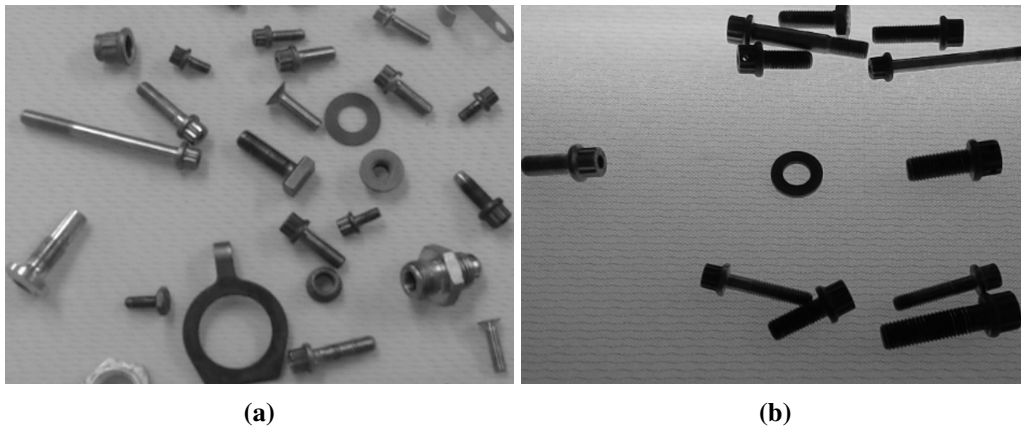


Figure 3.5: Image of fasteners obtained on the conveyor belt using normal ambient lighting (a) and using top camera view of *PPIC-A* with reduced shadows and reflection (b).

***PPIC-B*: Configuration with Ambient Light and with Single Camera**

PPIC-B employs again a single camera with a right angle towards the surface and backlighting and light diffusing textile conveyor belt. However, *PPIC-B* requires sources of illumination on top of the platform with a light diffusing textile for polarizing them. Figure 3.6 shows a sketch of *PPIC-B*. Figure 3.7 shows an

image of a bolt obtained with normal ambient lighting (3.7a) and with *PPIC-B* (3.7b).

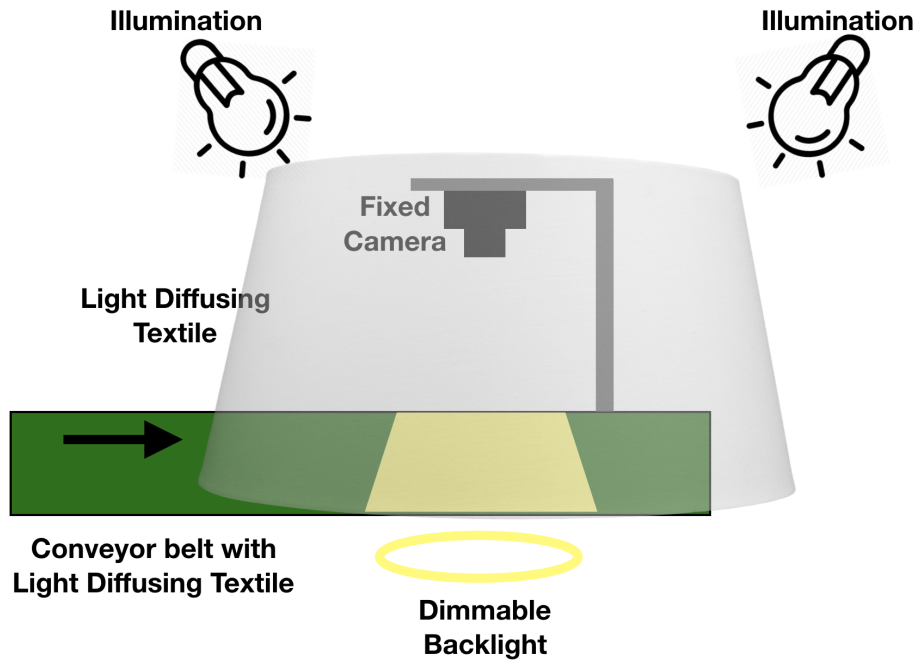


Figure 3.6: The sketch of *PPIC-B* with a single camera, light diffusing conveyor belt, and backlighting. There are illumination on top of platform and light diffusing textile to polarize the light.

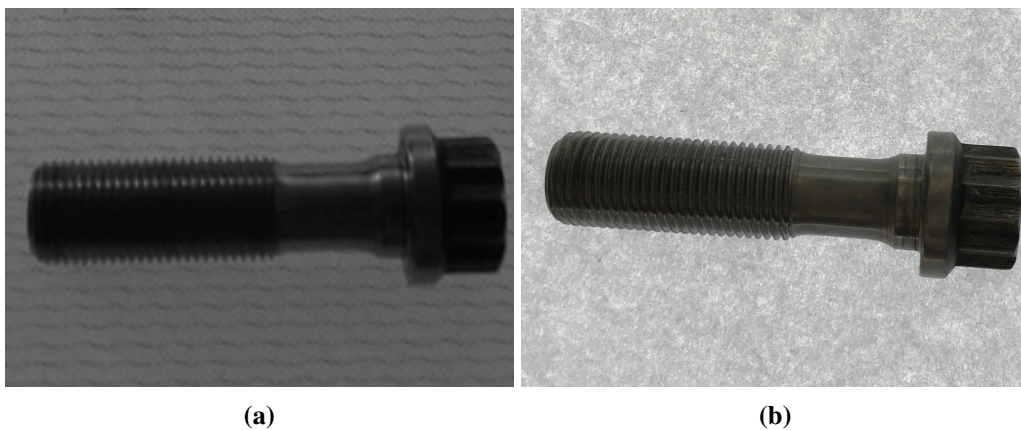


Figure 3.7: Image of a bolt obtained using normal ambient lighting (a) and using camera view of *PPIC-B* (b).

***PPIC-C*: Configuration with Ambient Light and with Two Cameras**

PPIC-C also utilizes sources of illumination on top of the platform with light diffusing textile for polarizing them. *PPIC-C* uses two cameras², one with 90° and another one with 60° angle towards the surface. Figure 3.8 shows a sketch of *PPIC-C*. Figure 3.9 shows an image of a nut from two views obtained with *PPIC-C*.

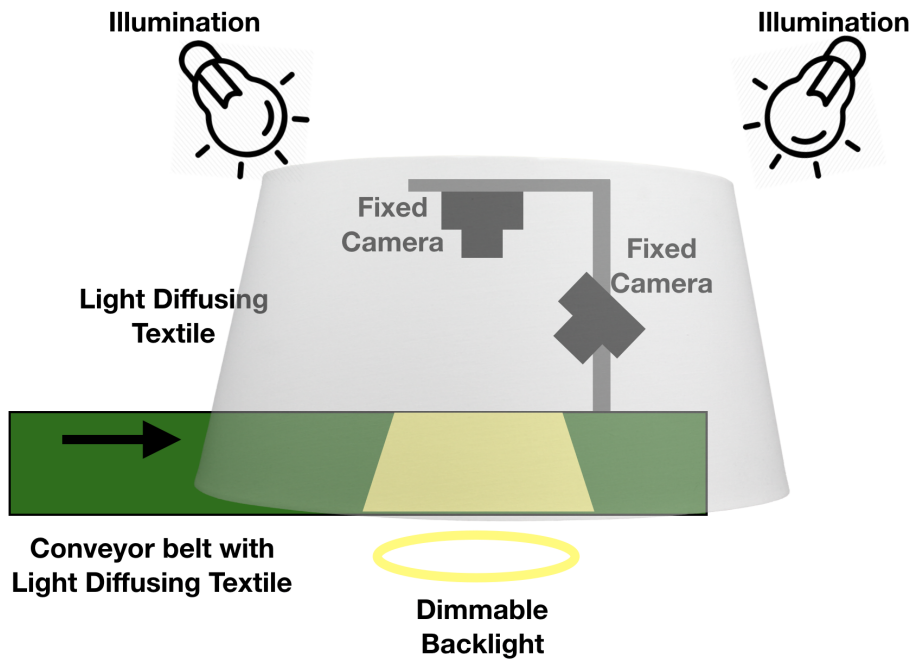


Figure 3.8: The sketch of *PPIC-C* with two cameras, illumination on top, and polarized backlighting supported conveyor belt.

²We only use two cameras for multi-view studies in this dissertation. However, the number of cameras and views can be extended.

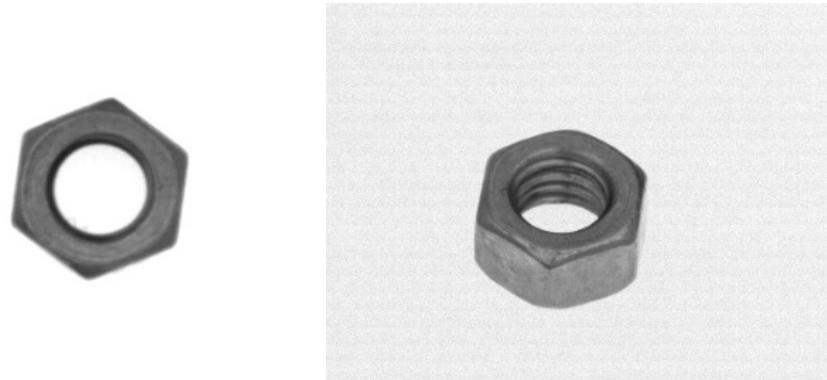


Figure 3.9: Image of a nut obtained using both camera views of *PPIC-C*.

***PPIC-D*: Configuration with Directed Illumination**

In this configuration, there is no light diffusing textile and backlighting. However, there is one illumination source on top of the conveyor belt. The illumination is directed with 45° angle towards the surface. The camera has 90° angle towards the surface. Figure 3.10 shows a sketch of *PPIC-D*. Figure 3.11 shows an image of a surface obtained with normal ambient lighting (3.11a) and with *PPIC-D* (3.11b)

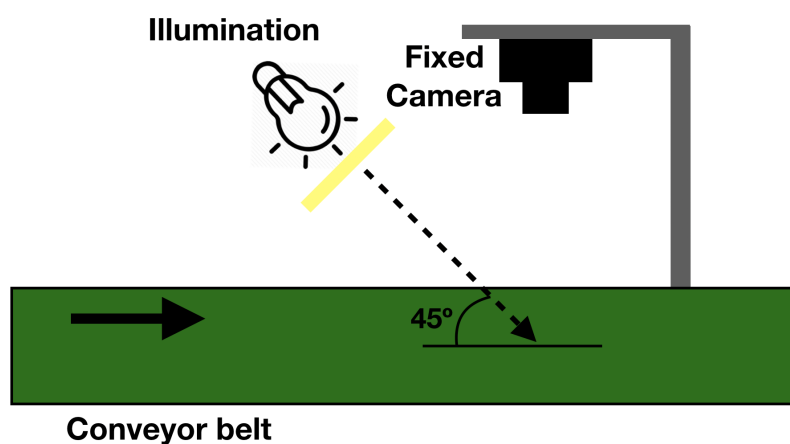


Figure 3.10: The sketch of *PPIC-D* with one camera on top, and one illumination source with 45° angle towards the surface.

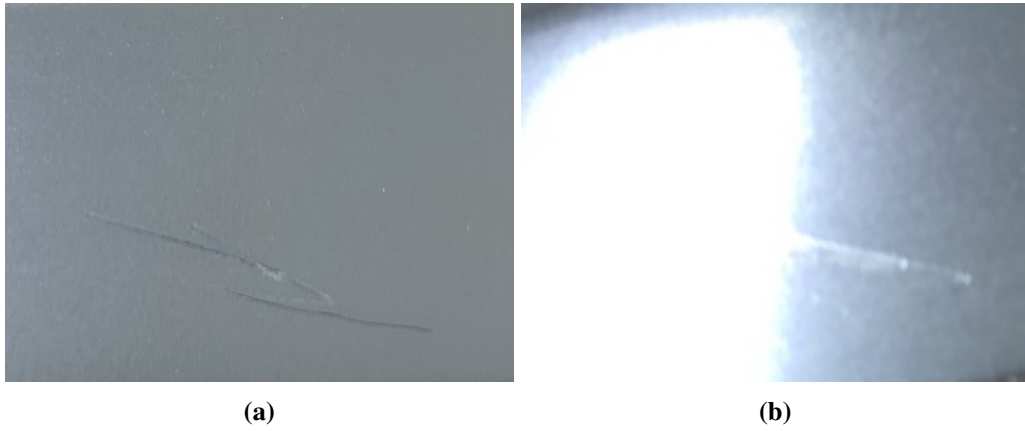


Figure 3.11: Image of damages on a surface a using normal ambient lighting (a) and using *PPIC-D* with reduced reflection (b).

Chapter 4

Damage Identification

Many studies have been conducted to detect the defect and damages for both industrial and non-industrial purposes. Chen et al. present a method to find damages on steel plates and use an industrial robot to polish the damaged areas [Chen et al., 2019]. They perform binary thresholding with a fixed threshold value, which makes the results deterministic for their special setup, but also makes the system vulnerable to small changes in light settings because the fixed threshold cannot adapt to changes in the lighting conditions. Similar thresholding techniques have been used in other studies [Senthikumar et al., 2014] [Chen and Deng,].

In addition to thresholding methods, convolutional neural network (CNN) approaches have been widely used to detect the damages. Masci et al. presented a convolutional neural network (CNN) approach for supervised steel defect classification in the steel strip market [Masci et al., 2012]. They classified 7 different defect types collected from a real production line, with an error rate of 7%, which outperformed a support vector machine (SVM) approach with various feature extractors. Park et al. also used a CNN for automatic surface defect inspection with an accuracy near to the human inspection [Park et al., 2016]. Cha et al. performed an automated visual inspection on civil infrastructures to increase safety [Cha et al., 2018]. A Faster R-CNN was trained to identify objects with the following classes: medium steel corrosion, steel delamination, high steel corrosion, concrete cracks and bolt corrosion. The first features of the input image were extracted with a CNN, the object/region proposals were made from these features, and finally the proposals were classified and the bounding box was fitted using a regressor.

Giben et al. described an approach to automatically monitor railway tracks to ensure passenger safety [Giben et al., 2015]. Cameras mounted on a wagon generate large volumes of images for visual inspection. The visual inspection was automated with a CNN for semantic segmentation. It could distinguish between the material categories ballast, wood, rough concrete, medium concrete, smooth

concrete, crumbling concrete, chipped concrete, lubricator, rail, and fastener. With the resulting pixel-wise segmentation they could identify damages and also evaluate the number of damages occurred. Semantic segmentation has also been used to detect defects in fabrics, fruits, and asphalt on roads [Latorella and Prabhu, 2000] [Li et al., 2002] [Sudakov et al., 2008].

Ferguson et al. et al. trained a Mask R-CNN on the GDXray dataset [Mery et al., 2015] containing X-ray images with annotated casting defects [Ferguson et al., 2018]. In another study, the authors presented an approach to detect damages for product printings. It is worth noting that the algorithm is a combination of raw pixel comparison of neighboring pixels and edge detection [Yangping et al., 2018]. First, the edges are identified using a Laplacian edge detector, using the first and second derivative in one dimension finding contrast changes. With a threshold, a binary mask of edges is created. The difference in the binary mask, as well as the pixel difference in non-edge patches of pixels in the image for the input is compared to the reference image. A threshold determines if there is a defect.

Baur et al. employed multiple variants of autoencoder architectures to detect anomalies from magnetic resonance images [Baur et al., 2018]. The general idea is to train the autoencoder to reconstruct the input and detect anomalies by subtracting the reconstruction from the input. Unseen data cannot be reconstructed by the autoencoder because it can lead to a higher reconstruction error. Furthermore, the segmentation is achieved by thresholding the reconstruction error of the autoencoder [Baur et al., 2018]. The authors used normal autoencoders, variational autoencoders, and variational autoencoders with the decoder part trained like an AnoGAN [Schlegl et al., 2017].

These studies have inspired the methods used in this chapter, in particular to validate H_1 . Section 4.1 describes the damage detection of fasteners, in which we compare a set of damage detection models for fasteners using different approaches, which will be used in Section 8.1. Section 4.2 describes the damage detection on industrial surfaces. These algorithms will be used to in Section 8.2¹.

4.1 Damage Detection of Fasteners

As explained in Chapter 1, a component may consist of multiple fasteners and also other components (see Figure 1.5). This section presents the damage identification methods and results for the fasteners. Parts of the presented work in this section have been previously published [Taheritanjani et al., 2019b] or is extracted from a study conducted at the chair for applied software engineering at

¹Parts of this chapter have been previously published [Taheritanjani et al., 2019b]

TU Munich [Schönfeld, 2018]. Subsection 4.1.1 describes the dataset properties. Subsection 4.1.2 explains the approaches and methods that are used to conduct this study. Finally, Subsection 4.1.3 presents the results and discusses the implications.

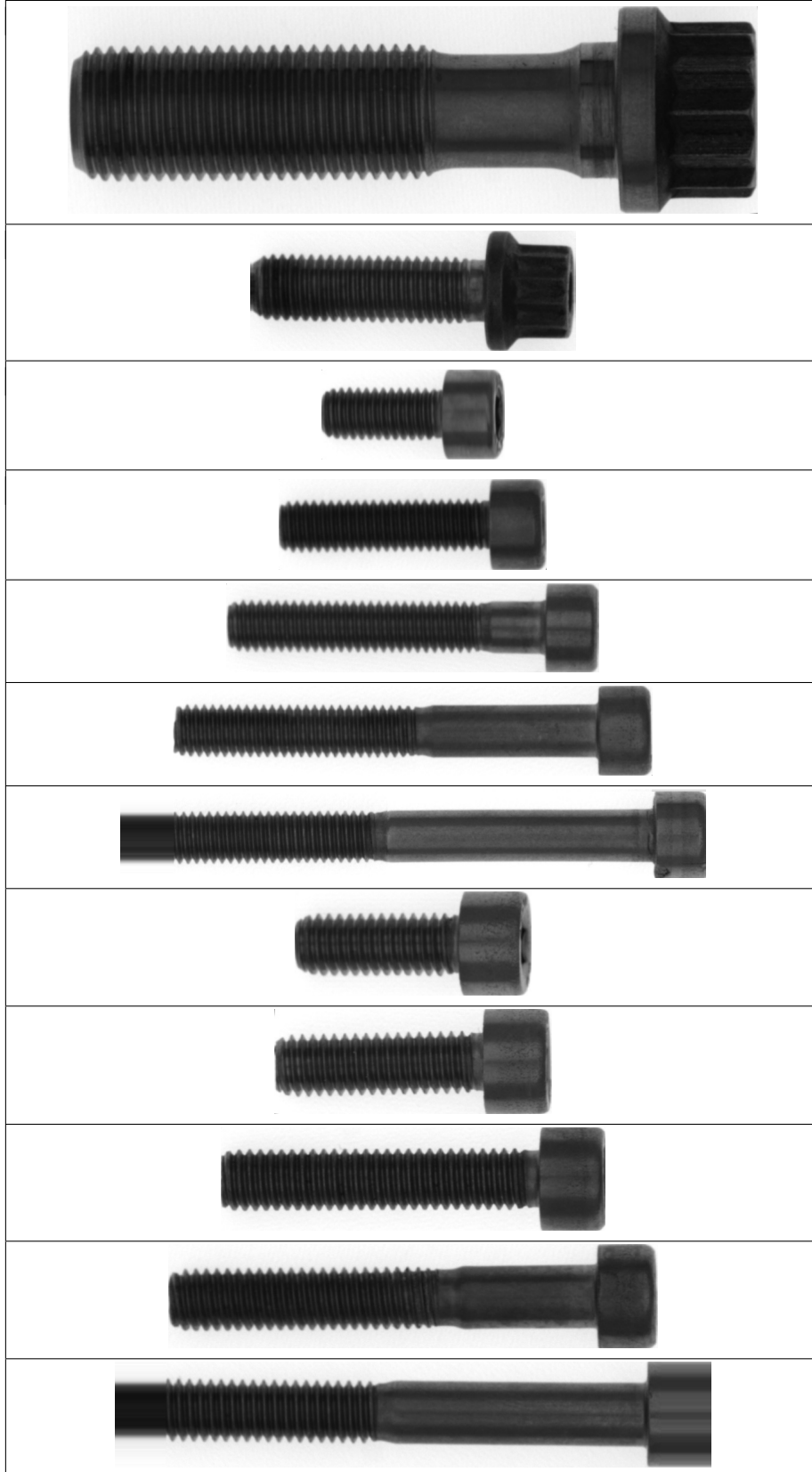
4.1.1 Dataset Collection

Since there is no dataset available that focuses on damages of fasteners, we used *PPIC-B*, described in Section 3.2.1 to collect the dataset in this study. The fasteners dataset consists of 2019 images of 12 different bolt types each with intact and damaged samples. Table 4.1 shows examples of these different types of fasteners. The bolts of type AS 31532 and AS 21514 are specific airplane engine fasteners. The other bolts M5 12, M5 20, M5 30, M5 40, M5 50, M6 16, M6 20, M6 30, M6 40, and M6 50 are standard bolts. They have metric ISO-threads (M) with a diameter of 5 and 6 millimeters and a length ranging from 12 to 50 millimeters. The dataset consists of 4 different instances per status of each fastener for the standard types. The AS 31532 bolt is represented with 2 damaged and 2 intact instances while the AS 21514 is presented in the dataset with 1 damaged instance and 2 intact instances. In total the dataset consists of 87 instances of fasteners.

To ensure that the model generalizes to the all kinds of damages and not specific instances of them, the model is tested on a holdout set of damaged and intact instances that it has not seen before. For each of the standard types an instance of an intact and an additional damaged fastener - not in the training dataset - is used in the holdout set. The test dataset consists of 207 pictures of damaged fasteners and 213 intact fasteners. The rest of the images are used for training and validation, with a random split of 75% for training and 25% for validation.

For the instance segmentation of the damages, we manually created the fine-grained annotations. The instances of damages are annotated with a mask marking the damaged area. In total, we annotated 373 images of the fasteners with 2438 masks defining regions of damages (2065 masks) and the area of the fasteners (373 masks). The instance segmentation dataset is also split into a training dataset with 248 images and 1720 annotations as well as a validation dataset with 125 images and 718 annotations.

In addition, we preprocess the recorded images to improve the detection performance of the damage detection algorithms. To separate the captured image from the background, we apply either a simple threshold for a uniform background, i.e., using a binary threshold for the background color in OpenCV, or an edge detection for a non-uniform background. In edge detection step, we blur the image using Gaussian Blur in OpenCV for noise robustness. Afterwards, using the Sobel filter in OpenCV, we apply a filter to detect edges. From the resulting pixel, the threshold mask or the edges of the outer contours are identified. To locate the fastener, we use the minimum area rectangle of the outer contour. Finally,

Table 4.1: Fasteners used in the dataset.

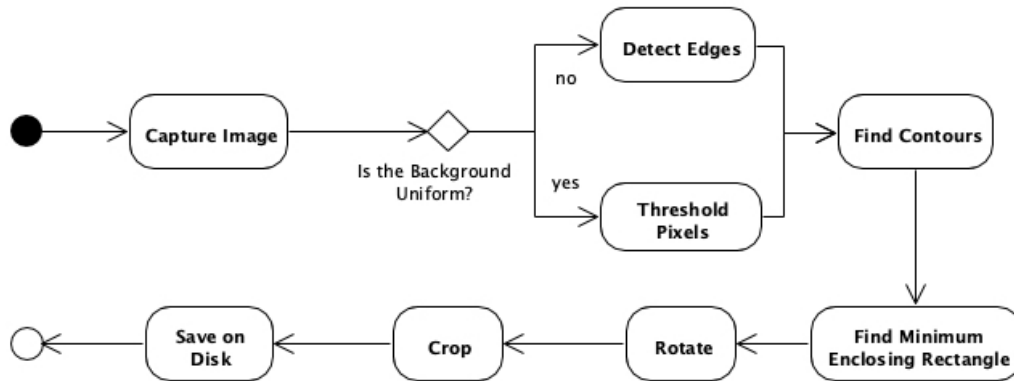


Figure 4.1: The preprocessing workflow (UML activity diagram).

the image of the fastener is rotated and cropped and the final result is a rectangular image just fitting the fastener. Figure 4.1 shows the preprocessing workflow.

4.1.2 Methods

Considering the challenges faced in the fine-grained visual categorization of fasteners [Taheritanjani et al., 2019a], perspective, reflection, and shadow with normal ambient lighting as well as the shape, threads, and head features on which we want to focus, we describe a set of methods to preprocess the data, create damages, and train the models.

We trained different machine learning models (both supervised and unsupervised) to detect the damages. Supervised tasks are the classification of the type of the fastener and the instance segmentation of damages, i.e., they determine if a fastener is intact or damaged. The unsupervised task is anomaly detection with anomalies representing damages. Figure 4.2 shows an overview of the supervised and unsupervised approaches in this study.

We also used the *Blackboard* pattern to combine multiple interdependent damage detection algorithms. *Blackboard* stores solutions and intermediate results that are accessed by the knowledge sources. Knowledge sources query the *Blackboard* for information that is provided by other knowledge sources. Table 4.2 shows the knowledge sources.

4.1.3 Results and Discussion

The results are divided into supervised and unsupervised approaches. Supervised approaches are the classification of the type of the fastener and the instance segmentation of damages, i.e., if a fastener is intact or damaged, while the unsupervised approaches are based on anomaly detection.

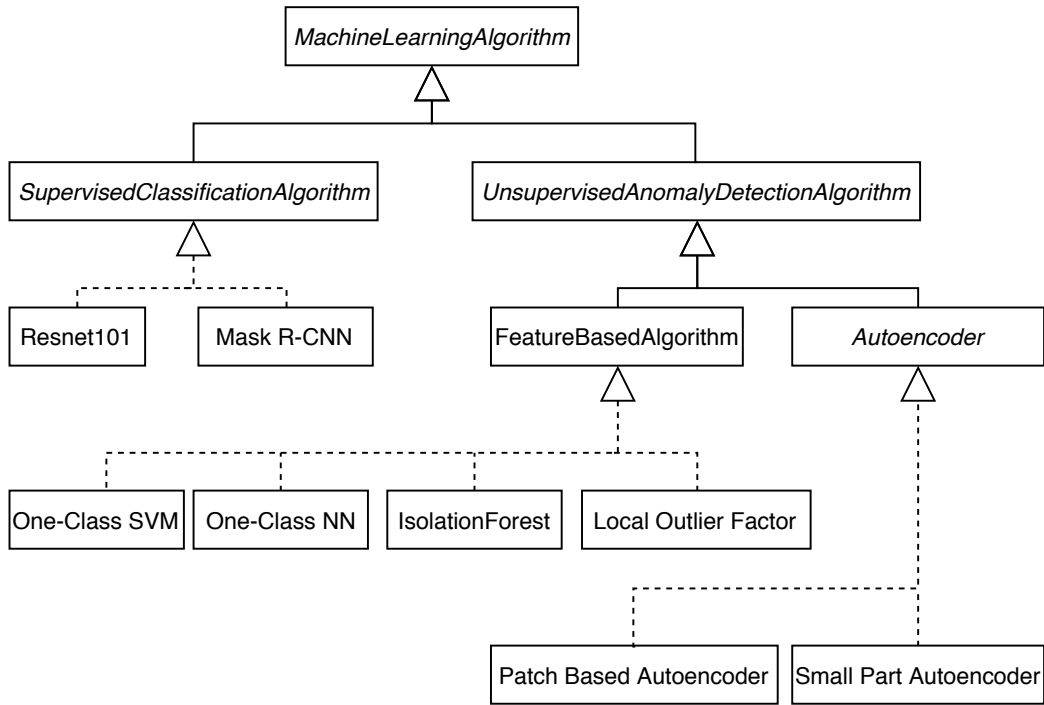


Figure 4.2: The overview of the algorithms which are used in this study [Taheritanjani et al., 2019b].

The first supervised approach is the classification that determines if a fastener is damaged or intact - damaged/intact classification. Using a Resnet101, all damaged and intact fasteners are correctly classified for the training dataset. In the test dataset, consisting of the damages the model has not seen before, the performance drops slightly. 99% of the damaged fasteners are correctly identified as damaged as well as 99% of the intact fasteners are correctly classified as intact.

The Mask R-CNN trained on the detection of damage instances achieved an mAP at an IoU of 50 of 0.5563 for the task of object detection on the validation dataset with fine-grained annotations. Combining the results from Mask R-CNN with simple rules (damaged area covers more than 1% of the whole image and number of damaged objects is greater than zero) can also output a damage detection model. Table 4.3 shows the performance metrics for the supervised classification algorithms. The rules have the same precision with lower recall values, due to a lower number of true positives. Combining the Resnet101 result with Mask R-CNN (considers a fastener to be damaged if either the Resnet101 or the Mask R-CNN determines it to be damaged) decreases the true negatives and precision but increases the generalizability since damage objects of unknown fasteners can also be identified.

Table 4.2: Knowledge sources that are used for damage detection with their input and output sources. Feature Vector: lower dimensional representation, Area Damaged: percentage of the fastener which is damaged, Numeric Anomaly: numeric value representing how anomalous a fastener is according to a damage detection algorithm, Model Output: output of a model determining whether the fastener is damaged or intact, Result: output of the blackboard representing the final decision.

Knowledge Source	Input	Output	Algorithm
TypeClassificationKS	Input Image	Type, Feature Vector	Resnet101
StatusClassificationKS	Input Image	Model Output, Feature Vector	Resnet101
InstanceSegmentationKS	Input Image	Number of Damages, Area Damaged	Mask R-CNN
PatchAutoencoderKS	Input Image	Numeric Anomaly	Patch-based Autoencoder
FastenerAutoencoderKS	Input Image	Numeric Anomaly, Feature Vector	Autoencoder
OneClassSVMKS	Feature Vector, Type	Model Output	One-class SVM
IsolationForestKS	Feature Vector, Type	Model Output	Isolation Forest
LocalOutlierFactorKS	Feature Vector, Type	Model Output	Local Outlier Factor
OneClassNeuralNetworkKS	Feature Vector, Type	Model Output	One-class Neural Network
NumberOfDamagesRuleKS	Number of Damages	Model Output	Rule-based
AreaDamagedRuleKS	Area Damaged	Model Output	Rule-based
ReconstructionErrorRuleKS	Numeric Anomaly, Type	Model Output	Rule-based
EnsembleKS	Model Output	Result	Ensemble

Figure 4.3 visualizes the Resnet101 output for a sample damaged fastener using Grad-CAM [Selvaraju et al., 2017] and Figure 4.4 shows an example of the Mask R-CNN result.

The unsupervised approaches are anomaly detection with anomalies that represent damages. Table 4.4 shows the performance metrics for the unsupervised anomaly detection methods. The anomaly detection methods operate on different extracted features from the image (Resnet101 features and autoencoder bottleneck features) and apply one-class SVMs, one-class neural networks [Chalapathy et al., 2018], isolation forests, and local outlier factors to determine if the set of features is abnormal. Rules based on the reconstruction error of autoencoders (patch-based and for the whole fastener) trained on images of intact fasteners use a threshold to determine whether a fastener is considered to be damaged. The patch-based au-

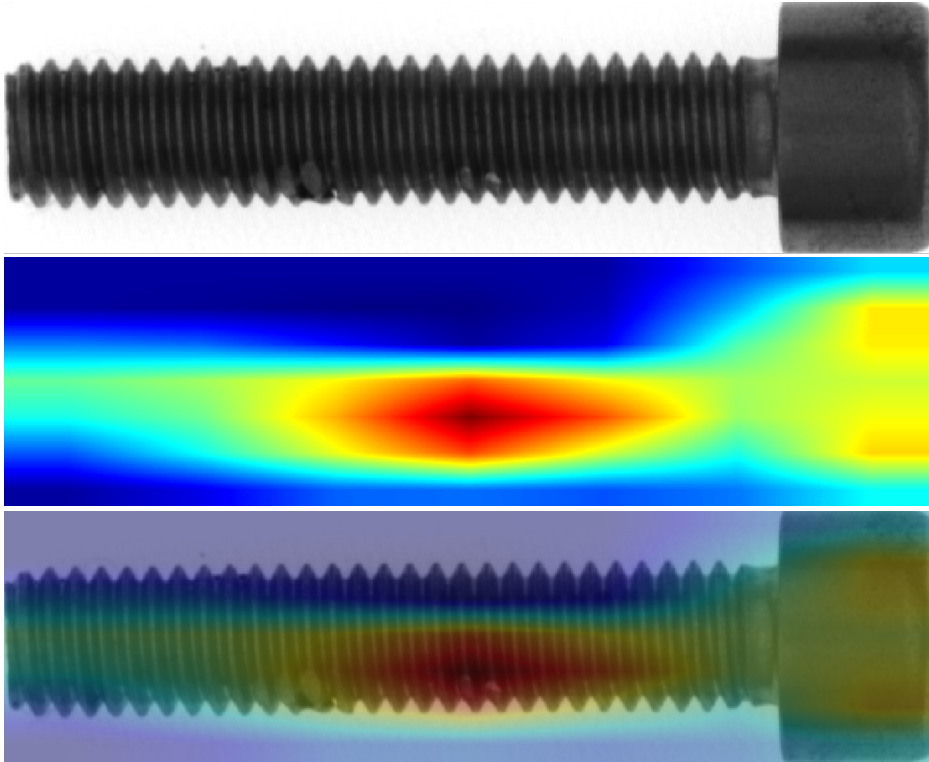


Figure 4.3: An example of the heatmap visualization for a damaged fastener. Low activation is shown with cold colors (namely blue). The colder area indicates less anomalies in that part of the image [Taheritanjani et al., 2019b].



Figure 4.4: An example of Mask R-CNN result for the damages including their masks and bounding boxes [Taheritanjani et al., 2019b].

Table 4.3: Supervised classification performance results on test dataset

Algorithm	Accuracy	Precision	Recall	Training Details
Resnet101	0.99	0.99	0.99	epochs=8, batch_size=8, lr=0.001
Mask R-CNN (Number of Damages Rule)	0.743	0.99	0.483	epochs=32, batch_size=2, lr=0.001
Mask R-CNN (1% Area Rule)	0.743	0.99	0.483	epochs=32, batch_size=2, lr=0.001
Mask R-CNN + Resnet101	0.988	0.986	0.99	-

toencoder rule has the highest accuracy among the unsupervised approaches with an accuracy of 84%, the true negatives of 98%, precision 97%, and F1-Score of 77%.

Figure 4.5a and Figure 4.5b show the output of the patch based autoencoder to the corresponding input as well as the reconstruction error (absolute value for subtracting the reconstruction from the input image) and the squared reconstruction error. The squared error reduces the noise and visualizes only larger damages. The patch-based autoencoder does the reconstruction on small image patches identifying errors in the texture of the fasteners.

The results for the damaged/intact classification, with the precision and recall of 99%, also imply a possible usage in the overhaul process. The mAP at an IoU of 50 of 0.5563 for the damage object detection is comparable to other object detectors on the COCO dataset [Redmon and Farhadi, 2018]. With the specificity of 0.995 and precision of 0.99, the instance segmentation can complement other damage detectors and provide visual insights about the damages. For a complete evaluation, the damaged/intact classification must be scaled with more samples recorded with an automated system, allowing automating the whole classification and sorting process in an overhaul plant. To compare, Table 4.5 shows the comparison between the fastener damage identification approaches used in this dissertation and state of the art results with regards to the rating scheme, described in Section 1.2.

Without the knowledge of damages, the task of unsupervised anomaly detection on images can be challenging. The distinction between damaged and intact fasteners is often fine-grained and affected by noise, light, the position of the fastener or other minor changes. The combination of feature extraction with one-class classifiers did not lead to satisfying results. We consider that the extracted features, especially from the autoencoders, cannot represent the damages very well. We can determine the notion of normality using autoencoders. However, their performance is still lower than the supervised methods - with an accuracy of

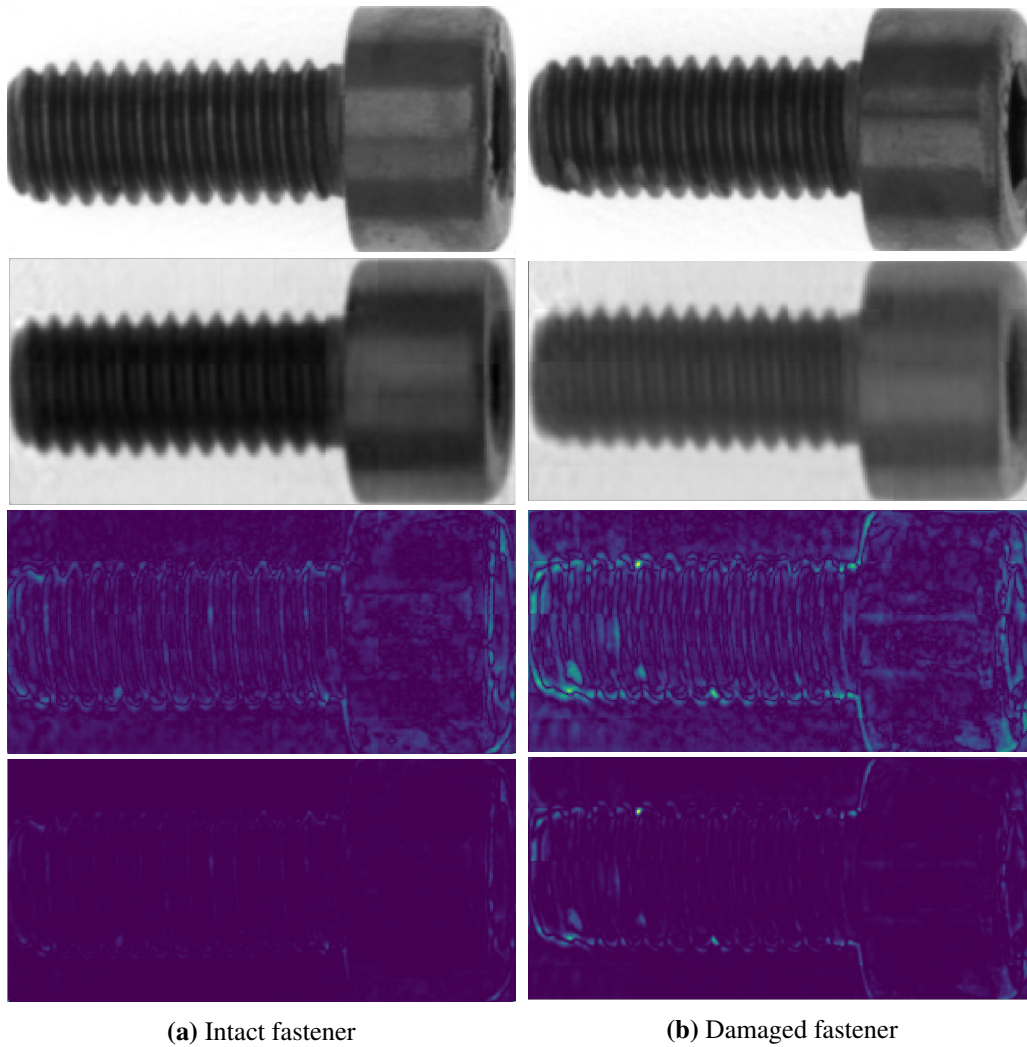


Figure 4.5: An example of the autoencoder input (first row), its reconstruction (second row), reconstruction error (third row), and square of reconstruction error (last row) for an intact and a damaged fastener [Taheritanjani et al., 2019b].

Table 4.4: Unsupervised classification performance results on test dataset.

Algorithm	Feature Extractor	Accuracy	Precision	Recall	Training Details
One-class Support Vector Machines	ResNet101	0.63	0.59	0.82	kernel="rbf", nu=0.1, gamma=1/8192
One-class Support Vector Machines	Autoencoder	0.55	0.57	0.35	kernel="rbf", nu=0.01, gamma=1/8192
Isolation Forests	ResNet101	0.63	0.59	0.83	n_estimators=100, bootstrap=False
Isolation Forests	Autoencoder	0.55	0.57	0.35	n_estimators=100, bootstrap=False
Local outlier Factors	ResNet101	0.66	0.60	0.89	-
Local outlier Factors	Autoencoder	0.53	0.54	0.24	-
One-class Neural Networks	ResNet101	0.65	0.70	0.49	epochs=50, hidden_nodes_number=32
One-class Neural Networks	Autoencoder	0.50	0.45	0.04	epochs=50, hidden_nodes_number=32
Autoencoder Reconstruction Rule	-	0.78	0.96	0.57	epochs=40, batch_size=4, lr=0.00005
Patch Autoencoder Reconstruction Rule	-	0.84	0.97	0.64	epochs=40, batch_size=4, lr=0.00005

84% and a recall of 64% for the patch-based autoencoder paired with a rule for the reconstruction error, which could be due to the distinguishing of the concept of normality from everything else. However, a visualization of the reconstruction error provides valuable insights for human operators showing where exactly the damage might be located.

The drawback of the supervised approaches is that we need to train the models with both intact and damaged instances of the fasteners. While they offer higher accuracy, the efforts for the dataset collection and the preprocessing steps are also high and the generalizability could be affected with introducing new unseen damages. However, unsupervised approaches need only the intact instances for the training phase that can reduce the efforts for the dataset collection and the preprocessing steps.

To improve the results and the generalization of the solutions, more data and

Table 4.5: Comparison between the fastener damage identification approaches used in this dissertation and the state of the art results in damage identification with regards to the rating scheme.

Study	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆
[Ferguson et al., 2018]	±	+	+	-	-	+
[Cha et al., 2018]	±	+	±	±	-	+
[Yangping et al., 2018]	-	+	±	-	+	±
[Baur et al., 2018]	±	+	+	+	+	not provided
[Wei et al., 2018]	±	+	+	±	+	±
Fastener damage identification approaches used in this dissertation	+	+	±	+	+ (unsupervised) ± (supervised)	± (unsupervised) + (supervised)

more different damages are needed. Furthermore, since the noise is an extra anomaly, constraining the data collection environment and reducing the noise can also increase the accuracy of the results. Moreover, the current images of the fasteners are from one top-down perspective. To detect the damages on all sides of the fasteners, they must automatically be rotated using vibration or another setup that uses multi-view cameras. In addition, the current dataset must be scaled to a larger number of classes. Some damages are not detectable optically and, therefore, cannot be identified using computer vision. To detect these types of damages, we can use for example other sensors and methods [Michaels, 2008] [Fugate et al., 2001] [Ciang et al., 2008].

Our dataset might not represent all the classes in the target environment. Therefore, to ensure consistent performance, more data preferably with more classes have to be recorded. Another threat to validity is the damages of the fastener in the dataset that might not represent all the cases of real damaged fasteners. To solve this issue, the data has to be collected at the target location. Currently, images are only recorded with one specific type of camera at a fixed distance. Using image processing and passing the distance/scale to the models can remediate this limitation.

4.2 Surface Damage Detection

This section presents the surface damage identification methods. Parts of the presented work in this section have been extracted from an ongoing study at chair for applied software engineering at TU Munich [Vinzencz, 2020]. Subsection 4.2.1 describes the dataset properties. Subsection 4.2.2 explains the approaches and methods that are used to conduct this study. Subsection 4.2.3 describes the evaluation and presents the results. Finally, Subsection 4.2.4 discusses the findings and

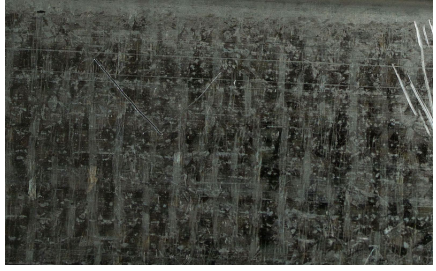
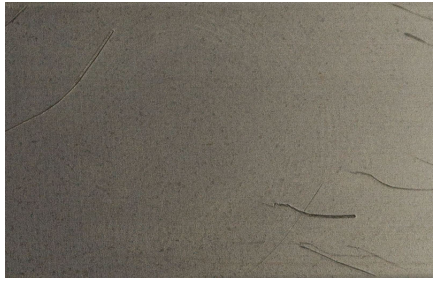
implications.

4.2.1 Data Collection

There are available datasets for surface damage detection. Severstal: Steel Defect Detection² dataset has images of steel surfaces with their encoded damage pixels in a csv file. Kolektor Surface-Defect Dataset [Tabernik et al., 2019] has images of microscopic fractions or cracks on the surface of the plastic embedding in electrical commutators annotated with bitmap labels. NEU surface defect database [Song and Yan, 2013] has 6 kinds of surface damages on the hot-rolled steel strip which are annotated in PascalVOC XML format. However, they are either from one single type of material or lack insufficient images from damaged or intact classes (see Table C.2 for a better comparison between the datasets).

Therefore, we captured the dataset using *PPIC-D* from two different materials. Table 4.6 shows the materials used in data collection.

Table 4.6: Surfaces used in the dataset (102 images with 15 damages per image on average and their fine-grained annotations and 355 images of the intact surfaces for novelty detection).

Name	Sample Image
Galvanized Metal	
Aluminum Metal	

We captured 65 images of galvanized metal and 37 images of aluminum metal. All the images were annotated using VGG Image Annotator (VIA) Tool, in order to have the ground truth bounding boxes of the damages for the evaluation. Each

²<https://www.kaggle.com/c/severstal-steel-defect-detection/>

damaged image contained at least 3 to 37 damage annotations. The whole dataset had 1572 annotated areas, having an average of 15.4 damaged areas per image. The size of annotated damages ranges from 1mm to more than 20 mm.

Moreover, to use anomaly detection and novelty detection algorithms, images of intact samples of the surfaces were captured. We collected 161 images of galvanized metal and 194 images of aluminum metal. These images formed the second dataset (intact dataset) and used later for training phase of anomaly detection algorithms.

4.2.2 Methods

We applied three different methods to identify damages on the surfaces. Subsection 4.2.2 presents the thresholding method. Subsection 4.2.2 explains the supervised image segmentation approach using Mask R-CNN, and Subsection 4.2.2 presents unsupervised anomaly detection method using autoencoders.

Thresholding

Thresholding technique is not required to split the dataset for training. Nevertheless, we used set of images for testing the performance of the algorithm. The first thresholding technique is Gaussian Otsu. It does not require any threshold parameter to be selected; the algorithm finds the best global threshold for binarization. However, some parameters must be provided, such as blur kernel or size of the bounding box of the detected damage. The algorithm starts with reading the image and applying median blur, followed by a Gaussian blur filter. We set the median blur kernel to 5×5 and the Gaussian kernel size to 3×3 . The Gaussian kernel blurs the edges and reduces contrast, which results in removing noise in Otsu thresholding. Finally, next to applying the thresholding technique, a minimum and maximum area check must be provided. The detected damage bounding box must be smaller than one third of the whole image and bigger than 60 pixels.

Supervised Image Segmentation

For Mask R-CNN, we split the data into train, validation and test datasets. We used 80% of the data for training, 8% for validation, and 12% for test (82 images in train, 8 images in validation and 12 images in the test dataset). We also applied a stratified split with respect to the distribution of images per class to ensure that each of the train, test, and validation contain the same proportion of the classes.

In addition, we applied image augmentation to regularize the training. We used the following augmentations:

- **Flipping**: we applied horizontal and vertical flipping to 50% of the dataset.

- **Color Jitter:** images recorded with small changes in ambient light, and dirty small parts can cause large differences in appearance. To avoid the network to overfit to the lighting conditions, or recognize dirty small parts which can appear darker, we applied random brightness and contrast augmentations. The brightness and contrast are randomly altered up to $\pm 25\%$ for each image.
- **Random Rotation:** we rotated the image to artificially increase the different rotations a nut can lay in. Each image is randomly rotated between $\pm 180^\circ$.

Finally, we trained a Mask R-CNN model to automatically segment the damages in the images. Table 4.7 summarizes the training parameters.

Table 4.7: Parameters and settings used to train Mask R-CNN model, pretrained with COCO dataset.

# of epochs	Learning Rate	Batch Size	Optimizer	Backbone
10 (heads)	0.001(epochs 1 to 30)	2	SGD(momentum=0.9, weight_decay=0.0001)	ResNet50
80 (all)	0.0001(epochs 31 to 60)			
	0.00001(epochs 61 to 90)			

Unsupervised Anomaly Detection

To apply unsupervised anomaly detection for the surfaces, we used autoencoders. An autoencoder tries to learn the most important features of the input by first encoding the input to lower dimensionality (encoding) and then reconstruct the original input from the encoded features (decoding).

We trained a stacked convolutional autoencoder using the intact dataset. The autoencoder consisted of five convolutional layers in encoder and five convolutional layers in decoder. Table 4.8 shows the architecture of the autoencoder and the training hyper parameters.

Table 4.8: Parameters and settings used to train autoencoder for anomaly detection.

# of epochs	Learning Rate	Batch Size	Optimizer	Architecture
20	1	8	SGD	4 convolutional layers with 64, 128, 128, and 256 filters, each with kernel_size=(3, 3) and activation= 'selu'

4.2.3 Evaluation and Results

We tested the methods described in Section 4.2.2 on the test datasets for the segmentation results with the mean average intersection of unit (*IoU*) metric. We used 12 damaged images in Mask R-CNN test dataset together with 12 intact images to test the methods for binary classification (damaged vs. intact). Table 4.9 shows the performance metrics for the different methods. In addition to the accuracy, we also used precision and recall metrics.

Table 4.9: The performance results on test dataset.

Method	Segmentation IoU	Classification Accuracy	Classification Precision	Classification Recall
Thresholding	23.5%	41.7%	41.7%	41.7%
Supervised Segmentation (Mask R-CNN)	48.1%	95.83%	100%	91.7%
Anomaly Detection (Autoencoder)	42.7%	83.3%	78.6%	91.7%

Figures 4.6 and 4.7 shows examples of the results obtained by different methods on both of the materials.

4.2.4 Discussion

The thresholding algorithm obtained an AP at an IoU of 50 of 0.235 for the task of damage detection on the test dataset with fine-grained annotations. Using this algorithm for binary classification (damaged area covers more than 1% of the whole image and number of damaged objects is greater than zero) resulted in 41.7% accuracy, precision and recall. These results were used as a baseline for comparison of the methods.

The Mask R-CNN trained on the detection of damage instances achieved an AP at an IoU of 50 of 0.481 for the task of damage detection on the test dataset with fine-grained annotations. Combining the results from Mask R-CNN with binary rules (damaged area covers more than 1% of the whole image and number of damaged objects is greater than zero) can also output a damage detection model.

The results for the damaged/intact classification, with the precision of 100% and recall of 91.7%, also imply a possible usage in surface damage detection in overhaul process. The AP at an IoU of 50 of 0.481 for the damage object detection is slightly less than other object detectors on the COCO dataset [Redmon and Farhadi, 2018]. The instance segmentation can complement other damage detectors and provide visual insights about the damages. For a complete evaluation, the damaged/intact classification must be scaled with more samples recorded with

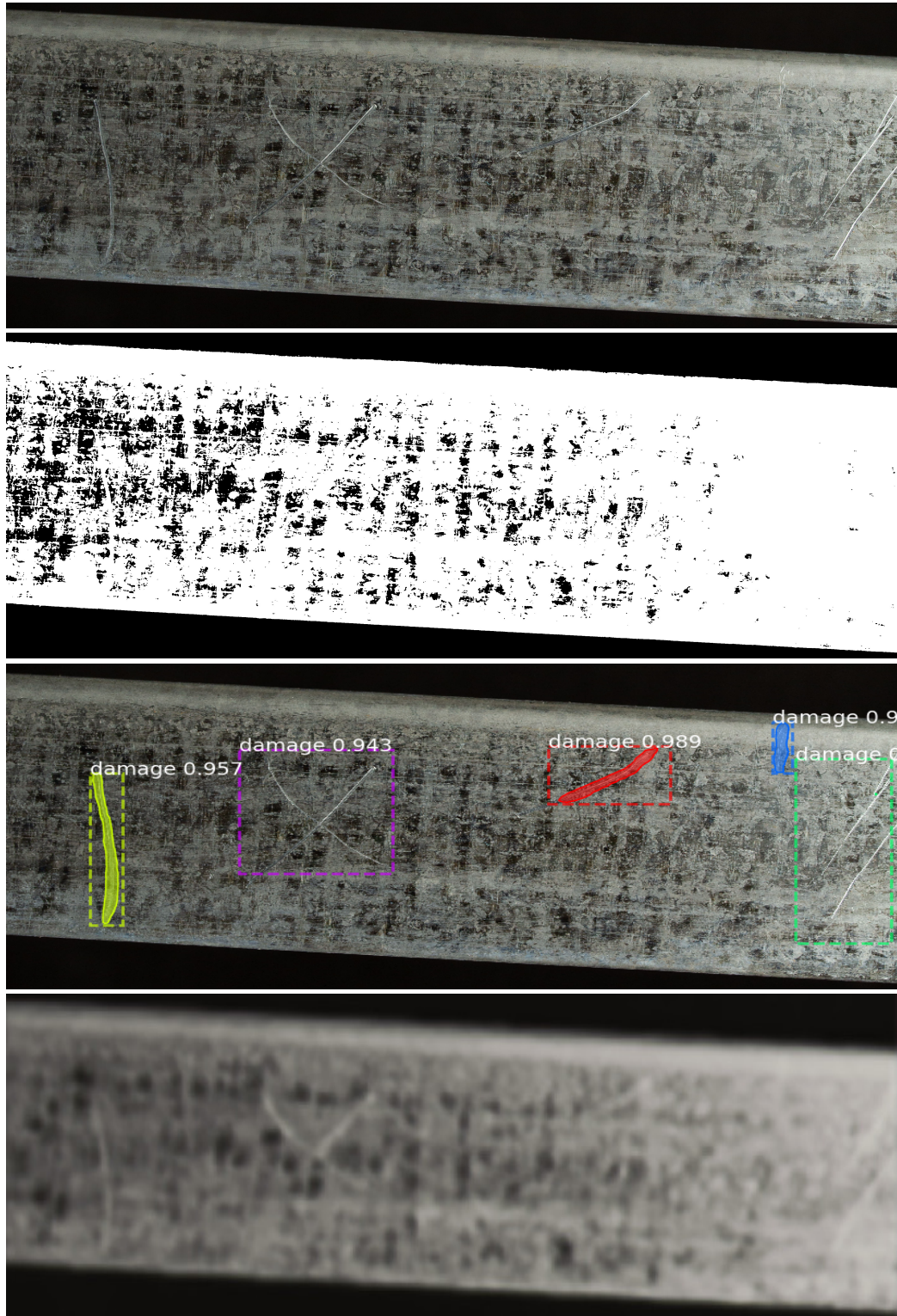


Figure 4.6: An example of the original image input (first row), thresholding result (second row), segmentation result using Mask R-CNN (third row), and its autoencoder reconstruction (last row) for galvanized surfaces.



Figure 4.7: An example of the original image input (first row), thresholding result (second row), segmentation result using Mask R-CNN (third row), and its autoencoder reconstruction (last row) for aluminum surfaces.

an automated system, allowing automating the whole classification and sorting process in an overhaul plant.

As described earlier in Section 4.1.3, without the knowledge of damages, the task of unsupervised anomaly detection on images can be challenging. The distinction between damaged and intact surfaces is often fine-grained and affected by noise, and the type of surface material. The extracted features from the autoencoders cannot represent the damages very well. Although, we can determine the notion of normality using autoencoders, their performance is still lower than the supervised methods - with an accuracy of 83.3% and a precision of 78.6% for the autoencoder paired with a rule for the reconstruction error, which could be due to the distinguishing of the concept of normality from everything else. However, a visualization of the reconstruction error provides valuable insights for human operators showing where exactly the damage might be located.

It is also worth to mention that the methods usefulness is varied on different types of materials. Damages and cracks on aluminum material are easier to spot by thresholding and autoencoder approaches, due to the uniform color and texture of the material. The damages on the galvanized material, however, are best detected by Mask R-CNN approach. Galvanized metal has a scrambled texture, which makes it challenging to distinguish the cracks versus the material itself.

The drawback of the supervised approaches is that we require to train the models with many instances of damaged surfaces. While they offer higher accuracy, the efforts for the dataset collection and the preprocessing steps are also high and the generalizability could be affected with introducing new unseen damages. However, unsupervised approaches require only the intact instances for the training phase that can reduce the efforts for the dataset collection and the preprocessing steps.

To improve the results and the generalization of the solutions, more data and more different damages are needed. Furthermore, the current images of the surfaces are from one top-down perspective. To detect the damages on all sides of the surfaces, they must automatically be rotated or other setups that use multi-view cameras must be employed. In addition, the current dataset must be scaled to a larger number of classes. Some damages are not detectable optically and, therefore, cannot be identified using computer vision. To detect these types of damages, other sensors and methods can be used [Michaels, 2008] [Fugate et al., 2001] [Ciang et al., 2008].

We only used aluminum and galvanized metals in the dataset, which do not represent all the classes in the target environment. Therefore, to ensure consistent performance, more data preferably with more types of materials must be recorded. Another threat to validity is the damages on the surfaces in the dataset that might not represent all the cases of real damaged fasteners. To solve this issue, the data must be collected at the target location. Currently, images are only recorded

with one specific type of camera at a fixed distance. Using image processing and passing the distance/scale to the models can remediate this limitation.

Chapter 5

Sorting the Fasteners based on their Similarity

Deep learning applications require large amount of data. When there are only a few data samples, the task is usually more complicated and challenging. One-shot learning is an example of this situation, when there are not many data samples; the prediction is based on only one single example of each class. Koch et al. described a method by using a siamese neural network, which has a unique structure to rank similarity between inputs naturally [Koch et al., 2015]. The siamese networks are capable of learning generic image features for making predictions on unknown class distributions. They provide a competitive approach which does not rely on domain-specific knowledge by exploiting deep learning techniques. Their model can predict on new data and generalize the predictive power to entirely new classes from unknown distributions based on the powerful discriminative features extracted.

In another study, Wang et al. proposed a deep ranking model to learn fine-grained image similarity [Wang et al., 2014]. The model characterizes the fine-grained image similarity relationship with a set of triplets. A triplet is a group of images including a query image, a positive image and a negative image. The positive image is similar to the query image, while the negative image is not the same class. The order in the triplets represents image similarity. The authors used a bootstrapping method to generate an unlimited amount of training data virtually. Due to the intrinsic difference between image classification and similarity detection tasks, the classic network for image classification is not optimal for calculating the similarity between images. Therefore, Wang et al. proposed a multi-scale network structure which contains the convolutional neural network with two low-resolution paths. The results outperformed the hand-crafted visual feature-based approaches and deep classification [Taylor et al., 2011] [Krizhevsky et al., 2012].

Bertinetto et al. generated an image patch similarity function to learn from annotated pairs of raw image patches. To train the networks, they use an extensive database that contains pairs of raw image patches, which can be matched or not matched. They can enrich this database with more samples to improve the performance of the models. After applying their approach on several problems and benchmark datasets, their methods lead to feature descriptors with better performance than manually designed descriptors (e.g., SIFT, DAISY) and other learned descriptors [Simonyan et al., 2014].

This section explains the clustering of fasteners based on their similarity, where we sort bolts using a single camera view from *PPIC-A*. Parts of the presented work in this section have been extracted from a study conducted at the chair for applied software engineering at TU Munich [Yu, 2020]. Section 5.1 describes the dataset properties. Section 5.2 explains how we trained and evaluated the models. Finally, Section 5.3 shows the results and discussion.

5.1 Data Collection

We collected the dataset using *PPIC-A*, described in Section 3.2.1. We used a HUAWEI Mate 20 cellphone camera to capture images, each with 3968x2976 pixels resolution. The camera shutter speed, aperture size and ISO were not fixed and the camera automatically set these configurations.





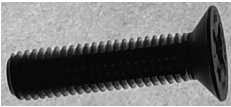


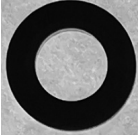











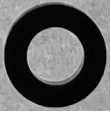
The dataset consists of 20 different classes fasteners. The fasteners were chosen in a way that they can be classified from a bird's eye view, i.e. the similar bolts with different heads are excluded from this study. The dataset contains a total of 1000 single-view images, 50 per class. Table 5.1 shows examples of fasteners in dataset.

The dataset was split into a train and test set during the training (see Section 5.3).

5.2 Training and Evaluation

We applied several data preprocessing methods prior to training the siamese network. First, we cropped the minimum enclosing rectangle of the fasteners and rotate them horizontally. Afterwards, we added the padding around the cropped images to have them in 1000×1000 pixels (which is less than the length of the longest edge of all the cropped images so that each image has the same size). We also converted all the images to grayscale. Afterwards, we resized the images from 1000×1000 to 200×200 . The resizing step improves the speed of computation, saves memory in GPU and also removes noisy high-level features which are not

Table 5.1: Fasteners used in the dataset (scaled to 0.5 of the original size).

important for the clustering.

In addition, we used the following three augmentations for the experiments:

- **Color Jitter:** Images are collected with small differences of brightness because of automatic camera settings. Moreover, the dirty or damaged small parts may cause differences in appearance. To address this issue, we applied random brightness augmentations. The brightness is randomly shifted up to $\pm 40\%$ for each image.
- **Rotation:** we rotated the images to four different angles, 0° , 90° , 180° and 270° . These 4 numbers must suffice to include all the possible angles, since we cropped the bounding boxes horizontally in data preprocessing steps. We extended the original dataset with the rotated images and obtained 200 images per class in the dataset.

Furthermore, all models are trained using the following parameters and the model with the highest validation accuracy was selected.

- **Initialization:** we initialized CNN layer weights with mean as 0 and standard deviation of 0.01, and CNN layer biases with mean as 0.5 and standard deviation of 0.01 [Koch et al., 2015].
- **Optimizer:** we selected standard SGD with a momentum of 0.9 and weight decay of 10^{-4} as the optimizer. Momentum and weight decay minimize the impact of noisy gradients and local minima, which usually results in training speed up and higher accuracy.
- **Loss Function:** we used standard cross-entropy loss for training.
- **Learning Rate:** after several experiments, we set the learning rate to 10^{-2} . We decided not to decrease the learning rate over time, because it did not improve the performance of the model in several experiments.
- **Batch Size:** although using a smaller batch size in each step results in faster execution of step, these steps usually are less accurate due to lower amount of data. After several experiments, we set the batch size to 20.
- **Epochs:** we used 40000 epochs (iterations) to reach a steady loss.

Table 5.2 shows an overview of the hyper-parameters and augmentations used during trainings. For evaluation of model performance, we measured the accuracy.

We could also consider training time as a metric. Training time is defined as the amount of time needed to adjust weights and bias of the model. We evaluated

Table 5.2: Overview of parameters and settings used to train models.

Optimizer	Iterations	Learning Rate	Batch Size	Augmentations
SGD (mmentum=0.9, weight_decay=0.0001)	40000	0.01	20	Random rotation to four different angles, 0°, 90°, 180° and 270°, random light intensity (color Jitter) between $\pm 40\%$

each trained model running on pair images from the test dataset that was randomly generated. Since the network sees the training samples in a random order, and the fully connected layers are randomly initialized, training models cannot have the exact same weights after a set number of epochs. Therefore, we trained each model independently three times. The reported results are an averaged testing accuracy over the three independent models.

5.3 Results and Discussion

Table 5.3 shows an overview of the training results for different configurations of the training and test. The results indicate that with increasing the number of images for training, we obtain higher accuracy. The reported accuracy is calculated on 1000 pair images from test dataset. Therefore, the model's accuracy is 99.8% on the images from the fasteners that the model was trained on.

To evaluate the model on new classes of fasteners which were not part of training, we tested the model with highest accuracy on images with images with multiple fasteners on a surface. Figure 5.1 shows an example of the images which used for testing the model. In these images, we used a mixture of fasteners that were used during training and new unseen classes of fasteners.

Table 5.3: Overview of the training results for sorting using Siamese networks, using 20 classes, 200×200 input image size, 1000 test pairs, and 40000 iterations.

#	Number of pair Images used for Training/Testing per Class	Accuracy
1	85/ 15	93.30%
2	85/ 15	95.30%
3	85/ 15	98.80%
4	150/ 50	99.80%
5	100/ 100	99.50%

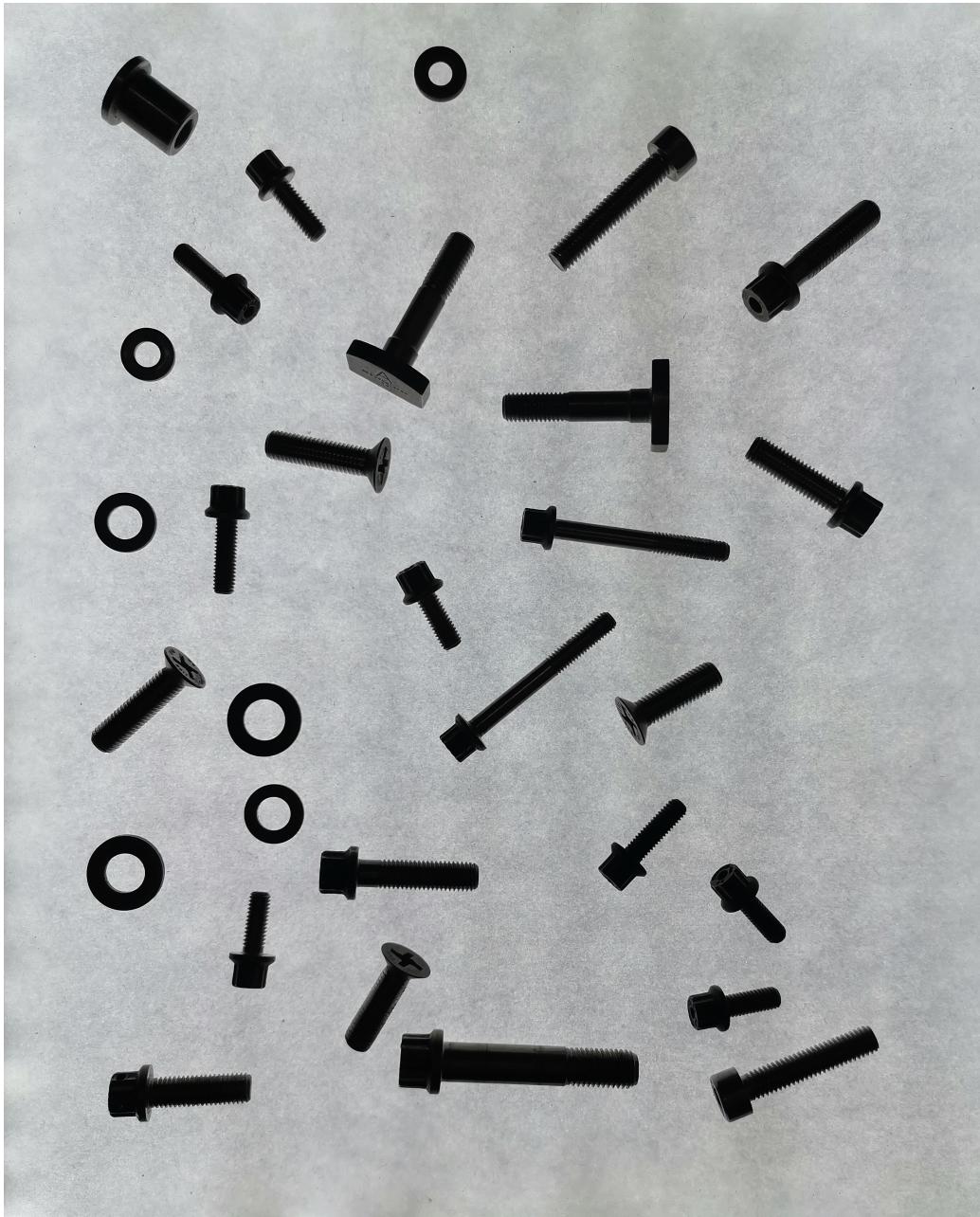


Figure 5.1: An example of the original input image that was used to test the accuracy of model on new classes fasteners.

Each image had at least one new class of fasteners. Depending on the combination of the new classes of fasteners, the accuracy of the model on the new classes was different. 70 images had only one fastener from new classes, i.e. the new fasteners cannot be paired with any other fastener on the image. 30 images

Table 5.4: Results of the model accuracy on new classes of fasteners, averaged in test images.

# of fasteners from new classes in image	# of images	Accuracy of fasteners of new classes	Weighted Accuracy
1	70	75%	96.6%
more than 1	30	20%	90.1%

had two or more than two fasteners of the same new classes. Table 5.4 shows the results of the model on the new classes of fasteners.

In 75% of the cases of having only one fastener from new classes, the model did not cluster them with any other groups of fasteners. In 20% of the cases of having two or more fasteners from same new classes, the model clustered these fasteners together without any error, while in 80% of the cases, some other fasteners from other classes were also clustered wrongly with the same new ones. To compute the real precision of the model, we used the weighted accuracy, which gives score to each right instance in a cluster. The weighted accuracy is calculated by:

$$WeightedAccuracy = \frac{NumberOfFastenersInCluster}{NumberOfAllFasteners} \sum ClusterAccuracy$$

Figure 5.2 shows the result of similarity detection on the input image shown in Figure 5.1.

The results of the tests indicates that the model obtains 99.8% accuracy on the fastener classes which were part of training and 90.1% accuracy on a mixture of training fasteners and new classes. To improve the results and the generalization of the solutions, more data from different fasteners classes are required. Moreover, the current images of the fasteners are from one top-down perspective. To detect the fasteners from all sides, another setup that uses multiple cameras must be utilized.



Figure 5.2: The result of computed similarity of the fasteners on the input image (Figure 5.1).

Chapter 6

Fine-grained Visual Categorization of Fasteners

In this chapter, we train a model using state of the art convolutional neural networks to classify the fasteners using *PPIC* configurations described in Chapter 3. Section 6.1 describes a single view classification method to classify bolts and washers. Section 6.2 uses multi-view convolutional neural networks to address certain different fasteners which from a single view appear to be of the same type, namely bolts. Finally, in Section 6.3, we explore synthetic data creation and its usage in training such models with less real data.

6.1 Single-view Classification

Categorization of fasteners can be defined as a Fine-grained Visual Categorization (FGVC) problem. The definition of FGVC originates from a Nilsback and Zisserman study [Nilsback and Zisserman, 2006] in which they introduced an object classification dataset for flower species. This problem was particularly challenging given the high intra-class and low inter-class variance nature among flowers, which made the older standard classification methods inefficient. Ever since, FGVC has gained popularity and has been used in different domains including classification of flowers, birds, dogs, aircrafts [Nilsback and Zisserman, 2006] [Parkhi et al., 2012] [Yang et al., 2012] [Maji et al., 2013]. FGVC has been also used for industrial purposes. Maji et al. introduced a large dataset of aircraft images for fine-grained visual categorization [Maji et al., 2013]. They obtained the images from aircraft spotter collections, maximizing internal diversity in order to reduce unwanted correlation between images taken by a limited number of photographers.

Aside from FGVC, for industrial purposes, work has also been carried out to

detect and classify fasteners, in particular in railway maintenance. Aytekin et al. analyzed the specific case of hexagonal headed fastener detection from depth images that were acquired using a high speed 3D laser range finder [Aytekin et al., 2015]. They described a fused approach with an appearance based and a histogram peak checking method. In another study, Feng et al. described a railway inspection system, which assesses the damage of multiple types of fasteners [Feng et al., 2013]. Their proposed system is insensitive to the illumination used and can model different types of fasteners using unlabeled data, and ranks the statuses of fasteners. Gibert et al. described another method for railway fastener detection by aligning training data, reducing intra-class variation, and bootstrapping difficult samples to improve the classification margin [Gibert et al., 2015]. They used histogram of oriented gradients features and a combination of linear support vector machine (SVM) classifiers to inspect ties for missing or defective rail fastener problems.

Laptev et al. designed a framework to incorporate expert knowledge on nuisance variations in the data during training deep neural networks [Laptev et al., 2016]. This framework handled prior knowledge on nuisance variations in the data, such as rotation or scale changes. Influenced by [Laptev et al., 2016], Xuan et al. designed a pearl classification machine, which automatically collected multi-view images of pearls [Xuan et al., 2017]. Using their machine, the pearls could be classified with a multi-view CNN. Usman and Rajpoot presented an algorithm to hierarchically classify tumor into three regions: whole tumor, core tumor and enhancing tumor [Usman and Rajpoot, 2017]. They extracted Intensity, intensity difference, neighborhood information and wavelet features from MRI scans with various classifiers.

This section explains the single-view classification of fasteners, where we classified bolts and washers using a single camera view from *PPIC-A*, described in Section 3.2.1. Parts of the presented work in this section have been previously published [Taheritanjani et al., 2019a]. Subsection 6.1.1 describes the dataset properties. Subsection 6.1.2 explains how we trained and evaluated the models. Finally, Subsection 6.1.3 shows the results and discussion.

6.1.1 Dataset Creation

Using *PPIC-A*, the images cannot present the screw and bolt recesses or the inner threads of nuts. Therefore, nuts and similar bolts with different head recesses are excluded from data creation.

We created the data in a number of iterations with different light intensities and a differing number of images. For the first iteration, we started with a small dataset, trained a model, and evaluated how well the created model performed. If the performance was below our expectations - less than 95% - we added ad-

ditional examples to the training data and tested the trained model again. After several iterations it was realized that we can divide the camera view into five regions. Placing the fastener in each of these regions can result in capturing a slightly different image in comparison with other ones. Figure 6.1 shows the five regions of the camera view and the differences between the sample captured bolt in each of them. Considering the vertical or horizontal position of the bolt along with its shaft, the bolt head and the reflections on the sides of its shaft are shown differently. Therefore, we must make sure that the dataset has sample images of the bolt in each of these regions.

We recorded 10 different images per region of each bolt - 50 images in total - and augmented 950 more images from these original images. Since the aero-engine washers are thin plates with a hole in the middle, without any threads and different sections such as a shaft and head, the quantity of actual washer images required to train a classifier may be less than with bolts and screws. Therefore, we only captured 10 images of washers without considering the region divisions and augmented 290 more images from them (see Figure 6.2).

6.1.2 Training and Evaluation

Figures 6.3 and 6.4 show the steps prior to saving the data on the disk and the preprocessing pipeline prior to training. After recording the images with the system, we converted the images to grayscale to filter out the color feature.

In addition, we created a square bounding box around the object and cropped the image to reduce both the background and the computation overhead during the training phase. The size of the bounding box was a fixed 500×500 square, so that the largest object in the dataset (bolts 11 and 19) fit in.

As the light reflection and shadows around the parts were minimized using light polarization and backlighting, it was possible to use rotation and translation augmentation techniques to enhance the dataset. In different runs, we realized that using augmentation without minimizing the shadows and light reflection is not helpful because the network learns these noises as part of the classification. For example, using a bolt image with shadows on its right side results in having shadows on the right side of the bolt after rotation and translation augmentations and the network may learn that this bolt should always have shadows on its right side. In this study, mirroring and flipping augmentations do not function well for threaded fasteners - right hand versus left hand fasteners.

In short, for the preprocessing, we applied the following methods: width and height shift of 10%, a range of 0.7 to 1.3 light intensity augmentation, to make the models more insensitive to slight light changes [McCarthy et al., 2013], and from 0° to 270° rotation just for the bolts. Using these augmentation techniques, we created 950 augmented data samples for each bolt and 290 for each washer,

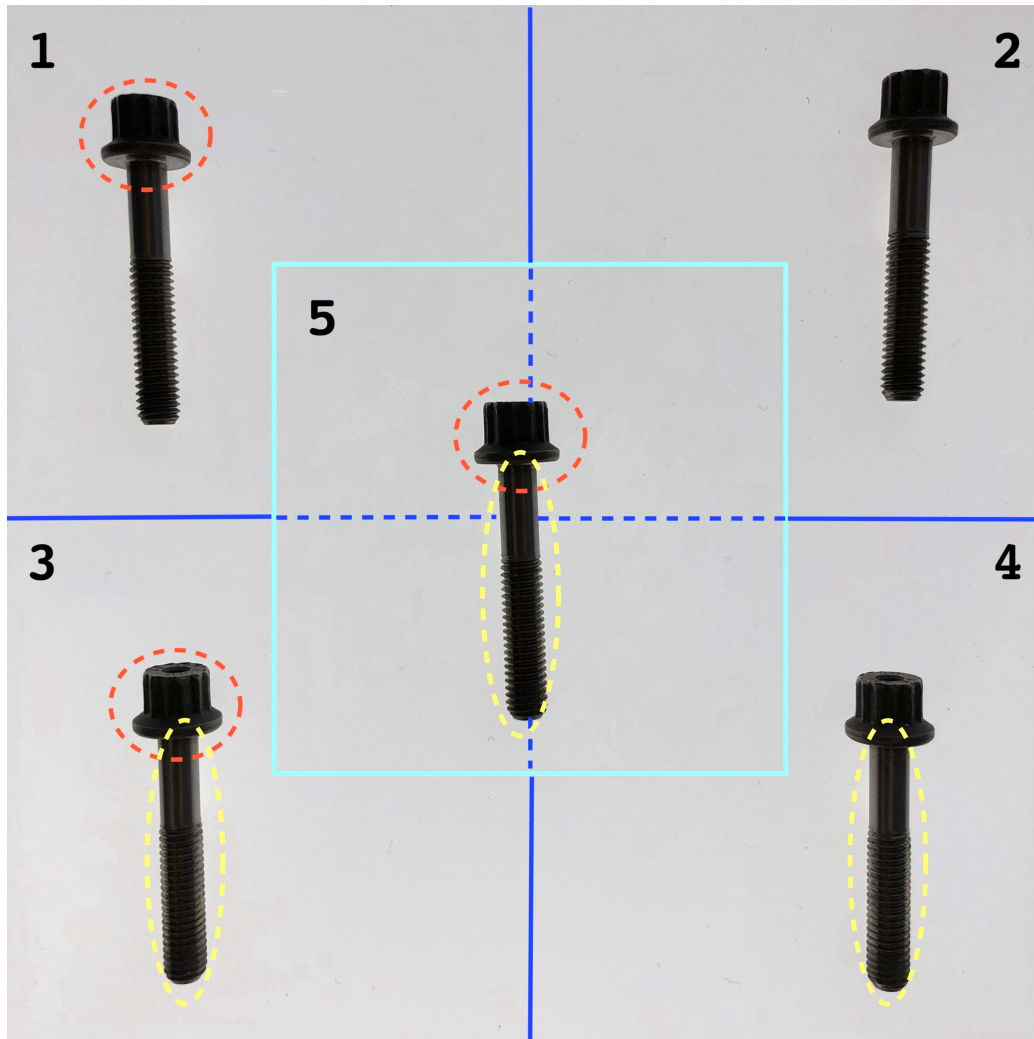


Figure 6.1: The camera view is divided into five regions shown in blue and aqua. The red dots indicate the differences between the bolt head when the bolt is moved vertically along its shaft. The yellow dots indicate the differences in light reflection on the bolt shaft when it is moved horizontally. The more we move the bolt to the left side of the view, the less reflection is captured on its left side, and vice versa [Taheritanjani et al., 2019a].

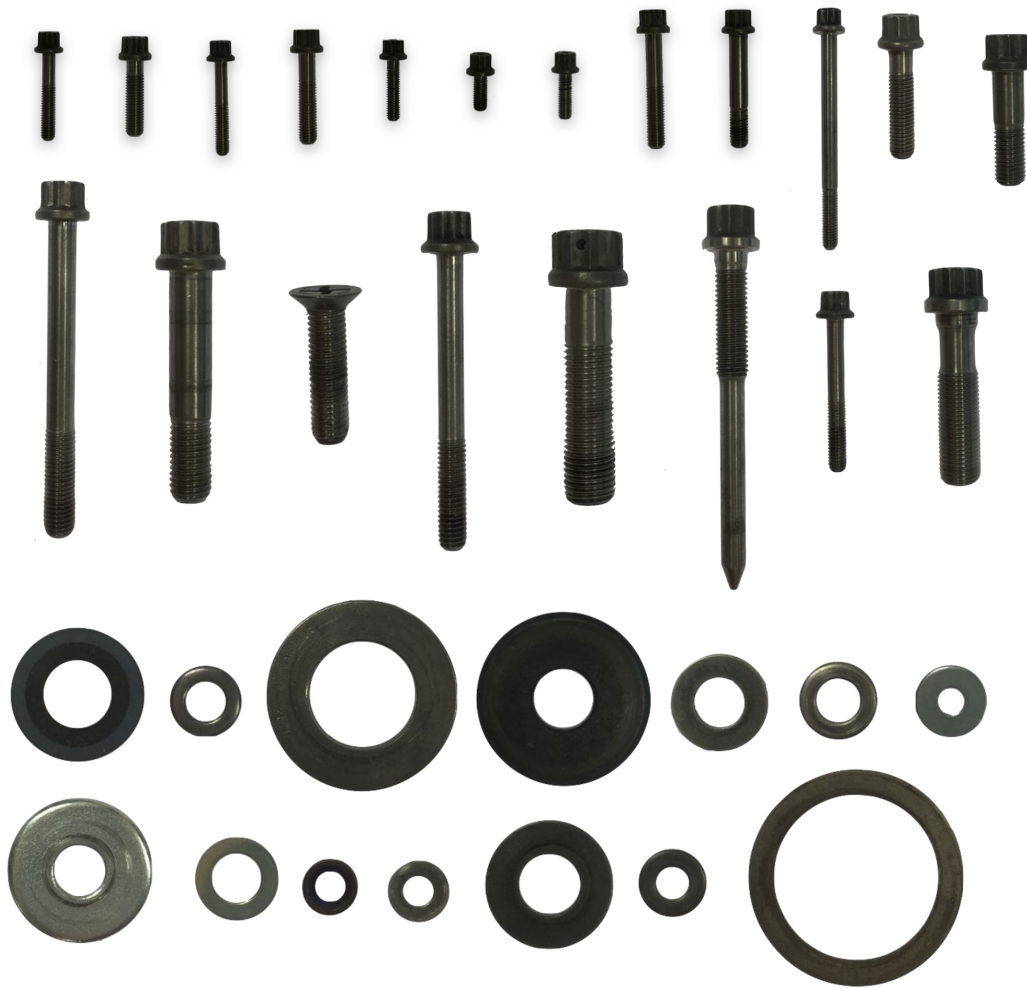


Figure 6.2: Twenty bolts and fourteen washers which were used for the classification.

and added them to the original images for the training.

We trained 23 different models for the fasteners. Table 6.1 summarizes the most important runs and their hyper-parameters - number of epochs, steps per epochs, and batch size. Apart from experimental runs to tune hyper-parameters of the convolutional neural network, we performed 9 different runs for training the classifier, each with different datasets and configurations. We started with AlexNet and VGG Net 19 for training the models and achieved less than 75% accuracy. Therefore, we decided to use the InceptionV3 Model with Keras on top of TensorFlow. We also used stochastic gradient descent (SGD) optimizer with the learning rate of 0.0001 and the momentum of 0.9 for all the trainings. 80% of the data was used for the train, 10% for validation, and 10% for the test in all of the experiments.

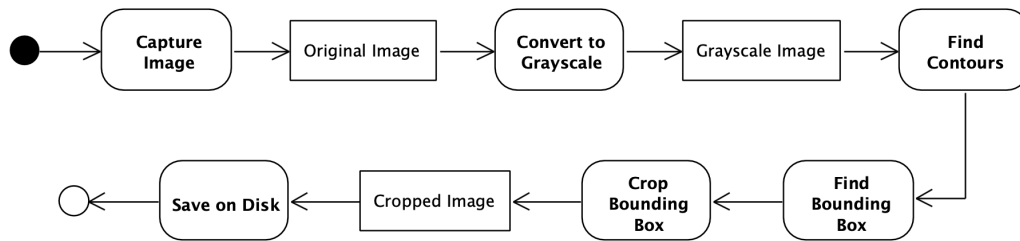


Figure 6.3: Overview of steps executed before saving the data on the disk (UML activity diagram).

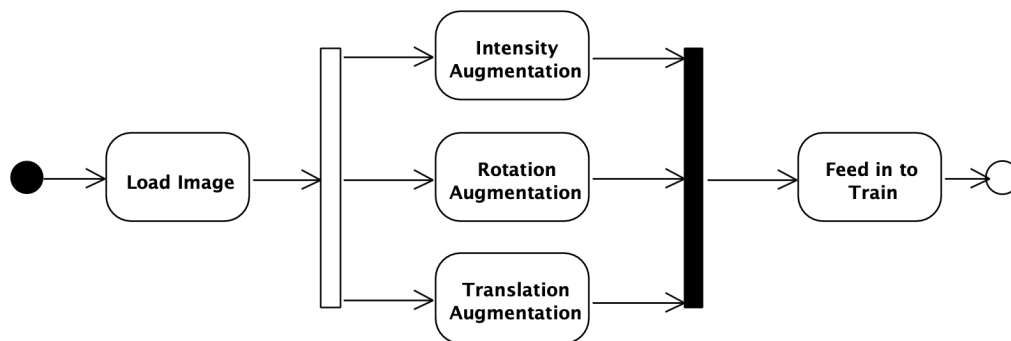


Figure 6.4: Overview of preprocessing steps executed before feeding the data to the train process (UML activity diagram).

Before using the dataset obtained by the system we conducted several experimental runs using images with normal ambient lighting to have a baseline for comparison. We started training using four similar bolts - bolts 1, 2, 3 and 4 in Figure 6.2 where bolt 1 and 2 were accordingly 5 and 2 millimeters smaller on their non-threaded shaft in comparison with bolt 3 and 4. However, the obtained trained model for bolts 1 to 4 using the normal ambient lighting never reached beyond 49% accuracy over the test dataset.

Subsequently we studied the performance of the following training datasets: 4 bolts (in four runs), 10 bolts, 10 bolts and 3 washers, 10 bolts and 6 washers, 10 bolts and 9 washers and 20 bolts and 14 washers - see Table 6.1. For 4 bolts we used the data obtained by the system and applied the preprocessing steps. By applying grid search and a process of trial and error in respect of the number of augmented images, number of epochs, number of steps per epoch, batch size and early stopping we finally obtained 98.9% accuracy on the test dataset, where the model was trained with 100 steps per epoch, and early stopping being applied after 50 epochs. We generated random data augmentations on the fly, which allowed us to have an infinite generated train dataset. Therefore, we changed the steps per

epoch to limit the number of training data during experiments.

For 10 bolts, we continued using the same configurations, although the number of validation steps during the training was increased from 3 to 30 resulting in overall 96.2% accuracy. Subsequently we continued adding washers to the dataset and increased the number of epochs in each experiment - runs 6 to 8 in Table 6.1. The accuracy of classification increased in each experiment with the addition of the washers. The surprising phenomenon during the experiments was the improvement in the classification results of the bolts after simply adding washers to the training dataset. We consider that this is because the addition of the objects with different shapes activates filters in convolution steps that were not active before. This could help to boost the filters to achieve superior results in comparison with previous results [Zilly et al., 2015].

Finally, we added another 5 washers and 10 bolts to the dataset and used 1210 steps per epoch for a total number of 250 epochs, a batch size of 20 and a validation step size of 100 and applied early stopping after 50 epochs - run 9 in Table 6.1.

Table 6.1: Overview of the training results, using InceptionV3 and Keras on top of TensorFlow, optimizer=SGD (Stochastic Gradient Descent), learning rate=0.0001 and momentum=0.9. The accuracy is reported on the test dataset which is 10% of the whole data.

#	Number of Classes (Bolts/ Washers)	Images per Class (Real/ Augmented)	Epochs	Steps per Epoch	Batch Size	Accuracy
1-4	4/ 0	5/ 95	100	20	20	51.1%
		20/ 180		40		79.5%
		25/ 470		100		89.3%
		50/ 950		200		97.7%
5	10/ 0	50/ 950	100	500	20	96.2%
6	10/ 3	10-50 / 290-950	100	545	20	96.7%
7	10/ 6	10-50 / 290-950	150	590	20	97.8%
8	10/ 9	10-50 / 290-950	200	635	20	99.4%
9	20/ 14	10-50 / 290-950	250	1210	20	99.4%

6.1.3 Results and Discussion

For training with all 34 fasteners in the dataset, we ultimately achieved 99.4% accuracy over the test dataset which contains 50 images of each fastener (see Table 6.1).

This study used *PPIC-A*, in which the input images cannot be obtained from multiple cameras. A single camera limits the approach to a single view and a sin-

gle view cannot extract all the discriminative features from some of the fasteners - namely bolts and nuts. Using multiple cameras, we can use multi-view CNNs to generalize the fasteners FGVC to all bolt types and nuts [Su et al., 2015]. We also used a polarized backlighting supported conveyor belt to create the datasets and the fasteners FGVC model can only be employed using this system. However, we assume aero-maintenance companies can simply provide such a system with reusing the existing materials and tools they have on their workstations - dimmable lighting and light diffusing textile.

6.2 Multi-view Classification

There are mainly two different approaches to classify 3D objects. The first approach uses 3D data, such as point clouds¹, or volumetric data, such as CAD models. The second approach projects the 3D shape of the object into 2D format such that CNN methods can be used for the classification.

Multi-view image analysis has been widely adopted for 3D shape classification. Wu et al. studied volumetric representation of 3D shapes and adopted 3D Deep Belief Nets [HOT06], to obtain superior results of shape classification on Princeton ModelNet [Wu et al., 2015]. Zhu et al. used autoencoders to learn 3D shape features with multi-view depth images, leading to accurate 3D shape retrieval [Zhu et al., 2016]. Su et al. described a multi-view CNN for 3D shape recognition, in which the multiple views features were integrated with an extra CNN [Su et al., 2015].

The term Multi-view CNN (or **MVCNN**) was used for the first time by Hang Su et al. in their study Multi-view Convolutional Neural Networks for 3D Shape Recognition [Su et al., 2015]. Multi-view CNN can be realized as an extension of a normal CNN. For each multi-view image (n images that are all of the same object), each image is passed through the feature extraction layers independently to produce n sets of feature maps. Afterwards, these n sets are reduced through max-pooling layers. The result is a set of feature maps of the highest activations between all n angles. These feature maps are then squeezed and used as input to the classification layers.

RotationNet also uses the same idea of Multi-view CNN [Kanezaki et al., 2018]. Similar to Multi-view CNN, it uses multiple views during training. However, during classifying unseen data, RotationNet only requires a subset of the views. RotationNet is able to classify the object even with only a single view instead of the same number used during training. It also differentiates itself by not requiring the objects to be aligned. The objects can lay in any orientation in

¹Point clouds are a set of points in the space with x, y, z coordinates.

a 3D space. RotationNet achieves this by introducing a latent variable which is optimized during training to learn which viewpoint is the most similar to one the network has seen during training.

This section describes the multi-view classification of fasteners, which focuses on the classification of nuts. However, multi-view classification can be generalized to all scale sensitive fasteners. We chose nuts because they are scale sensitive, and due to their shape, it is also challenging to capture their inner threads. Parts of the presented work in this section is extracted from a study conducted at the chair for applied software engineering at TU Munich [Birkeland, 2018]. Subsection 6.2.1 describes the dataset properties. Subsection 6.2.2 explains how we trained and evaluated the models. Finally, Subsection 6.2.3 shows the results and discussion.

















6.2.1 Dataset Creation

We used *PPIC-C* to capture the dataset (see Section 3.2.1). Although *PPIC-C* minimizes light reflection and shadow, it cannot entirely exclude them. Therefore, the fasteners could still have reflections and shadows from changes in natural lights from the surrounding in capture environment. To address this issue, the dataset was captured over a period of different days and time of day to capture light differences. The configuration consisted of two industrial cameras (Genie Nano GigE), angled at 90 and 45 degrees relative to the surface. Each image is captured with a resolution of 500×500 pixels. Some aero-engine nuts have heads and their shape is not symmetric. Therefore, these nuts were captured randomly in one of two positions, *head up* or *head down*. We chose to restrict the nut to only lay in one of these two positions due to the limitations of only having two cameras. Allowing the object to lay in any 3D position, such as on its side, may hide some of the discriminative features.

The dataset consists of 8 different classes of nuts; *A-1*, *A-2*, *A-3*, *A-4*, *M5-DIN934*, *M5-DIN985*, *M6-DIN934*, *M6-DIN985*, as shown in Table 6.2. The nuts in the dataset were selected in order to represent fasteners that can share multiple features but have a few features that distinguish them. The dataset contains a total of 400 multi-view images, 50 per class. Because each multi-view image consists of 2 images there is a total of 800 images.

The dataset was split into a train, validation and test dataset. The training and validation datasets are used during training, while the test dataset is only used for testing the expected performance on unseen data after deployment. The train dataset contains 72% of the images, while the validation and test dataset contain 14% each. The test dataset contained images taken on a different day and time to ensure that the it consists of unseen lighting features. The accuracies of each model are reported on the unseen test dataset.

Table 6.2: Eight nuts which are captured in the dataset.

Nut Name	Top View	45 °View
A-1		
A-2		
A-3		
A-4		
M5-DIN934		
M5-DIN985		
M6-DIN934		
M6-DIN985		

6.2.2 Training and Evaluation

Prior to training and to ensure that the network learns the generalization of the object rather than overfitting to the instances in the train dataset, we performed data augmentation. However, the augmentations must not change the discriminative features the network should recognize. Nuts are scale sensitive and their inner threads contain discriminative features. Therefore, the size information must not be altered and we must not use any augmentations such as cropping, scaling, horizontal or vertical flipping (the last two augmentations would change the direction of the threads). We used the following preprocessing steps and augmentations for all the experiments:

- **Resize:** we resized the image from the original size 500×500 to 224×224 , which is the accepted input size for ResNet. The resizing speeds up the computational time, and removes noisy high-level features which are not important for the classification. Lowering the spatiality of the input image allows us to focus more on the important features of the nut instead of small artifacts such as dirt and small scratches.
- **Color Jitter:** images recorded with small changes in ambient light, and dirty small parts can cause large differences in appearance. To avoid the network to overfit to the lighting conditions, or recognize dirty small parts which can appear darker, we applied random brightness and contrast augmentations. The brightness and contrast are randomly altered up to $\pm 40\%$ for each image.
- **Random Rotation:** we rotated the image to artificially increase the different rotations a nut can lay in. Each image is randomly rotated between $\pm 90^\circ$.

We defined the training parameters and a specific training procedure to ensure that the different training methods are fairly compared. This allowed us to not only compare the accuracy of each model, but also their respective computational, memory and time efficiency. We trained all models using the following parameters/hyper-parameters, and chose the model with highest validation accuracy to compute the testing accuracy.

- **Pre-training:** all the networks are pre-trained on ImageNet dataset² to speed up the training. Since all the implementations were built on top of standard architectures, we used the automatic loading functions in the

²<http://www.image-net.org>

deep learning frameworks to load the pre-defined architectures. In addition, we replaced the classification layers with randomly initialized fully connected layer, where the last layer has the same number of outputs as we have classes.

- **Optimizer:** for the choice of optimizer, we used standard stochastic gradient decent (SGD) with a momentum of 0.9 and weight decay of 10^{-4} . Momentum and weight decay help to speed up the training time and accuracies by minimizing the impact of noisy gradients and local minimas [Kingma and Ba, 2014] [Zeiler, 2012].
- **Learning Rate:** the learning rate determines how large the optimization steps are. Since we use a pre-trained model, the training starts with a proper weight initialization and is not improved by a high starting learning rate. We set the learning rate to a moderate static value of 10^{-3} . We could reach similar accuracies as lowering the learning rate over time by instead increasing the batch size, but in a shorter training time [Smith et al., 2017]. Therefore, we did not schedule a decrease in learning rate over time, and scheduled an increase in batch size.
- **Loss Function:** we utilized standard cross entropy loss for trainings [Tsai et al., 2007].
- **Batch Size:** the batch size determines how many samples must be loaded into memory and forward propagated together before calculating a optimization step. A smaller batch size allows for faster and more frequent steps. However, these steps are generally less precise. A larger batch size is slower but more accurate. We used an increased batch size instead of a decaying learning rate. By testing various batch size schedules, we found that a training procedure of a batch size of 4, for the first 20 epochs, followed by 5 epochs with a batch size of 20, gave us an acceptable trade-off between accuracy and training time.
- **Epochs:** an epoch is one full forward pass of the whole train dataset. Therefore, the number of epochs determines how many times the network sees each instance in the dataset. The number of epochs required to fully train a network depends on the size and complexity of the data. Using the batch size schedule defined in the previous points, we set the number of epochs to 25.

Table 6.3 shows an overview of the hyper-parameters and augmentations used during trainings.

Table 6.3: Overview of parameters and settings used to train models.

Optimizer	Epochs	Learning Rate	Batch Size	Augmentations
SGD (momentum=0.9, weight_decay=0.001)	25	0.001	4 (epoch 1-20) 5 (epoch 20-25)	Random rotation between $\pm 90^\circ$, random light intensity (color jitter) between $\pm 40\%$

To evaluate the model performance, we measured the accuracy and average computation time metrics. Average computation time measures the average time taken to classify a single nut during inference. Average computation time is calculated by:

$$AverageComputationTime = \frac{1}{n} \sum_n ComputationTime$$

It is also possible consider memory consumption as a metric. Memory consumption is defined as the amount of memory taken by the weights and bias of the model in the memory (either RAM or GPU's memory). The larger model size results in less space in memory to store training samples, and decreases the maximum batch size at any given time. However, since we did not define the memory consumption as a requirement in the study, we have not computed it.

We evaluated each algorithm running on AlexNet and ResNet18. Since the network sees the training samples in a random order, and the fully connected layers are randomly initialized, training models cannot have the exact same weights after a set number of epochs. Therefore, we trained each algorithm on each architecture independently three times. The reported results are an averaged testing accuracy over the three independent models.

6.2.3 Results and Discussion

Tables 6.4 and 6.5 show the averaged test accuracies and computation time over the three independently trained models. The two baseline algorithms SingleCNN and MultiCNN, show the performance of a standard CNN without any multi-view functionality. While Multi-view CNN and RotationNet uses multi-view for information sharing between the views. RotationNet built on ResNet18 outperforms other implementations in terms of speed and accuracy. Multi-view CNN outperforms the SingleCNN using AlexNet, while it suffers from a large drop in accuracy on ResNet18, where SingleCNN experiences a substantial increase.

Table 6.4: Classification accuracy on test dataset.

	SingleCNN	MultiCNN	Multi-view CNN	RotationNet
AlexNet	84.33%	79.67%	98.0%	42.67%
ResNet18	98.0%	95.67%	53.67%	100%

Table 6.5: Average computation time per nut during inference in ms.

	SingleCNN	MultiCNN	Multi-view CNN	RotationNet
AlexNet	13.25	22.32	26.85	6.36
ResNet18	14.21	28.33	29.21	8.01

In the SingleCNN and MultiCNN implementations, ResNet18 algorithm outperforms AlexNet while SingleCNN results outperform the MultiCNN's. However, in the case of splitting the MultiCNN into two separate networks and looking at their individual performance, we can observe a large difference. Table 6.6 shows the average accuracy results for each individual network from MultiCNN.

Table 6.6: MultiCNN individual network's test accuracy. *Network1* uses the 90° view images and *Network2* uses the 45° view images.

	Network1	Network2
AlexNet	36.33%	84.33%
ResNet18	22.67%	97.67%

Table 6.6 shows that *Network1* is not able to determine between the different classes and reduces the overall accuracy. *Network1* only uses images from the top view (see Table 6.2), which can only see the two discriminative features *OuterDiameter* and *InnerDiameter*. Using this information, *Network1* cannot correctly classify the nuts which share these features. On the other hand, *Network2* performs close to the SingleCNN approach by just using the 45° view. This shows that the 45° view is able to capture most of the important features. As expected, Table 6.5 also shows that the MultiCNN spends roughly twice the time to classify a single nut because each nut is being processed by two models instead of one.

Therefore, a single view CNN is able to learn the difference between highly similar nuts, but only given a view which contains almost all the discriminative features. Given a less than optimal view, such as the top view in the study, the model struggles as there is not enough discriminative features visible to distinguish between them. In addition, the MultiCNN approach does not address this problem, because there is no information sharing between the models. When one model is uncertain, it is not able to learn from other views and will produce widely

inaccurate results. When these results are aggregated into an overall classification result, the uncertain models reduce the accuracy.

According to Tables 6.4 and 6.5, the information sharing algorithms such as Multi-view CNN and RotationNet work well given a suitable architecture. The variation between two architectures produce differences in testing accuracy. It can be due to the fact that we used a set of training settings and parameters. Not all models can be trained effectively using the same hyper-parameters. Given more complex training algorithms such as Multi-view CNN and RotationNet, they can utilize different hyper-parameters.

RotationNet reaches a slightly higher accuracy than Multi-view CNN, 100% compared to 98%. Given longer training time, Multi-view CNN may also able to reach 100%. Nevertheless, according to Table 6.5, RotationNet is drastically faster during inference than Multi-view CNN, and also outperforms both of the baseline approaches.

One limitation of the study is the number of cameras. We showed that which particular view can observe the features that are discriminative for the classification accuracy. Using only two cameras, we were able to confidently classify nuts given that they are either in a *head up* or *head down* position. However, in real-life scenarios where the nut can lay in any random 3D orientation, we cannot ensure that all the discriminative features are visible, and may not be able to confidently classify the nut with the captured dataset,

Another limitation of this study is the dataset size. 50 images per class is considered a low amount, to capture all the different positions and conditions the nut can lay in. However, the image capture configuration produces smooth even lighting conditions and we specifically limited the amount of positions the nut can lay in. In this case, 50 images with augmentations could be enough to generalize the dataset. However, introducing more classes and cameras will require the dataset to be larger to preserve the same results.

6.3 Using Synthetic Data for Classification

CNNs are characteristically data-centric algorithms. The performance of a CNN algorithm depends on the availability of a dataset of images that can capture each target object's intra-class variability [Krizhevsky et al., 2012]. For classification of fasteners, we must train a CNNs using images that capture the distinguishing features of each of the fasteners. The task of collecting image datasets can be challenging and time consuming.

Synthetic images have been used for training in a variety of problems. Peng et al. used 3D CAD models to generate synthetic images and tested the invariance of convolutional neural networks to low level cues, such as object texture, color, 3D

pose and 3D shape, background scene texture, and color [Peng et al., 2015]. Peng et al. used part of the Office dataset [Saenko et al., 2010], which has the same 31 categories of common objects, such as cups and keyboards, in each domain. They compared the usage of 3D models with real texture against 3D models with uniform gray texture. The authors concluded that a CNN trained on synthetic images with real texture perform better than images with a gray texture. Gerogakis et al. used a mixture of synthetic images and real images to train a network for object detection in indoor scenes [Georgakis et al., 2017]. The synthetic images were created by superimposing images of the target objects on indoor backgrounds. Instead of rendering images from 3D models, cropped real images from the BigBird dataset were used [Singh et al., 2014]. In addition, the background images had depth map information (RGB-D images). The authors concluded that training an object classifier using a mixture of 90% synthetic images and 10% real images can produce comparable results to a classifier trained on only real images.

Rajpura et al. [Rajpura et al., 2017] used 3D models of refrigerators and items inside refrigerators from Archive3D1 and ShapeNet [Chang et al., 2015] to perform object detection in refrigerator scenes. Synthetic images were generated by rendering 2D images of a 3D synthetic scene where the products are placed inside the refrigerators. The authors trained a CNN using a fully real train dataset, a fully synthetic train dataset, and a synthetic train dataset containing 10% real data. The CNN fully trained with only synthetic images underperformed against the one with real images but the mixed train dataset boosts the detection performance by 12% which signifies the importance of transferable cues from synthetic to real.

Sarkar et al. trained a CNN image classifier on synthetic images rendered from 3D models of five different objects [Sarkar et al., 2017]. The 3D models were created by scanning the real target objects using a 3D scanner. Synthetic images were created by rendering the 3D models on 3 different types of backgrounds: a plain white background, a random indoor background taken from indoor categories of PASCAL dataset [Everingham et al., 2010], and a chosen background similar to that of test images. Additionally, images were rendered with two object texture settings: full texture and no texture. The authors found that synthetic images of fully textured objects overlaid on a mixture of white and chosen backgrounds produce the best results compared to other texture and background settings.

This section describes the experimental results of using synthetic data in combination with real data to train the classification models. This is inherently different approach than typical data augmentation techniques. Using data augmentation, we use image transformation to artificially expand the size of a train dataset by creating modified versions of images. We typically perform data augmentation prior to training the network and after loading the batches to the memory. On the other hand, when using synthetic data for training, we capture the train dataset using artificially rendered images from a 3D model, such as a CAD model. We

typically store these images on the disk before loading the batches to the memory. We trained CNNs that used both real and synthetic images. We define real images as the natural images of the fastener, captured using a camera. Synthetic images, on the other hand, are artificially rendered images. They are 2D renditions of 3D models of fasteners. Parts of the presented work in this section is extracted from a study conducted at the chair for applied software engineering at TU Munich [Abdelraouf, 2018]. Subsection 6.3.1 describes the dataset properties. Subsection 6.3.2 explains how we trained and evaluated the models. Finally, Subsection 6.3.3 shows the results and discussion.

6.3.1 Dataset Creation

The real and synthetic images must capture the fasteners and their corresponding 3D models from different angles and in different positions. For this purpose, we performed transformations to the 3D models of fasteners. We used two types of transformations: translation to change the position of the target object, and rotation to change the angle. Each transformation has a range. This prevents the target objects from being translated or rotated away from the camera view.

To obtain 3D models, we scanned the selected a fastener using 3D scanner. However, we were unable to create 3D models with satisfactory quality (see Figure 6.5) for two reasons. First, the small size of the target fastener required a great level of precision to create an accurate 1:1 model. Secondly, the reflective surface of the small parts bounced the scanner's light off and caused the scanner to capture a lot of clutter. Figure 6.5 shows a fastener and its corresponding scanned 3D model. The scanner was not able to accurately scan fine details like the small part threads.

To address this problem, we used readily available 3D models for the experiment. We downloaded a 3D model from the Traceparts website³. Figure 6.6 shows an example image of each downloaded fastener.

To generate real images, we used *PPIC-A*, described in Section 3.2.1. We placed the fastener on the conveyor belt and used backlighting underneath. An iPhone 6s placed over the plane was used as the camera in this study. We captured images such that the fastener is fully within the viewfield of the camera. The real images taken by the iPhone camera were resized to conform with the height and width required by the image classifier.

For generating synthetic images, we used a synthetic scene: an artificial environment created in 3D modeling software. Figure 6.7 shows an example of a synthetic scene, where a 3D model is placed in a synthetic scene in 3D modeling software. The 3D modeling software renders a 2D image of the environment to

³<https://www.traceparts.com>

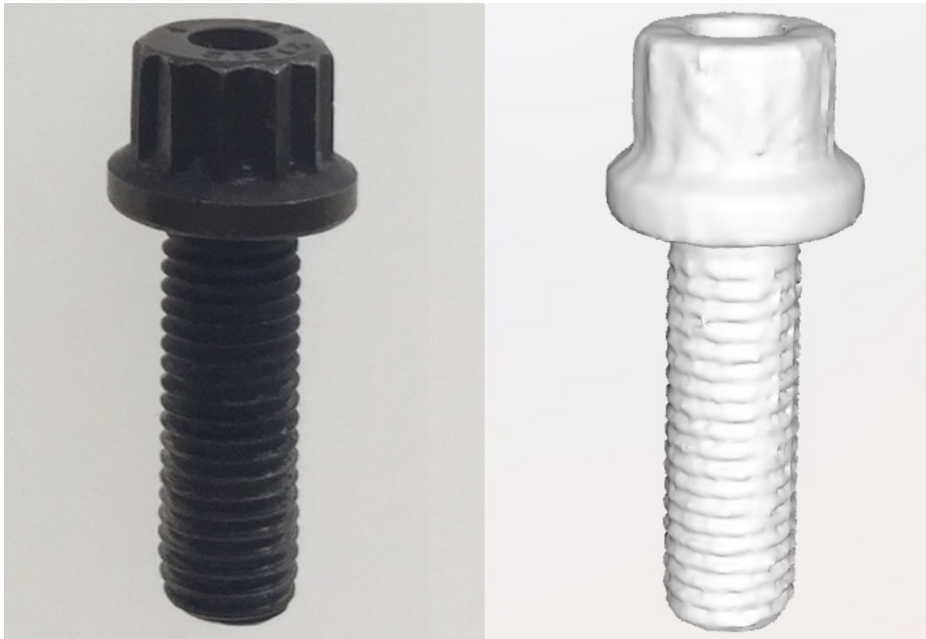


Figure 6.5: An example fastener (on the left) and its 3D model created by 3D scanner (on the right). The 3D model is rendered without texture for clarity.

create a synthetic scene. We created the synthetic scene in the Rhinoceros 3D modeling software⁴. First, an image of the real horizontal plane is captured and used as a background for the synthetic scene. A synthetic lighting source was placed underneath the plane to mimic the lighting effect of the real environment and the 3D model was placed on the horizontal plane of the environment. Subsequently, we created a data generator script that uses the Rhinoceros library to manipulate 3D objects in the Rhinoceros software. The synthetic data generator obtained the rotation and translation ranges of the 3D model.

After capturing and generating the images and saving them to a dataset folder on the disk, the dataset was split into a train dataset, a validation dataset and a test dataset. Table 6.7 presents the number of images per class for each dataset. We generated 5 different datasets using different range of real data: a fully synthetic train dataset R_0S_{100} , a 2.5% real train dataset $R_{2.5}S_{97.5}$, a 5% real train dataset R_5S_{95} , a 10% real train dataset $R_{10}S_{90}$, and a fully real train dataset $R_{100}S_0$.

6.3.2 Training

We trained VGG16 and VGG19 models for each data split [Simonyan and Zisserman, 2014]. For the choice of optimizer, we used the Stochastic Gradient

⁴<https://www.rhino3d.com>



Figure 6.6: Examples of images for each class that uses in the experiments. For each pair, the image on the left is a sample real image, while the image on the right is a sample synthetic image.

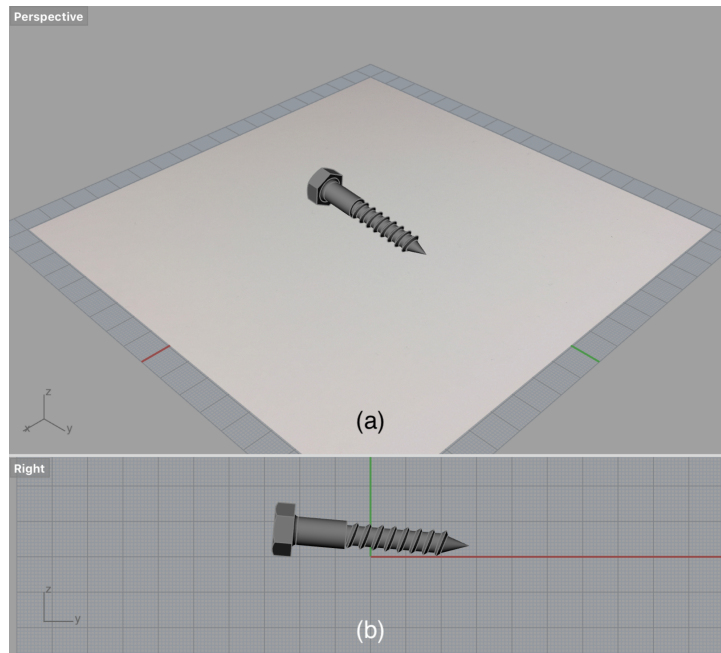


Figure 6.7: An example synthetic scene where a screw lies on a horizontal plane. The isometric view of the scene (a). The side view (b).

Descent (SGD) and Adam [Kingma and Ba, 2014]. Similar to the multi-view classification, we utilized transfer learning by using pre-trained CNN models on the ImageNet dataset [Pan and Yang, 2009]. Furthermore, the CNN models are tweaked to train and evaluate images of size 500×500 pixels. Since we used pre-trained CNN models, each layer was pre-loaded with optimized weights. During the trainings, we froze some of the early network layers to preserve these optimized weights. For VGG16, the first 10 or 14 layers were frozen. For VGG19, the first 12 or 16 layers were frozen.

Table 6.8 shows an overview of the hyper-parameters and settings used to train the models. Notice that due to using the synthetic data, we did not utilize

Table 6.7: The number of images per class for the specified data split.

Dataset	Train (Synthetic)	Train (Real)	Validation	Test
R_0S_{100}	600	0	100	150
$R_{2.5}S_{97.5}$	585	15	100	150
R_5S_{95}	570	30	100	150
$R_{10}S_{90}$	540	60	100	150
$R_{100}S_0$	0	600	100	150

Table 6.8: Overview of parameters and settings used to train models.

Optimizer	Epochs	Learning Rate	Batch Size	Frozen Layers
SGD, Adam	30	0.0001	4, 8, 16	10/14 (VGG16) 12/16 (VGG19)

Table 6.9: Class-wise classification accuracy of a VGG16 networks.

Dataset	Flat Head	Hex Large	Hex Small	Long-hex Large	Mushroom Large	Mushroom Small	Average
R₀S₁₀₀	99.33%	48.00%	98.67%	96.67%	80.67%	78.00%	83.56%
R_{2.5}S_{97.5}	100%	79.26%	99.26%	99.26%	83.70%	80.74%	90.37%
R₅S₉₅	100%	95.83%	100%	97.50%	90.83%	75.83%	93.33%
R₁₀S₉₀	100%	94.44%	98.89%	100%	98.89%	90.00%	97.037%
R₁₀₀S₀	100%	100%	100%	100%	100%	100%	100%

data augmentations in this experiment.

6.3.3 Results and Discussion

Tables 6.9 and 6.10 provide the results of class-wise classification accuracy of training a VGG16 and a VGG19 network respectively. The presented classification accuracy is the result of evaluating each CNN model using the respective test dataset.

The results show that the results of trainings on fully synthetic train datasets underperform the results of trainings on fully real train datasets. However, A VGG16 trained on a train dataset with 10% real data produces comparable results to a train dataset with fully real data. Moreover, the VGG19 network achieved an accuracy of 99.33% using the R₁₀₀S₀ dataset, compared to an accuracy of 97.22% as a result of using the R₅S₉₅ dataset. The Flat Head and Long-hex Large fasteners had the highest class-wise accuracies. This can be due to the different shape of

Table 6.10: Class-wise classification accuracy of a VGG19 networks.

Dataset	Flat Head	Hex Large	Hex Small	Long-hex Large	Mushroom Large	Mushroom Small	Average
R₀S₁₀₀	100%	81.33%	88.67%	92.00%	98.67%	52.67%	85.56%
R_{2.5}S_{97.5}	100%	100%	86.67%	99.26%	99.26%	90.37%	95.93%
R₅S₉₅	100%	100%	96.67%	98.33%	100%	88.33%	97.22%
R₁₀S₉₀	100%	90.00%	96.67%	98.89%	98.89%	96.67%	96.85%
R₁₀₀S₀	100%	100%	100%	100%	99.67%	97.00%	99.33%

these two fasteners in comparison with the others. The other four fasteners look more similar, which results in a lower class-wise classification accuracy for them.

One limitation of this study is the 3D model of the fasteners in order to generate synthetic images. Since the 3D scanning of the fasteners could not provide acceptable result, we must use 3D models from other sources. In addition, we only used VGG networks and did not perform any experiment on other CNNs. Other CNNs may perform and generalize better with synthetic data. We also limited the studies to the settings and parameters explained in Section 6.3.2. Using other hyper-parameters such as training the network with more epochs may lead to different results.

Another limitation of the study is the number of views we used for the experiment (only one). Using more views and utilizing the Multi-view CNN (explained in Section 2.3.2) could result in better performance for the synthetic data. We also used 600 images per class and 2.5% to 10% of the real data for the mixed datasets. Changing these amounts can highly vary the results.

In this chapter, we described a set of methods for categorization of fasteners in overhaul processes. To compare, Table 6.11 shows a comparison between the fastener categorization approaches used in this dissertation and the state of the art results with regards to the rating scheme, described in Section 1.2.

Table 6.11: Comparison between the fasteners categorization approaches used in this dissertation and the state of the art results in single-view/multi-view image classification with regards to the rating scheme.

Study	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆
[Xuan et al., 2017]	±	-	-	±	-	±
[Usman and Rajpoot, 2017]	±	+	-	-	-	+
[Sironi et al., 2018]	-	+	±	±	+	+
[Su et al., 2015]	±	+	±	-	-	±
[Qi et al., 2017]	-	+	±	±	-	±
[Liu and Kang, 2017]	±	+	±	±	-	±
[Kanezaki et al., 2018]	±	+	±	-	-	+
Fasteners categorization approaches used in this dissertation	+	+	±	+ (multi-view) ± (single-view)	±	+

Chapter 7

Bin Picking

Picking small parts and placing them into specific containers in overhaul plants can be formulated as a bin picking problem in which all small parts and fasteners are grouped and piled on a specific part of the workstation. Most industrial bin picking solutions determine the pose of an object and the grasp point by matching a CAD model to the point cloud obtained from a 3D camera [Spennath et al., 2013] [Dieter Schraft and Ledermann, 2003] [Palzkill and Verl, 2012]. However, many machines use different fasteners produced by different manufacturers with different CAD models, which may not always be made available by vendors. Moreover, 3D bin picking solutions, which can operate without CAD models of objects, can be prohibitively expensive for many projects. In addition, the characteristics and nature of industrial fasteners and small parts introduce various problems, such as reflection and shadow [Taheritanjani et al., 2019a], which makes fastener picking more challenging. Finally, using only a single 2D image, it is impossible to extract the 3D position of objects. Without any knowledge of the 3D orientation of objects, robots can only pick the non-occluded parts that are placed flat relative to the camera lens [Kim et al., 2012] [D’Avella et al., 2020].

The Bin picking approaches can be divided into two main categories: 1) methods that only compute grasp points and 2) methods that utilize an object detection framework to find suitable grasp points on the detected objects. The former provides a heat map that shows the probability of a successful grasp and selects the highest one. However, such methods require the orientation of the grasped object. Recently, using multiple robots in parallel [Levine et al., 2018] or computing a heat map from artificial grasps in simulations for parallel grippers [Mahler et al., 2017] have improved object grasping performance.

The latter category attempts to find suitable grasp locations by estimating the pose of an object and then extracting grasp points. Numerous bin picking 3D software applications use such methods to search for local maxima in a depth map to begin with pose estimation of the known object by fitting a CAD model

and determining its pose using a lookup table [Dieter Schraft and Ledermann, 2003] [Palzkill and Verl, 2012] [Spenrath et al., 2013]. Then, a collision free path is computed to a predefined grasp location on the object.

Deep neural networks have been used for pose estimation [Do et al., 2018]. Brachman et al. described a 6D pose estimation system for object instances and scenes which only needs a single 2D image as input [Brachmann et al., 2016]. Wu et al. proposed to jointly optimize the model learning and pose estimation in an end-to-end deep learning framework [Wu et al., 2019]. Their method produces a 3D object model and a list of rigid transformations to generate instances to minimize the Chamfer distance.

Object detection and semantic segmentation have received wide attention by the research community and many studies have been conducted in these areas. Gupta et al. applied iterative region segmentations and manipulation of regions to sort small objects on a tabletop [Gupta et al., 2014]. State-of-the-art results have been published using various frameworks, such as Fast R-CNN [Girshick, 2015], Faster R-CNN [Ren et al., 2015], and Mask R-CNN [He et al., 2017]. Some studies have used CNNs to solve segmentation tasks for image classification [Wang et al., 2018] [Chen et al., 2018] [Chen et al., 2017], while others have used fully convolutional networks [Long et al., 2015] [Noh et al., 2015]. Hema et al., Schwarz et al., and Danielczuk et al. designed segmentation methods for bin picking tasks [Hema et al., 2007] [Schwarz et al., 2018] [Danielczuk et al., 2019].

In this section, we describe two different methods to recognize individual parts automatically. These methods can be used in combination with a robotic arm to automatically pick parts and place them into compartments. In case of using a pneumatic, hydraulic, or servo-electric robotic gripper, parts must be picked by a two-fingered pinch grasp. Therefore, the robot must receive the orientation of the part and the center point between its two pinch fingers as the grasp point. Similarly, when using a vacuum robotic gripper, small parts must be picked by a suction cup directly from the grasp point relative to the orientation of the part. Therefore, the orientation and grasp point of parts must be computed.

Section 7.1 describes an approach that uses image thresholding and contour detection for the parts on a uniform background to segment objects and to find pickable parts and a suitable grasp point. Similarly, Section 7.2 uses Mask R-CNN approach for the parts that occlude each other to segment objects and to find pickable parts and a suitable grasp point. This approach can use different section segmentations to calculate the orientation, i.e., different classes must be segmented to compute the grasp point and determine the orientation or pose of the object. On the other hand, this approach can also utilize principal component analysis [Jolliffe, 2011] and Image Moment [Karakasis et al., 2015] on the detected masks. We demonstrate that this is especially beneficial for picking small parts in overhaul processes because, by using 2D image, we can propose pickable

objects to robots with vacuum, pneumatic, hydraulic, or servo-electric grippers. Therefore, our approach is independent of the robotic gripper and can segment objects to be picked using only 2D images.

7.1 Bin Picking on Uniform Background

This section describes a thresholding computer vision approach to segment the parts for bin picking on a uniform background. We define uniform background as a background where we can apply computer thresholding algorithms to find the fastener contours. Therefore, a uniform background consists of a uniform color, where the light reflection and shadow effects are reduced. A uniform background can be achieved using backlighting and light polarization (see Chapter 3). In addition, to use thresholding algorithms, the fasteners must not occlude each other¹. To perform bin picking for such fasteners, there are two options: 1) properly placing the fasteners on the surface prior to bin picking, or 2) using a mechanical solution using vibration and conveyor belt (see 3.2) to separate the fasteners. We use the latter in this study. Subsection 7.1.1 describes the data collection process. Subsection 7.1.2 explains the methods we used in this section. Subsection 7.1.3 describes how we trained and evaluated the models and finally, Subsection 7.1.4 shows the results and discussion.

7.1.1 Data Collection

We acquired data from 23 different bolts. The images were captured using *PPIC-A*, described in Section 3.2.1, with a fixed camera (Logitech c920) on top of the conveyor belt. We captured 95 images and converted them to grayscale. We also cropped the borders of the image to ensure that there is a safe distance between the image margin and the conveyor belt border. Each image was 1440×1080 pixels and pictured the bolts that were separated from each other on the vibration plate and brought under the camera by the conveyor belt. Figure 7.1 shows examples of the images in the dataset.

Since we had a uniform background using the backlighting under the conveyor belt, we performed the image thresholding algorithms and found the bounding box of object contours in the images. We cropped the bounding boxes and used them for training an occluded/not-occluded classifier. Figure 7.2 shows the preprocessing steps to capture the data for training the classifier.

¹We do not consider to terminate the occlusion. Depending on the setup, it is sufficient to achieve only a partial non-occlusion, in which considerable amount of the objects are not occluded each other.

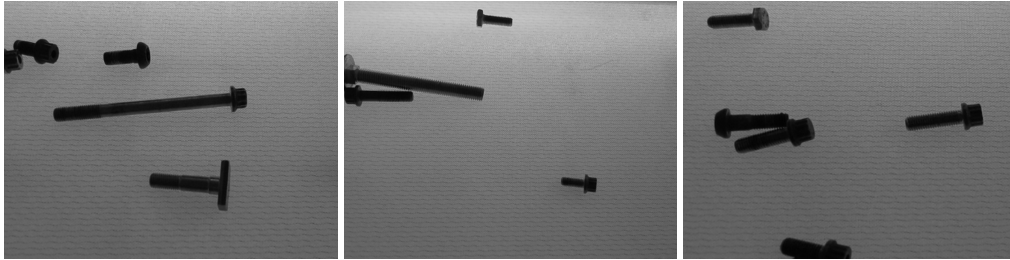


Figure 7.1: Examples of the images captured for training a classifier, using *PPIC-A*.

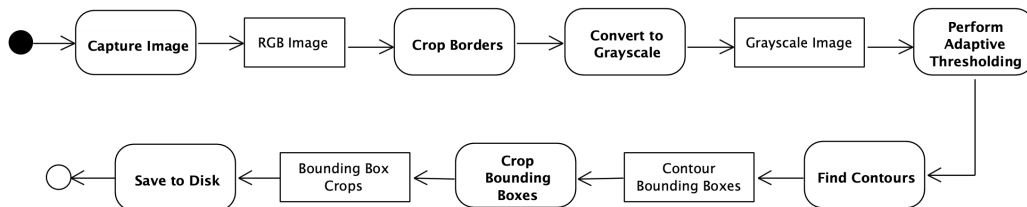


Figure 7.2: Overview of preprocessing steps executed prior to train an occluded/not-occluded classifier (UML activity diagram).

7.1.2 Method

The cropped images obtained in 7.1.1 can contain multiple fasteners. This may rise one the following issues: 1) the fasteners occlude each other, 2) they are placed next to each other without any distance in between, or 3) they are placed in a close range of each other. In case of occlusion or placing next to each other, the thresholding algorithms cannot detect the fastener contour separately. Figure 7.3 shows an example of a possible result when fasteners occlude or are placed next to each other. In case of fasteners placing in a close range of each other, the robotic arm may not be able to pick the fasteners or may collide the other fasteners when picking one, which results in non-deterministic behavior. Therefore, to avoid having multiple fasteners in one image, we first train a classifier to detect the occlusion or presence of multiple fasteners. Using *PPIC*, it is possible to ignore such cases; they are sent back to a vibration plate on the beginning of the conveyor belt and can be placed differently.

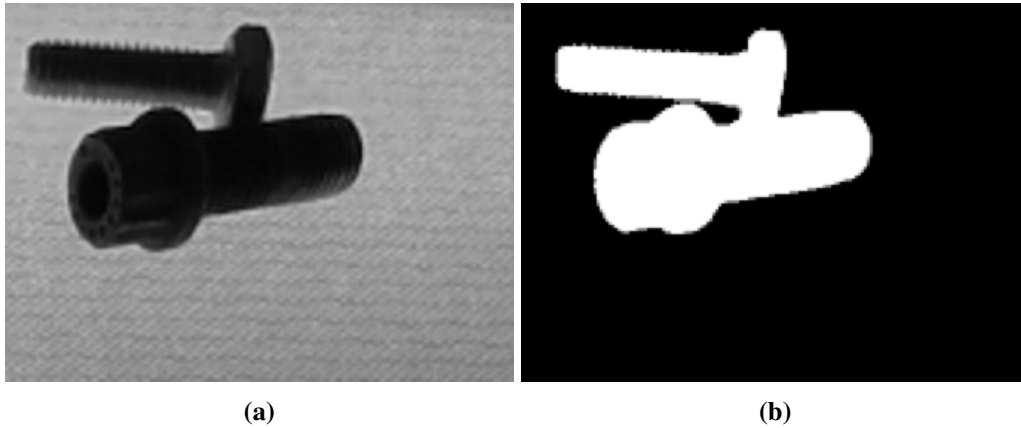


Figure 7.3: Example of fasteners that are placed next to each other (a) and their contour obtained from adaptive thresholding algorithm (b).

Classifier Training

Prior to computing the orientation and grasp point of the bolts, we first train a model to classify pickable/non-pickable bolts, i.e. whether a crop consists of a single bolt with no occlusion or there are multiple bolts, possibly occluding each other. We train a binary classifier using the crops obtained from applying thresholding and bounding box computation in 7.1.1. We define an offset of 90 pixels for width and height of the detected bounding boxes to ensure the fasteners are distant from the contour of the image. There were 134 single-bolt not-occluded images and 797 occluded or multiple-bolts images.

The pickable/non-pickable classifier is trained with DenseNet121 pretrained on ImageNet dataset [Deng et al., 2009]. We used 15% of the dataset for validation, i.e., 22 images of pickable bolts and 120 images of non-pickable bolts. During training, we set the class weight argument to give a larger weight to under-represented class (pickable) and a lower weight to overrepresented classes (non-pickable). Table 7.1 gives an overview of the training parameters and settings.

Table 7.1: Parameters and settings used to train pickable/non-pickable classifier.

# of epochs	Learning Rate	Optimizer	Augmentations
25	0.001	SGD (momentum=0.9)	10% width and height shift, Blurring, random rotation from 0 to 270°, range of 0.7 to 1.3 light intensity

Orientation Computation

The bolt orientation can be inferred as the line segment between the center of the bolt and its head. To calculate the orientation, we used principal component analysis (PCA). PCA can identify two axes that account for the largest amount of variance in the data [Jolliffe, 2011]. We can compute the orientation of the bolts (the green axes in Figure 7.4) by selecting the axis that minimizes the mean squared distance between the input pixels and their projection onto it and calculating the axis's slope.

PCA does not return the directional orientation by default. The direction of the axis can be oriented towards the head or the shaft of the bolts. To address this issue, we calculated the centroid of the cropped image (see 7.1.2) and the center of the cropped image. The average pixel intensity of the image is higher where the bolt head is placed. Therefore, the centroid of the image is not in the middle and is shifted towards the head. Hence, we can ensure that the slope of the line segment from the middle of the crop towards the centroid is less than 90° from the orientation of the bolt towards the head. Using this slope to check the PCA-calculated orientation, we can adjust the orientation towards the head.

Grasp Point Computation

We considered potential grasp point as the middle of the bolt. To compute the grasp point, we used the image moment, which is a weighted average of an image's pixel intensities. Image moment can capture the centroid (geometric center) of the object as the statistical properties of the shape, which can be used as the grasp point of the object. In Figure 7.4, the pink circles show the grasp point calculation result obtained using image moment for two bolts.

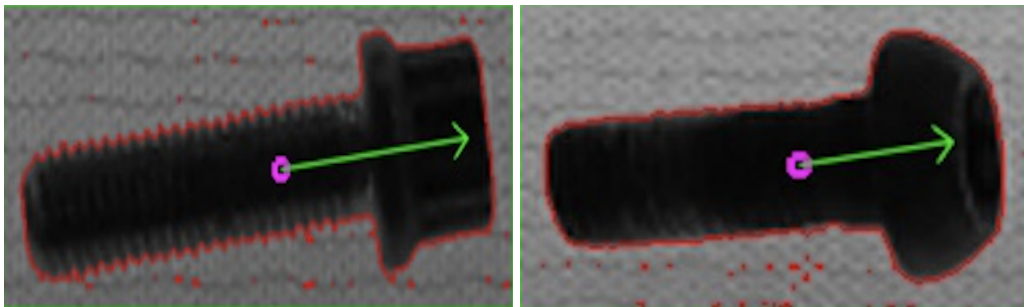


Figure 7.4: Examples of orientation (green axis) and grasp point (pink circle) calculations for two bolts.

7.1.3 Evaluation

To evaluate the method, we used *PPIC-A* to capture 30 different images. Each image contained multiple fasteners laid randomly on the conveyor belt. In total, these images had 72 pickable bolts, which were counted manually to compare and measure the results (pickable bolt is defined a stand alone bolt within the margin of 90 pixels from its contour in each direction).

To evaluate the occluded/not-occluded classifier, we used the accuracy, precision and recall. The orientation errors are calculated in rotation angle (the angle in radian by which the computed orientation must be rotated to reach the actual orientation). The grasp point errors are the distance between the computed grasp point and the actual grasp point (Δd in pixels).

7.1.4 Results and Discussion

Table 7.2 summarize the classifier, orientation, and grasp point computation evaluation results.

Table 7.2: Overview of classification, orientation and grasp point calculation results.

Accuracy	Precision	Recall	Orientation Computation Error (radian)	Grasp Point Computation Error (pixel)
96.1%	98.2%	88.6%	0.029	8

The pickable/non-pickable bolts classifier obtained 96.1% accuracy during evaluation. Moreover, The computation error of the orientation and grasp point was 0.029 radian and 8 pixels respectively. One challenge during pickable/non-pickable classification is the bolts which are partially within the cropped image, i.e. part of their head or shaft is outside the image. In case of using a pneumatic, hydraulic, or servo-electric robotic gripper, bolts must be picked by a two-fingered pinch grasp. We used a fixed offset from the part contour during cropping the image to ensure avoiding any collision with other parts on the surface or grasping multiple parts. If part of the bolt is outside the crop, we cannot ensure the safe offset distance between the bolt and other parts. Therefore, these instances must be classified as non-pickable. However, it is challenging for the classifier to distinguish between a pickable bolt that its contour is close to the border and a bolt that part of its shaft is outside the crop. One solution could be to label all the bolts images with their contour close to the border to non-pickable. However, this can slow down the detection of pickable bolts.

7.2 Bin Picking on Non-uniform Background

This section presents two different methods for bin picking on non-uniform backgrounds, especially when the objects occlude each other. For this section, we focused on flat-shaped objects, such as crank², that stack on each other under ambient light conditions. However, these methods also can be generalized and used for other type of fasteners. Parts of the presented work in this section have been previously published [Taheritanjani et al., 2020].

Subsection 7.2.1 describes the data collection process and dataset properties. Subsection 7.2.2 describes the process of training an instance segmentation model. In Subsection 7.2.3 we explain the methods and approaches. Subsection 7.2.4 describes how we trained and evaluated the models and finally, Subsection 7.2.5 shows the results and discussion.

7.2.1 Dataset Creation

We acquired data from three different cranks. The images were captured using a fixed monocular camera (Genie Nano GigE) on top of a workstation. For training and validation, we used 12 grayscale images. Each image was 1280×1024 pixels and pictured a pile of cranks on the workstation in ambient light. Figure 7.5 shows examples of the images in the dataset.

We annotated each image using the VGG Image Annotator [Dutta and Zisserman, 2019]. For pickable cranks, we annotated their outer contour and two of the holes on their surface. Since the robotic arm can only grasp parts that are not occluded by others and placed flat, we annotated the outer contour of all unpickable parts as a different category. Figure 7.6 shows an example image with annotations from the dataset.

7.2.2 Model Training

A robotic arm requires a grasp point and the orientation of a crank to pick it for packaging; therefore, we trained a Mask R-CNN model to segment cranks in the images. We used Mask R-CNN with Resnet50 as the network backbone, and the weights were pretrained using the COCO dataset [Lin et al., 2014]. We used 15% of the dataset for validation, i.e., two images for each crank. During training, each pixel was labeled as either background class 0 or other classes. The error was calculated by finding the mean over the loss for all classes, and the network was validated by taking the mean pixel intersection over union (IoU), with the mean

²Crank is part of an axle in gearboxes that is used for converting reciprocal to circular motion and vice versa.

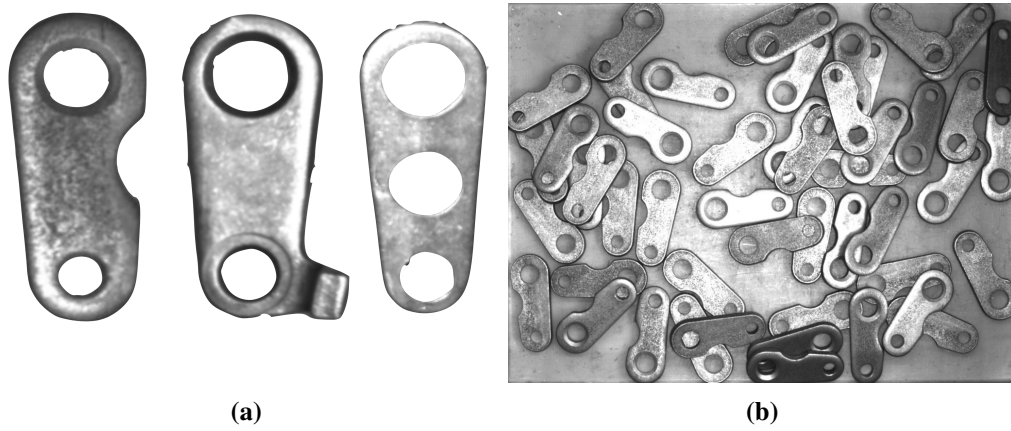


Figure 7.5: The datasets contain three different cranks (a). Example image (b) [Taheritanjani et al., 2020].

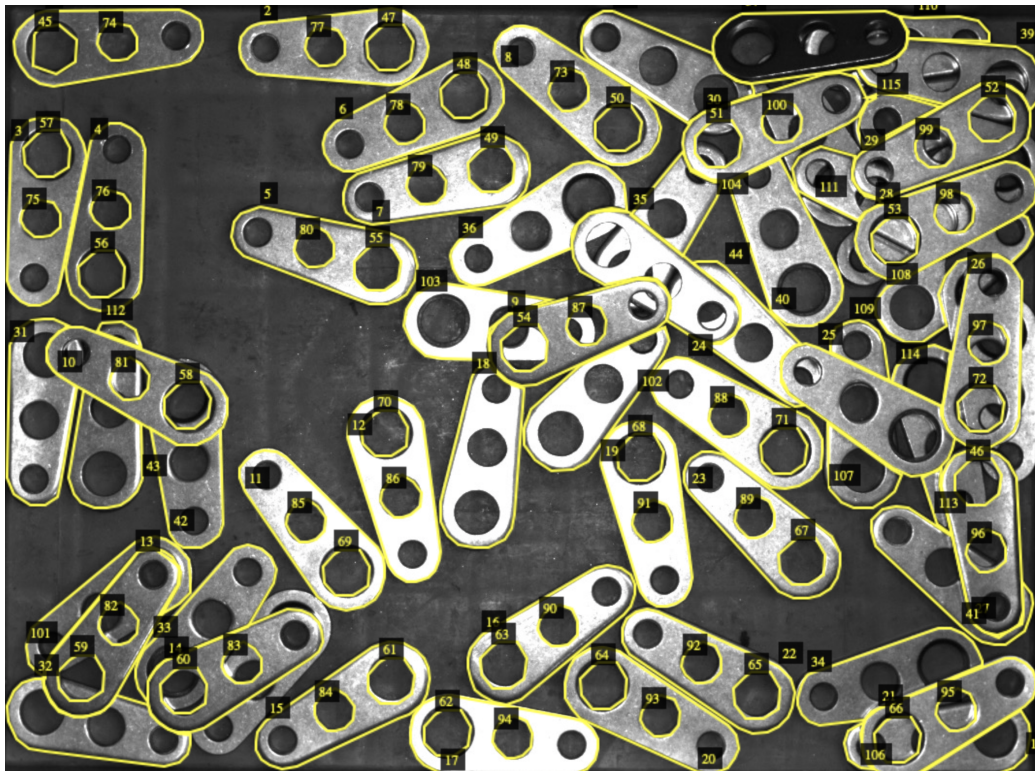


Figure 7.6: Example annotated image in the dataset. While only the outer contour of unpickable cranks are annotated, we also annotated two inner holes of pickable instances, and their outer contour. The pickable and unpickable cranks were labeled as different classes [Taheritanjani et al., 2020].

Table 7.3: Parameters and settings used to train Mask R-CNN model.

# of epochs	Learning Rate	Augmentations
15 (heads)	0.01 (epochs 1 to 15)	10% width and height shift, flip left to right, flip up to down, Blurring, random rotation from 0 to 270°, range of 0.7 to 1.3 light intensity
45 (all)	0.001 (epochs 16 to 60)	

taken over all classes, including the background. Table 7.3 gives an overview of the training parameters and settings.

For comparison, we trained two types of segmentation models. First, we used images of each crank to train a model for each crank, i.e., three models for three different cranks. Second, we used all cranks images, together to train a single model for all of cranks.

7.2.3 Method

As shown in Figure 7.7, we computed the orientation and grasp point of pickable cranks using the cranks' masks, and evaluated the performance of each of these methods.

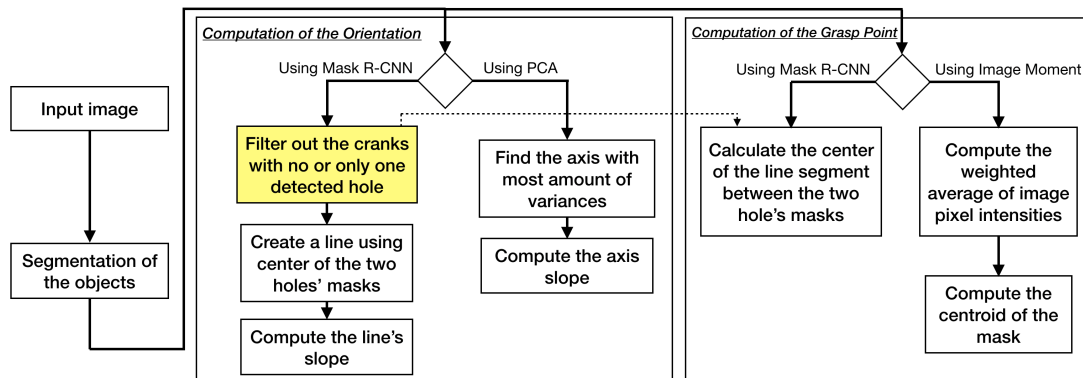


Figure 7.7: Overview of the required methods for fasteners bin picking in overhaul processes. The computation steps to calculate the orientation of the pickable cranks, using Mask R-CNN or PCA, and the grasp point of the pickable cranks, using Mask R-CNN or Image Moment. The mask filtering algorithm (marked in yellow) must be calculated only once, for both the orientation and the grasp point computations.

Segmentation

Using the trained Mask R-CNN models, we segmented the cranks in the images. We annotated pickable and unpickable cranks in the train and validation

datasets; therefore, the trained models could segment the pickable cranks. We did not use the computed bounding boxes of the results. Only the segmentations were used to feed into the next steps to compute the orientation and grasp point. Figure 7.8c shows an example of the detected pickable cranks.

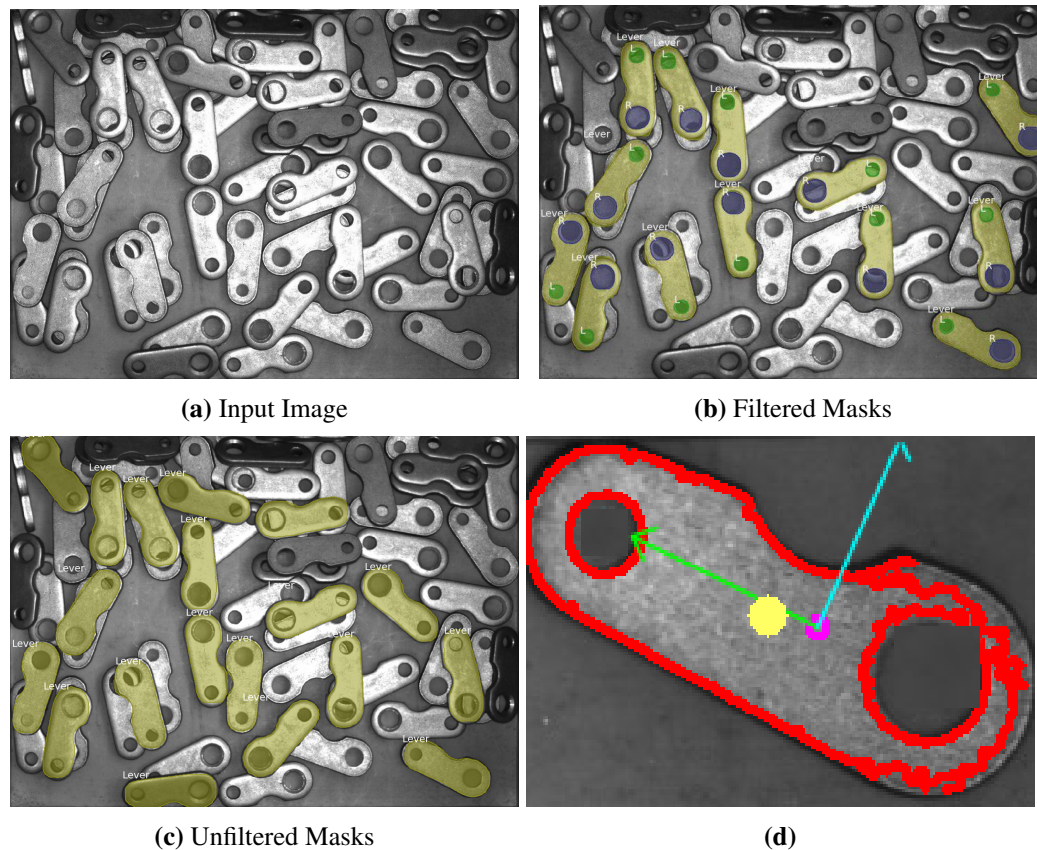


Figure 7.8: Example original input image (a), the visualization of the pickable cranks after filtering the masks, using only Mask R-CNN (b), and the visualization of all of the found pickable cranks using Mask R-CNN (c). Using the results from (c), we apply PCA and Image Moment to find the orientation (green and blue pivots) and the grasp point (yellow circle) for the lever on the lower right corner of the input image (d) [Taheritanjani et al., 2020].

Orientation Computation

We considered the orientation of a crank as the angle between the hypothetical horizontal line and the line between the center of its holes, i.e., the slope of the line between the center of the holes. For example, all cranks in Figure 7.5a have an orientation of 90° . To compute the orientation of the pickable cranks, we considered using either Mask R-CNN or PCA (Figure 7.7). Note that, we annotated

Algorithm 1 Filtering out cranks with none or only one hole detected (marked with yellow in Figure 7.7) [Taheritanjani et al., 2020].

```

1: for crank_ROIs do
2:   mask  $\leftarrow$  binary_fill_holes(crank_ROI)
3:   pickablecrank.insert(tempcrank)
4:   pickable1st.insert(temp1st)
5:   pickable2nd.insert(temp2nd)
6:   tempcrank, temp1st, temp2nd  $\leftarrow$  null
7:   for hole1_ROIs do
8:     hole1_mask  $\leftarrow$  binary_fill_holes(hole1_ROI)
9:     if mask = mask or hole1_mask then
10:      tempcrank  $\leftarrow$  crank_ROI
11:      temp1st  $\leftarrow$  hole1_ROI
12:      go out of the loop and continue
13:   if tempcrank is null then
14:     continue with the next crank_ROI
15:   for hole2_ROIs do
16:     hole2_mask  $\leftarrow$  binary_fill_holes(hole2_ROI)
17:     if mask = mask or hole2_mask then
18:       temp2nd  $\leftarrow$  hole2_ROI
19:       go out of the loop and continue
20:   if temp2nd is null then
21:     tempcrank, temp1st  $\leftarrow$  null

```

two of the inner holes of the pickable cranks in the dataset. Therefore, we can filter the cranks to find their orientation, where the crank's outer mask contains two inner holes. To filter out the masks, we filled the detected masks' matrices of the cranks and the detected masks' matrices of the holes with 1s, and performed a bitwise OR operation on them. If their bitwise OR is equal to the detected mask's matrix (filled with 1s), the crank mask contains the hole. Algorithm 1 shows the pseudocode to filter out cranks with none or only one hole detected. A filtered segmentation mask is shown in Figure 7.8b. Detection of the inner holes of the crank make it possible to calculate its orientation by computing the slope of the hypothetical line segment between the center of the masks of the holes.

The other approach to calculate the orientation is to use PCA. Figure 7.8d visualizes the two axes computed by PCA for an example crank. By selecting the axis that minimizes the mean squared distance between the input pixels and their projection onto it and calculating the axis's slope, we can compute the orientation of the crank (the green axis in Figure 7.8d).

Grasp Point Computation

We considered potential grasp point as the middle of the line segment between the center of crank's two holes. For cranks with three holes, we used position between their big and middle holes. To compute the grasp point of pickable cranks, we can use either Mask R-CNN or Image Moment (Figure 7.7). Section 7.2.3 discussed how we filter out the cranks with none or only one detected hole. Using the holes' masks of the pickable cranks, we find the middle point in the hypothetical line segment between the center of the holes' masks, which can be used as a crank's grasp point.

The other approach to compute the grasp point is to use the Image Moment. In Figure 7.8d, the yellow circle shows the grasp point calculation result obtained using Image Moment for an example crank.

7.2.4 Evaluation

To evaluate the described approaches, we used a fixed monocular camera and workstation to capture 20 different images for each crank type (60 images in total). Each image contained five to 50 cranks laid randomly on the workspace. In total, these images had 581 pickable cranks, which were counted manually to compare and measure the results. We evaluated all images with the methods discussed in the Sections 7.2.3, 7.2.3, and 7.2.3.

The orientation errors are calculated in rotation angle (the angle in radian by which the computed orientation must be rotated to reach the actual orientation). The grasp point errors are the distance between the computed grasp point and the actual grasp point (Δd in pixels).

In many bin picking studies, the results were compared using the mean IoU [Wang et al., 2018] [Chen et al., 2018] [Chen et al., 2017]. We also used accuracy, precision and recall relative to the detected non-occluded flat laid objects in the images to evaluate the segmentation results, and we used the mean absolute error to evaluate the orientation and grasp point computations. The ground truth values for the orientation and grasp point were acquired manually for comparison. For example, Figure 7.8a shows 22 cranks that are not occluded and laid flat on the surface and 38 unpickable cranks (occluded and/or non-flat). In Figure 7.8c, the trained Mask R-CNN model detected 16 pickable cranks correctly (true positive) and three incorrectly (false positive). Therefore, the accuracy of the segmentation method was 85%. In Figure 7.8d, the orientation error is 0.052 radians and the grasp point error is 13 pixels.

Table 7.4: Overview of segmentation results.

# of crank types	Mean IoU	Average Accuracy	Precision	Recall
1	96%	77%	77%	61%
3	97%	80%	79%	69%

Table 7.5: Overview of orientation and grasp point calculation results.

Approach	% of Detected Pickable Cranks	Orientation Computation Error (radian)	Grasp Point Computation Error (pixel)	Calculation Time (second)
Mask R-CNN	53%	0.033	11	5 to 20
PCA & Image Moment	70%	0.113	21	1 to 3

7.2.5 Results and Discussion

Tables 7.4 and 7.5 summarize the segmentation, orientation, and grasp point computation evaluation results. The model trained with all three cranks obtained slightly greater mean IoU compared to individual models trained using only one crank type. The model trained with all three cranks demonstrated higher accuracy, precision and recall than the models trained with a single crank type.

Using Mask R-CNN to filter out cranks with no or only detected hole resulted in low accuracy in some images, especially where there were very few pickable cranks in a large pile. Using PCA and Image Moment to compute the orientation and grasp point showed a 17% greater percentage of detected pickable cranks. Despite that, its orientation and grasp point computation errors were also higher (around three times greater for the orientation computation and approximately two times higher for the grasp point computation).

The method that use PCA and Image Moment to compute the orientation and grasp point ran up to six times faster than the Mask R-CNN based method. Note that we examined calculation time only for comparison purposes. The time can differ depending on different data types, libraries, implementation details, and hardware.

The Mask R-CNN approach to compute orientation and grasp point uses the mask filtering algorithm to filter out cranks with no or only one detected hole. One reason for the higher percentage of detected pickable cranks using PCA and Image Moment may be due to the surface of the underlying cranks being visible through the inner holes of the pickable cranks. In such cases, the model can fail to detect the holes. In addition, if we use the Mask R-CNN to compute the orientation, the crank's large and small holes are detected as different classes. Therefore, we can always maintain the same orientation calculation for all pickable cranks (the

slope of the line, when we start from the center of big hole towards the center of the small hole). Using PCA, it is also possible to compute the orientation towards the big hole, by splitting the crank relative to the second axis obtained by PCA, and apply PCA individually on each split. The split that has a bigger 2nd eigenvalue has also the bigger hole part. However, the drawback of using PCA is that it may fail to find the bigger hole, if the surface of the underlying cranks are visible through the small hole of the pickable cranks. Thus, the relative direction of the axes to the crank's big and small holes can be flipped for different images. In cases where the surface of the underlying cranks are not visible through the small hole of the pickable crank, PCA computes one feature axis towards the big hole. In other case, PCA may compute one feature axis towards towards the small hole. The unpredictable calculation of the feature axis can make this approach not robust for rare cases, when the robot must place the part in a specific position in the container. In this case, the Mask R-CNN based approach can be used, where the detected inner holes' segments are used to compute the orientation precisely.

Note that the calculation time has a direct correlation with the number of the cranks in the image. The Mask R-CNN model requires approximately 20 seconds to find segmentations. The mask filtering process in Algorithm 1 requires an order of $O(n^2)$ to compare filled masks. Therefore, using the detected inner holes to compute the orientation and grasp point, this approach requires more calculation time compared to the PCA and Image Moment based approaches.

We computed PCA and Image Moment on the ROI of the masks output from Mask R-CNN. Here, higher orientation and grasp point computation errors when using PCA and Image Moment is primarily due to the computations on the entire crank's ROI rather than the crank's Mask. Moreover, the results indicate that training only one model for all crank types performs better than training a single model per category. This can be due to having a relatively low amount of data for the training (only 10 images per crank type). With only 10 annotated images for training, the dataset cannot represent all positions and orientations of the objects in the target environment. Therefore, more annotated images should be employed to realize better performance and generalization.

Many 3D CAD-based solutions³ enable LED lightning technology to generate 3D point clouds. Without relying on these type of sensors, this approach can cost up to 40 times less. The main drawback of the proposed approaches is that objects in the images must be annotated manually for training. Due to the cluttered order of the objects and noise (light reflection and shadows), it is very challenging to obtain acceptable annotations using automatic contour detection methods, such as Canny edge detection [Liu et al., 2012]. However, we can rerun the computations

³<https://www.isravision.com/en/ready-to-use/robot-vision/bin-picking/>

of the pickable cranks after some are picked by the robot. Moreover, Danielczuk et al. showed the usefulness of pushing actions to separate objects and move them away [Danielczuk et al., 2018]. Since picking objects from the pile results in a new scene in the image, it is possible that using cranks that are no longer occluded may allow us to obtain new segments for new pickable cranks. Therefore, the results of this study imply practical usage of automatic bin picking for packaging in overhaul processes. To compare, Table 7.6 shows the comparison between the orientation and grasp point computation approaches for bin picking used in this dissertation and the state of the art results with regards to the rating scheme, described in Section 1.2.

Table 7.6: Comparison between the orientation and grasp point computation approaches for bin picking used in this dissertation and the state of the art results in the orientation and grasp point calculation for bin picking with regards to the rating scheme.

Study	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆
[Mahler et al., 2017]	±	+	±	-	±	±
[Schwarz et al., 2018]	±	+	±	±	-	±
[Wu et al., 2019]	+	+	±	-	+	±
Orientation and grasp point computation approaches for bin picking used in this dissertation	+	+ (PCA & Image Moment) ± (Mask R-CNN)	±	+	±	±

Chapter 8

Validation

In order to validate H_2 and H_3 , we used an illustrative case study approach [Yin, 2011] with two industrial demonstrators¹. Section 8.1 describes the *SPARCS* (Small PARTs Classification System) that utilizes the automatic classification approach mentioned in Chapter 6 to classify bolts. It also uses the 2D bin picking approach described in Section 7.1 to place the bolts in the compartments. Section 8.2 describes *SUMA* (SURface MAintenance) that employs the automatic damage detection methods to identify dirt, scratches, and dents on surfaces prior to automatic grind and polish. There is no public available reference to compare the automatic approaches results with the current manual classification results, in particular a benchmark as a basis for the comparison between the automatic and manual approaches. Section 8.3 describes how we obtained benchmarks to compare with the automatic methods in sections 8.1 and 8.2. Section 8.4 compares the benchmarks with the automatic approaches and discusses the findings. Finally, Section 8.5 discusses the threats to validity.

8.1 SPARCS

This section describes the project *SPARCS*, which was funded by Verein Deutscher Ingenieure (VDI)² to automatically classify the small parts (fasteners) which are collected from airplane engines. The goal of the project is to build a system that automatically detect known and unknown classes of fasteners used in airplane engines, and sort them inside the system. Figure 8.1 shows *SPARCS*'s use case model.

The rest of this section organized as follows: Subsection 8.1.1 describes the demo scenarios. Subsection 8.1.2 presents the platform which was designed for

¹ H_1 was validated in chapter 4

²<https://vdivde-it.de/>

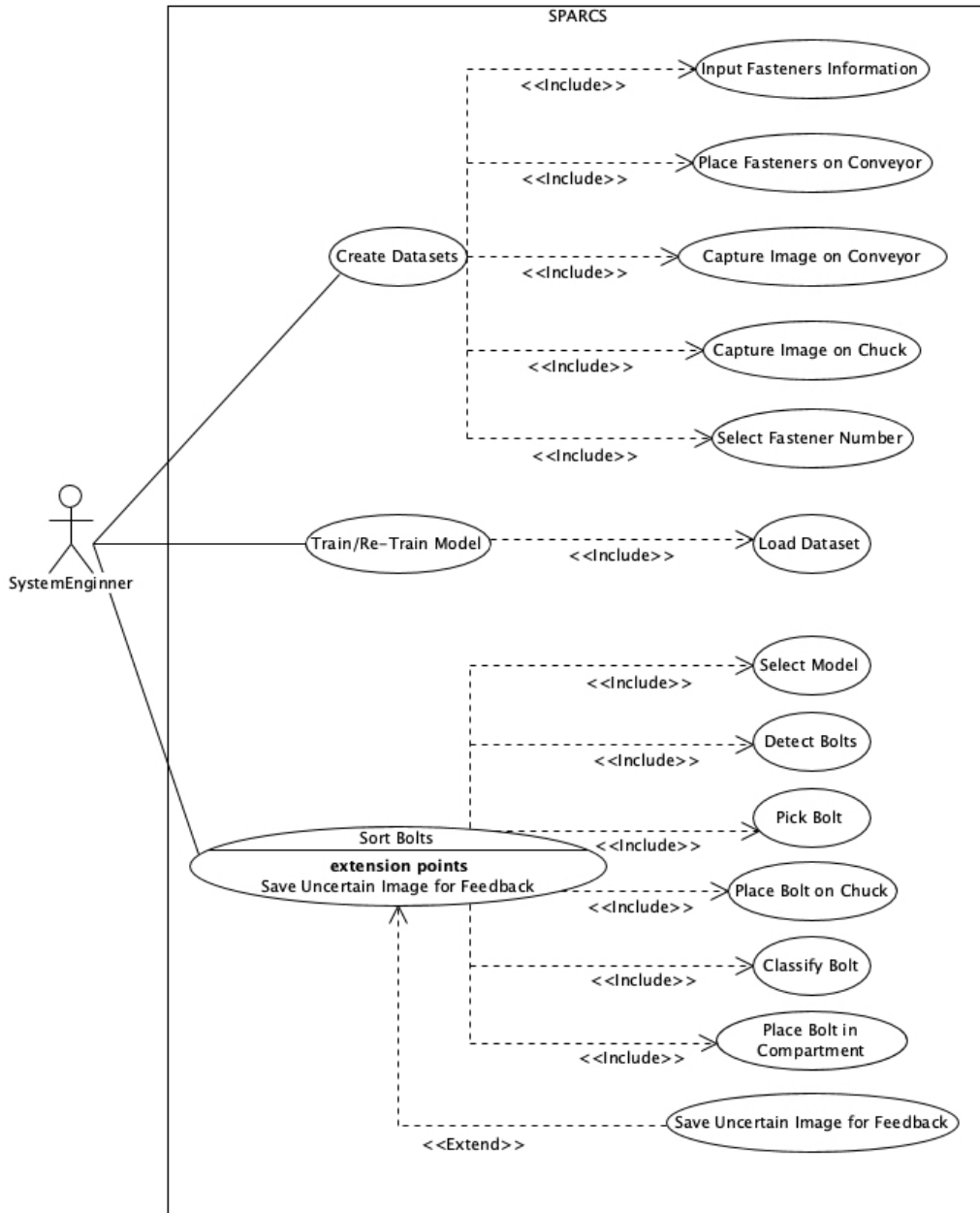


Figure 8.1: SPARCS use cases (UML use case diagram).

this project, including all software and hardware components, and subsection 8.1.3 presents the results of the evaluation.

8.1.1 Scenarios

A scenario informally describes how one of the actors interact with a system [Bruegge and Dutoit, 1999]. This section describes the scenarios that are implemented in *SPARCS*.

The scenario *CreateDatasetForComponent* describes the following steps after a *SystemEngineer* receives a container of fasteners that are not yet captured in a dataset. First the *SystemEngineer* reads the catalog belonging to the container where they find the serial number of the component it comes from, and the serial numbers of all the fasteners associated with that component. The *SystemEngineer* input the information into *SPARCS* and then place the container fasteners on the conveyor belt. *SPARCS* detects the fasteners under the camera and captures images. Afterwards, *SystemEngineer* selects the correct serial number from a list. The images together with their label is finally stored in the dataset. The same actions apply to capturing images of fasteners heads on the chuck. Figure 8.2 shows the simplified activity diagram for this scenario.

The scenario *SortBolts* shows how a *SystemEngineer* initiates the sorting process for a specific component. Firstly, the *SystemEngineer* selects the model corresponding to the component and places the container fasteners on the conveyor belt of *SPARCS*. At each predefined interval, the camera captures an image and detect if the image belongs to the category of bolts or not. If the image is a bolt, then *SPARCS* uses a robotic arm to pick the bolt and place it on a chuck to capture the image of its head from above. The two images are then fed into *SPARCS* for classification. *SPARCS* will predict the type of each bolt and instruct its robotic arm to place the fastener in a compartment corresponding to its predicted serial number. Figure 8.3 shows the activity diagram for this scenario.

RetrainModel describes how the *SystemEngineer* adds new data to an already existing model to further improve the classification accuracy. After selecting the pretrained model, the *SystemEngineer* follows the steps described in scenario *CreateDatasetForNewComponent*, to create a dataset. The *SystemEngineer* loads the new dataset into *SPARCS* and initiates the retraining. *SPARCS* trains on the new data and outputs the classification score as the model trains.

OnlineLearning describes the scenario in which *SPARCS* benefits from the expert knowledge to enhance its classification accuracy. During *SortBolts* scenario, *SPARCS* receives images from the fasteners and returns the classification result as prediction scores (i.e., percentages for each class). If the highest prediction score is lower than a predefined threshold, it means that *SPARCS* is not certain about the class of the fastener. Therefore, it saves the image in a folder that will

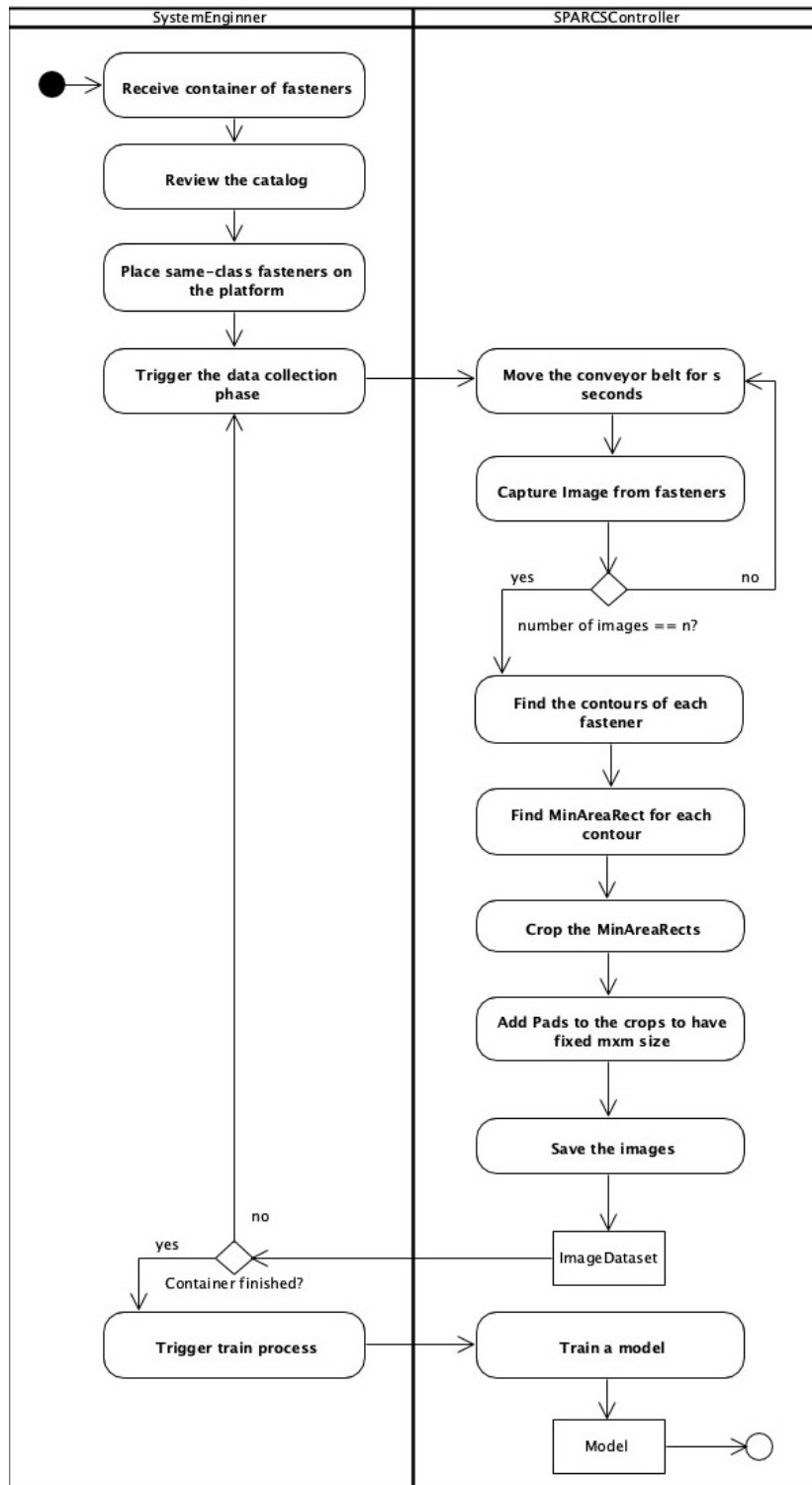


Figure 8.2: Activities in *CreateDatasetForComponent* scenario of SPARCS (UML activity diagram).

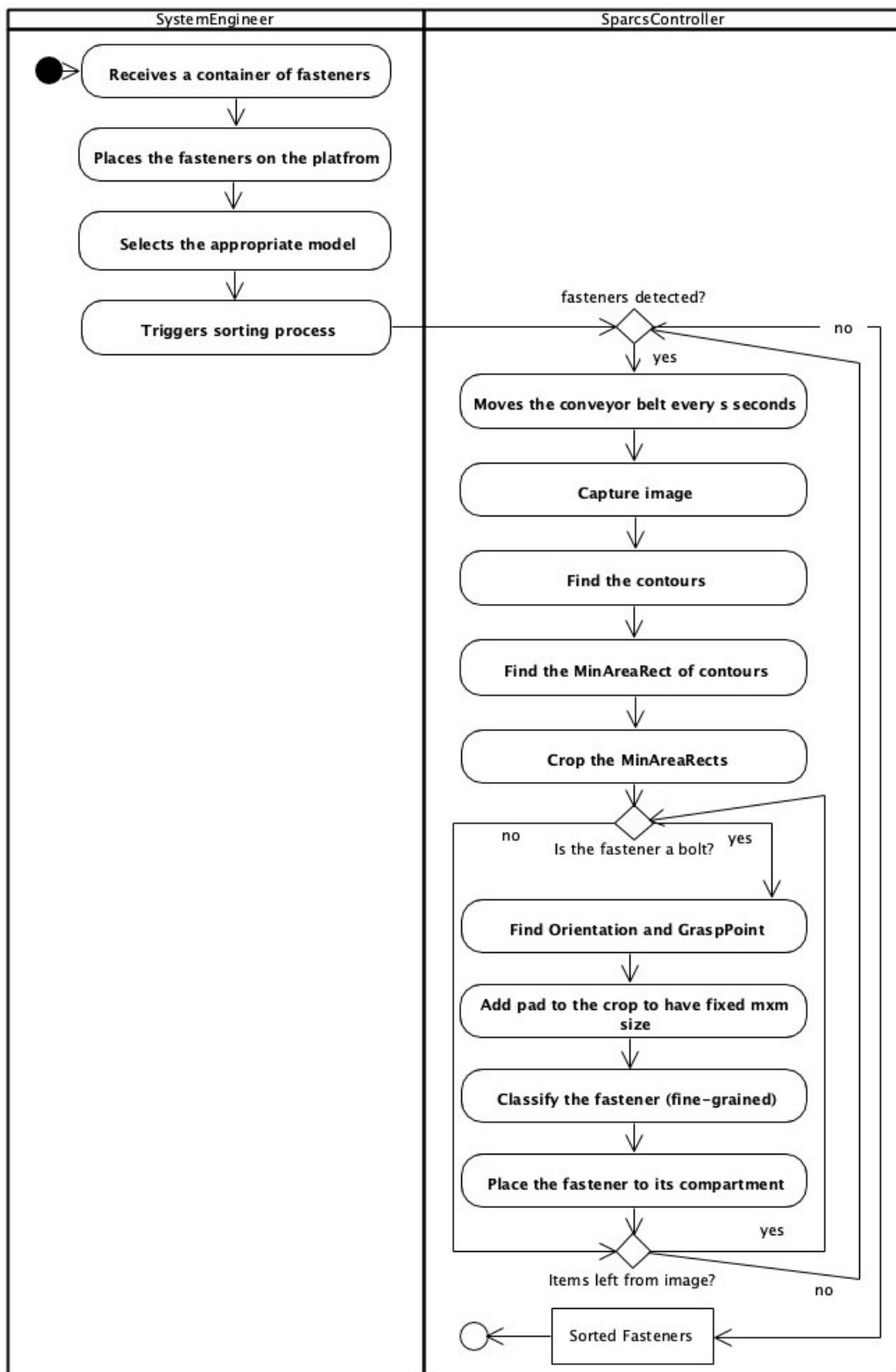


Figure 8.3: Simplified activities in *SortBolts* scenario of *SPARCS* (UML activity diagram).

be checked and labeled by a technician expert to be used as part of the dataset in *RetrainModel* scenario.

8.1.2 SPARCS Platform

SPARCS was developed as an evolution of iterations. The first iteration was a prototype that utilized a rotating glass table covered with opal foil, which was supported with backlighting. The opal foil and glass surface of the table acted as *PPIC-A*'s conveyor belt texture (light diffusion textile). This prototype was used at the lab for research purposes. Figure 8.4 shows this prototype.

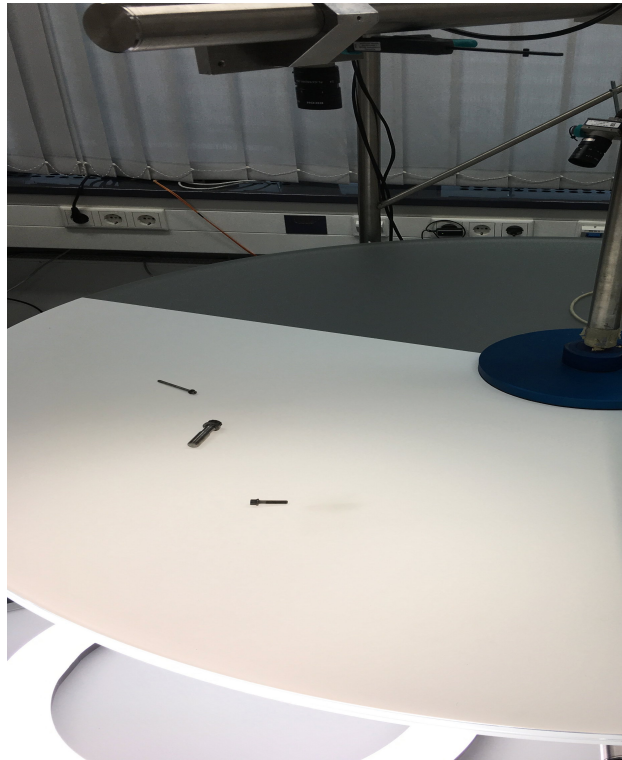


Figure 8.4: The realization of *PPIC-A* with a rotating glass table covered with opal foil and supported with backlighting, used in the first iteration of *SPARCS*.

Although the first iteration utilized *PPIC-A* design, due to the slippery surface of glass table, the fasteners could roll easily, which made the table movements impractical. Considering the slippery surface and being unable to move the rotating glass, capturing images to create datasets was a slow and tedious process. The second iteration was designed to address this issue. It utilized a vibration plate with LED backlighting. The surface of the plate was not slippery and we were able to use vibration to create dataset for a fastener. In addition, we could use

the vibration to see different sides of the fasteners during damage detection. Figure 8.5 shows the second iteration of the hardware platform, with one camera on top and the PLC controller for the vibration plate, which was used at the research lab to conduct further researches.

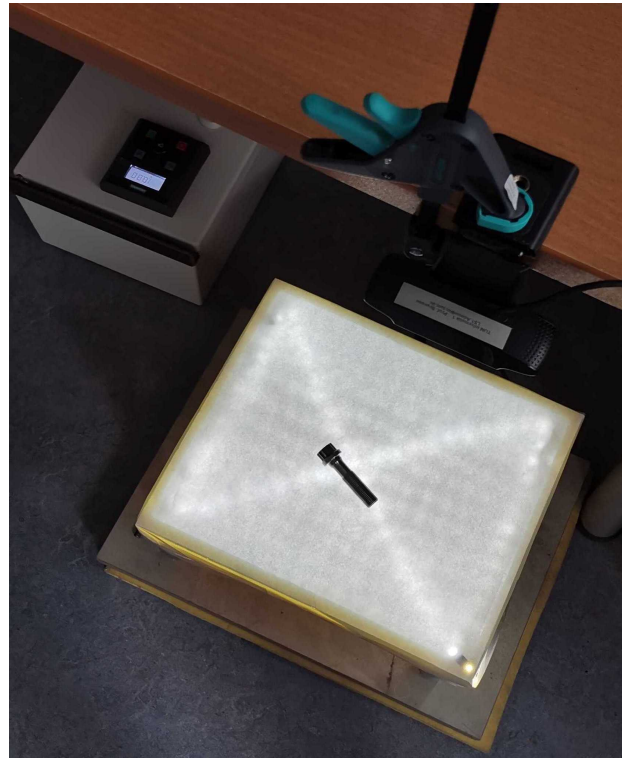


Figure 8.5: The realization of *PPIC-A* with a vibration plate and LED backlighting, used in the second iteration of *SPARCS*.

The current solution of *SPARCS* utilizes *PPIC*, described in Section 3.2. It was designed to include the robotic arm and compartment storage to automate the classification and bin picking tasks (see Figure 8.6). It uses two conveyor belts, one for bringing the fasteners under the camera and another one to return the leftover fasteners to the beginning. It also uses additional components, such as a chuck, a robotic arm and compartment storage. The chuck is used to obtain an image from the bolt heads and use it as the second view for multi-view classification. To lower the cost of configuration, webcams are used as *PPIC* cameras.

We placed the webcam as close as possible to the conveyor belt surface, to ensure the captured images have an acceptable sharpness needed for the classification. To ensure that the robotic arm does not collide with the camera during the sorting process, the camera was placed on the side of conveyor belt and tilted it to be able to capture the whole width of the conveyor belt.

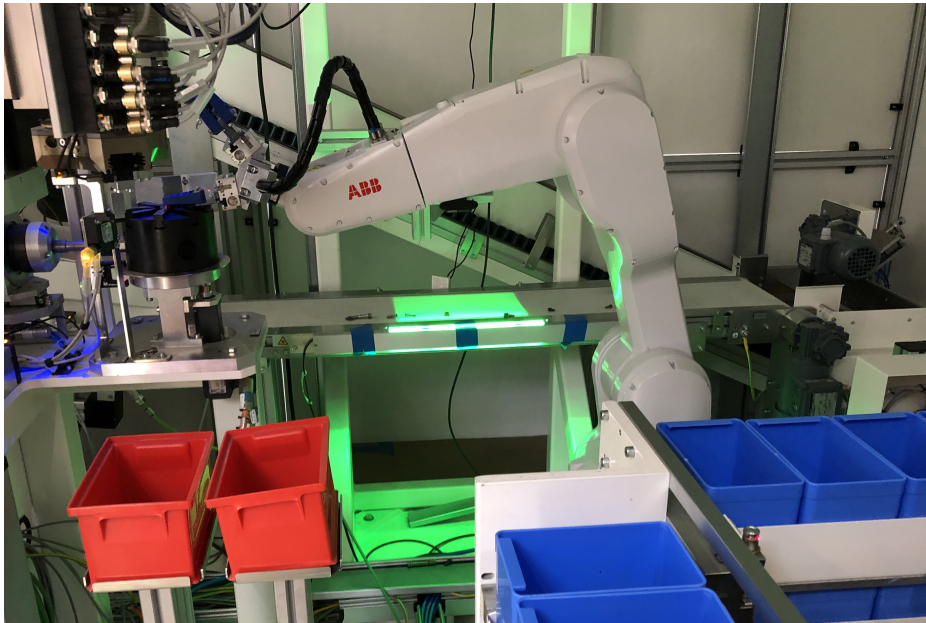


Figure 8.6: The realization of *PPIC* with a vibration conveyor belt with light diffusing textile and backlighting, used in current solution of *SPARCS*.

The *SPARCS* computer controls the cameras and triggers all the preprocessing and classification methods, which are hosted on the same machine. Controlling the conveyor belt, chuck, robotic arm and the compartment storage is performed by the PLC controller. Figure 8.7 shows the hardware/software mapping of the system.

The platform used for *SPARCS* uses a conveyor belt with light diffusing textile and backlighting. Therefore, it ensures a uniform background during processing of images for classification and bin picking. We used the methods described in Section 7.1 to compute the grasp point and orientation of the fasteners. We also trained a bolt/not-bolt classifier to detect if a contour is a pickable bolt. We considered a bolt is pickable, if it is not occluded by any other fastener and there is a certain distance between its contour and its neighbor fasteners³ This ensures minimal collision between the robotic arm and other fasteners on the conveyor belt. This minimal collision between robotic arm and other fasteners is necessary to reduce the image processing computations and avoid picking any unwanted fastener.

Since the camera is tilted, we projected the obtained images using geometric transformations in OpenCV⁴. The projection addressed the perspective issue

³In the implementation we set this distance to 3 cm.

⁴https://docs.opencv.org/trunk/da/d6e/tutorial_py_geometric_transformations.html

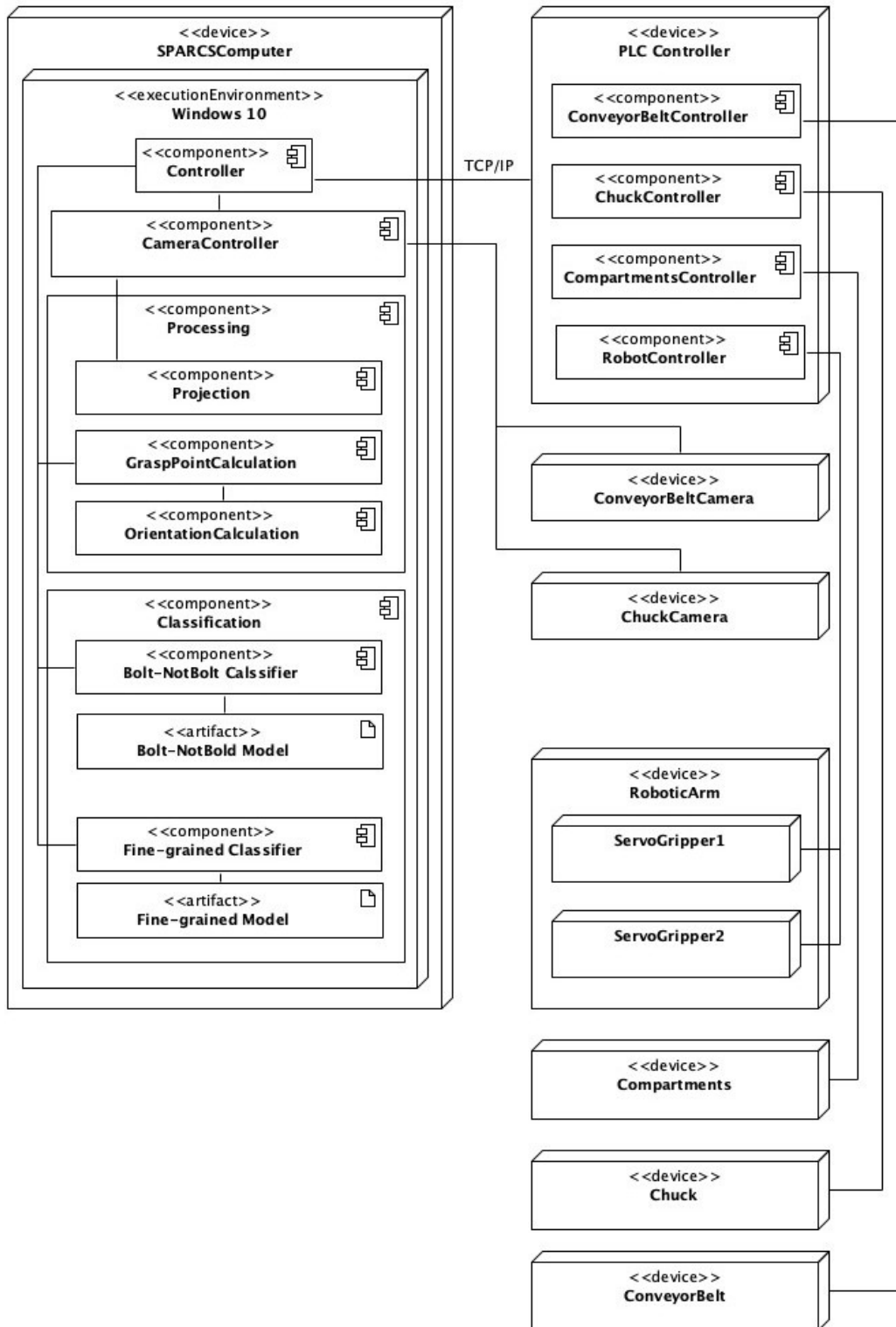


Figure 8.7: SPARCS hardware software mapping (UML deployment diagram).

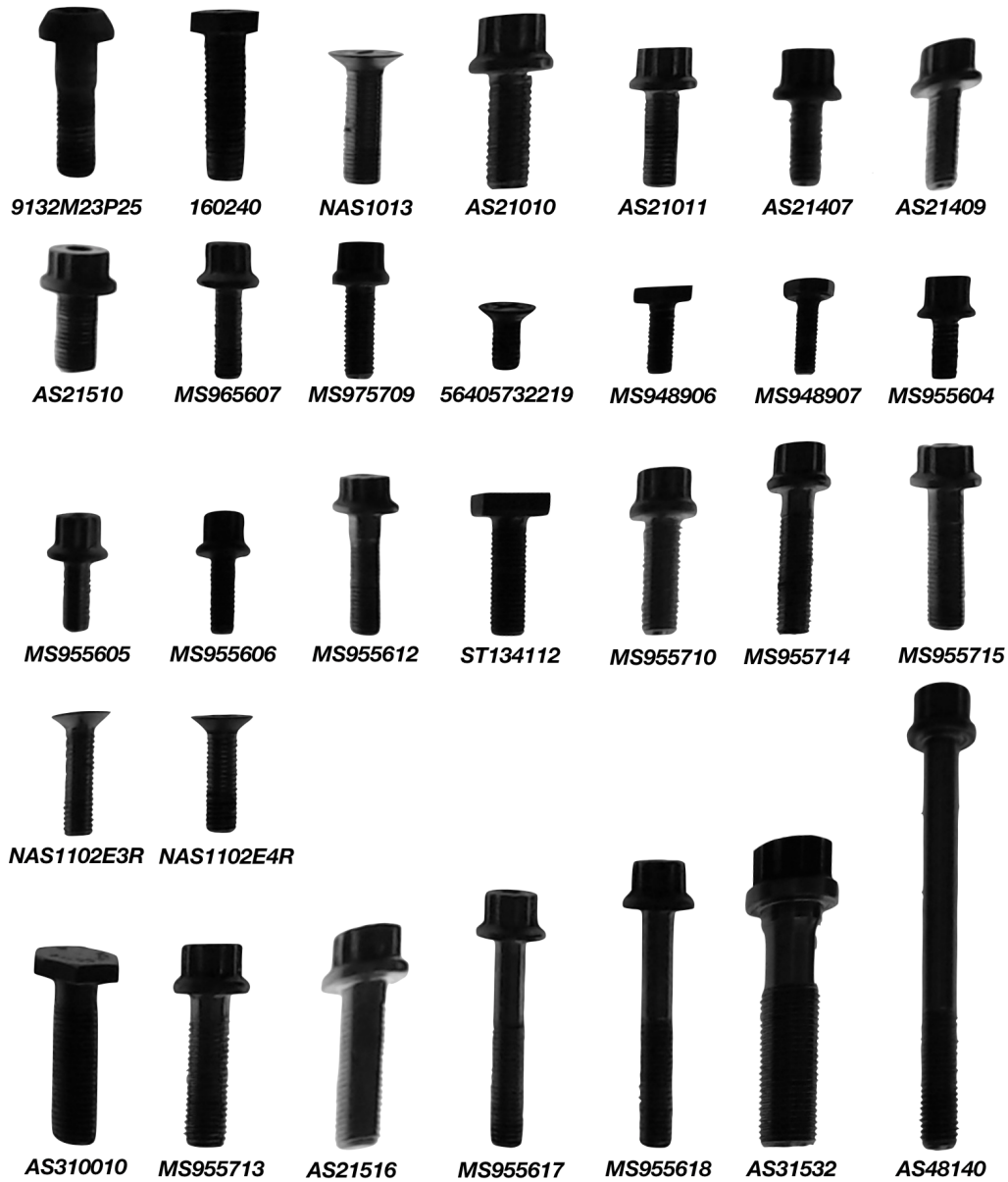


Figure 8.8: Overview of the 30 bolts used in *SPARCS* dataset. Some bolts may look distorted due the image projection prior to dataset capturing.

during having fasteners on different positions of the conveyor belt. Finally, after grasping the bolt, the robotic arm places it on the chuck (heads-up) that we capture an image of its head. Together with the image obtained on the conveyor belt, the two images are sent to multi-view classifier for the fine-grained classification.

We trained a multi-view classification model for 30 sample bolts, obtained from a specific airplane engine component. We also trained a single-view classification model, using the bolt images obtained on the conveyor belt, mainly to have a base line for comparison. Figure 8.8 shows an overview of the fasteners used in the dataset.

8.1.3 Results

This section describes the evaluation process and results for *SPARCS*. Tables 8.1 and 8.2 show an overview of the results obtained during bin picking and fine-grained classification.

Table 8.1: Overview of bolt/not-bolt model, and grasp point and orientation calculation results for 30 classes of bolts.

Accuracy	Precision	Recall	Orientation Computation Error (radian)	Grasp Point Computation Error (pixel)
94%	92%	100%	0.03	11

Table 8.2: Overview of classification results for 30 classes of bolts.

Model	Accuracy	Precision	Recall
Single-View	87.5%	85.3%	91.2%
Multi-View	79.1%	80.3%	82.8%

The results shown in Table 8.1 are similar to the findings in Section 7.1.4. The single-view model accuracy in Table 8.2 is better than multi-view model accuracy. As described in Section 6.2.3, the poor performance of a multi-view model compared to a single-view model can be justified as the low amount of discriminative features in the other views used by the multi-view model (in this case, the image obtained from the bolt head on the chuck view). Therefore, a single view CNN is able to learn the difference between highly similar bolts, but only given a view which contains almost all the discriminative features. Figure 8.9 shows the images obtained from the heads of different bolts and images obtained on the conveyor belt. Given a less than optimal view, such as the chuck view in *SPARCS*,

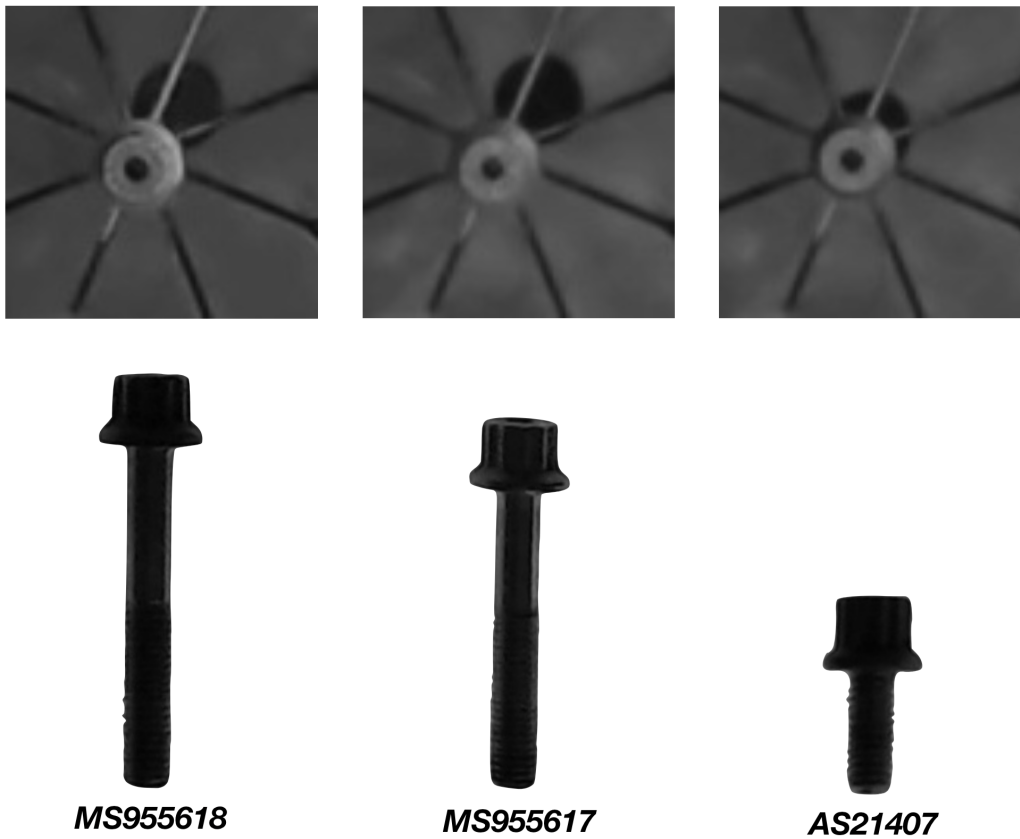


Figure 8.9: Images of the heads of three bolts (top) and their respective side view (bottom). The similarity of the heads reduces the accuracy of multi-view model after aggregating the results into an overall classification.

the model struggles as there is not enough discriminative features visible to distinguish between them. The Multi-CNN approach does not address this problem, because there is no information sharing between the models.

Due to the better performance of the single-view classification model, and to reduce the overhead time required to place the bolt on the chuck to capture the bolt head, we decided to use the single-view model for the demo. For training with all the 30 fasteners in the dataset, we ultimately achieved 87.5% accuracy over the test dataset which contains 20 images of each fastener.

8.2 SUMA

This section describes the project *SUMA*, which was done in the practical course at the chair for applied software engineering at technical University of Munich in the summer semester 2019⁵ to automatically detect and maintain the damages and dirt on the surface of components.

Maintenance of surfaces represent a classic example of surface detection problems. Currently, skilled workers only decide which tools should be used for the whole process of the surfaces. The lack of enough skilled workers and the need to protect them against pollutants and injuries highlight the main issues in current process. In addition, the working environment must be in a way that an ergonomic work is possible.

The goal of the project *SUMA* is to build a partially automatic system to assist the maintenance of surfaces. It must independently recognize the shape of the 3D surface, and decide which tools are used for maintenance. The challenges include dealing with an explosion-prone environment due to large amounts of fuel. With the increasing pollution of waters all over the world, the project contributes to a more sustainable use of watercraft for work and leisure. Figure 8.10 shows *SUMA*'s use case model.

Subsection 8.2.1 describes the problem. Subsection 8.2.2 presents the robot and hardware platform which was used in this project, and subsection 8.2.3 presents the results of the evaluation.

8.2.1 Scenarios

This section describes the scenarios that are implemented in *SUMA*.

The scenario *CreateDamageAndDirtDataset* describes the following steps after a *SystemEngineer* receives a new surface, with multiple dirt and damages on its surface. The *SystemEngineer* configures the camera and lighting, and captures images from the surface. Afterwards, *SystemEngineer* annotates the dirt and damages on the images. The images together with their annotations is finally stored in the dataset.

Detect3DSurface describes the scenario in which the *SystemEngineer* detects the 3D point clouds of the component surface prior to damage and dirt detection. In this scenario, he *SystemEngineer* initiates the image capturing using the 3D camera and loads the new image into *3DSurfaceDetection* software. *3DSurfaceDetection* return the point clouds of the surface which is used for path planning of the robot.

⁵https://ase.in.tum.de/lehrstuhl_1/component/content/article/106-teaching/1037

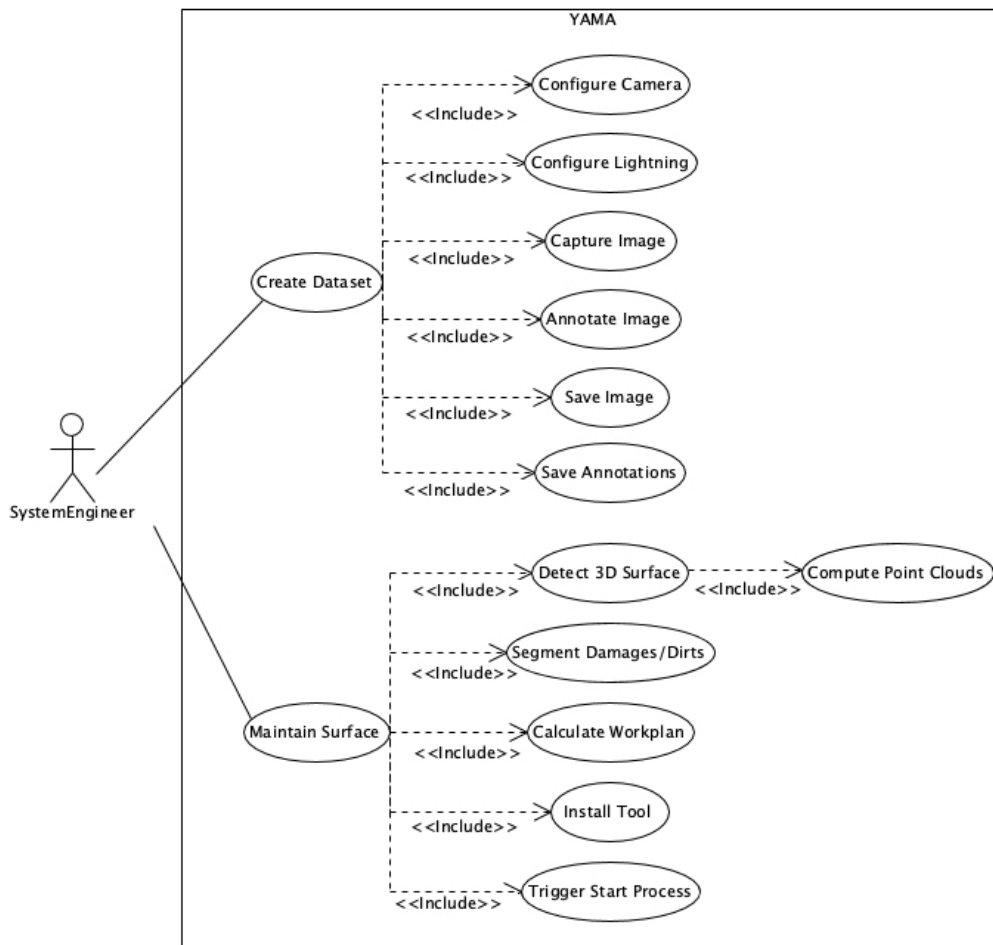


Figure 8.10: SUMA use cases (UML use case diagram).

The scenario *DetectDamageAndDirt* shows how a *SystemEngineer* detects damages and dirt on the surfaces of components. Firstly, the 2D camera captures an image of a part of surface. This image is fed into the *Classification* component of *SUMA* and the model detects if the image contains any damages or dirt.

MaintainSurface shows how the *SystemEngineer* uses the 3D point clouds and the masks of the damages and dirt on 2D image from previous scenarios and maintain the surface. After obtaining the results of *Detect3DSurface* and *DetectDamageAndDirt*, these data are sent to the *SUMA* app to create a workplan. The workplan includes the visualization of the 2D pictures with the found damages, dirt and dents, and the 3D image of the surface. The *SystemEngineer* is able to approve and decline the calculated workplan and also to delete or add bounding boxes for damages/dirt or dents. The *SystemEngineer* installs the proper grinding tool on the robotic arm and triggers the start process to which executes the

workplan.

8.2.2 *SUMA* Platform

This section describes the hardware platform used in *SUMA*. *SUMA* uses a robotic arm and a camera on top. The 3D camera has also stereo output to capture 2D images and the robotic arm has a PLC controller. Figure 8.11 shows the hardware platform and Figure 8.12 illustrates the hardware software mapping of *SUMA*.

SUMA uses the supervised method described in Section 4.2 to detect the damages and dirt on the surfaces. We collected 200 images from different damaged surfaces, such as damages on the car surfaces, and annotated them using the VIA tool. To reduce the reflection on the surface⁶, the images were collected using *PPIC-D* and three classes of annotations were defined: dirt (of any kind), scratches, and dents. We trained a Mask R-CNN model to detect the dirt and damages on the surfaces. Figure 8.13 shows the segmentation results obtained from the Mask R-CNN.

To be able to perform the path planning for the robotic arm, *SUMA* uses the point clouds obtained from the 3D camera. After receiving the depth image and extracting the point clouds, we applied point reduction algorithms to reduce the amount of calculations for next steps. At this point, we were able to extract the surfaces and apply surface smoothing algorithm to reduce the noises. Finally, we calculated the norms of the points⁷.

Using the norms, *SUMA* applies a path planning for the robotic arm. Together with the coordination of dirt and damages, these data are sent to workplan generator to create a workplan.

⁶Due to the size and location of damaged surfaces, we were not able to bring them to the lab for data collection. However, the same techniques and concepts used in *PPIC-D* was used to collect the data.

⁷A norm is a function that assigns a positive length or size to vectors in a vector space.



Figure 8.11: The hardware platform used in *SUMA*. The robotic arm is equipped with a 3D camera and sees the surface in front of it.

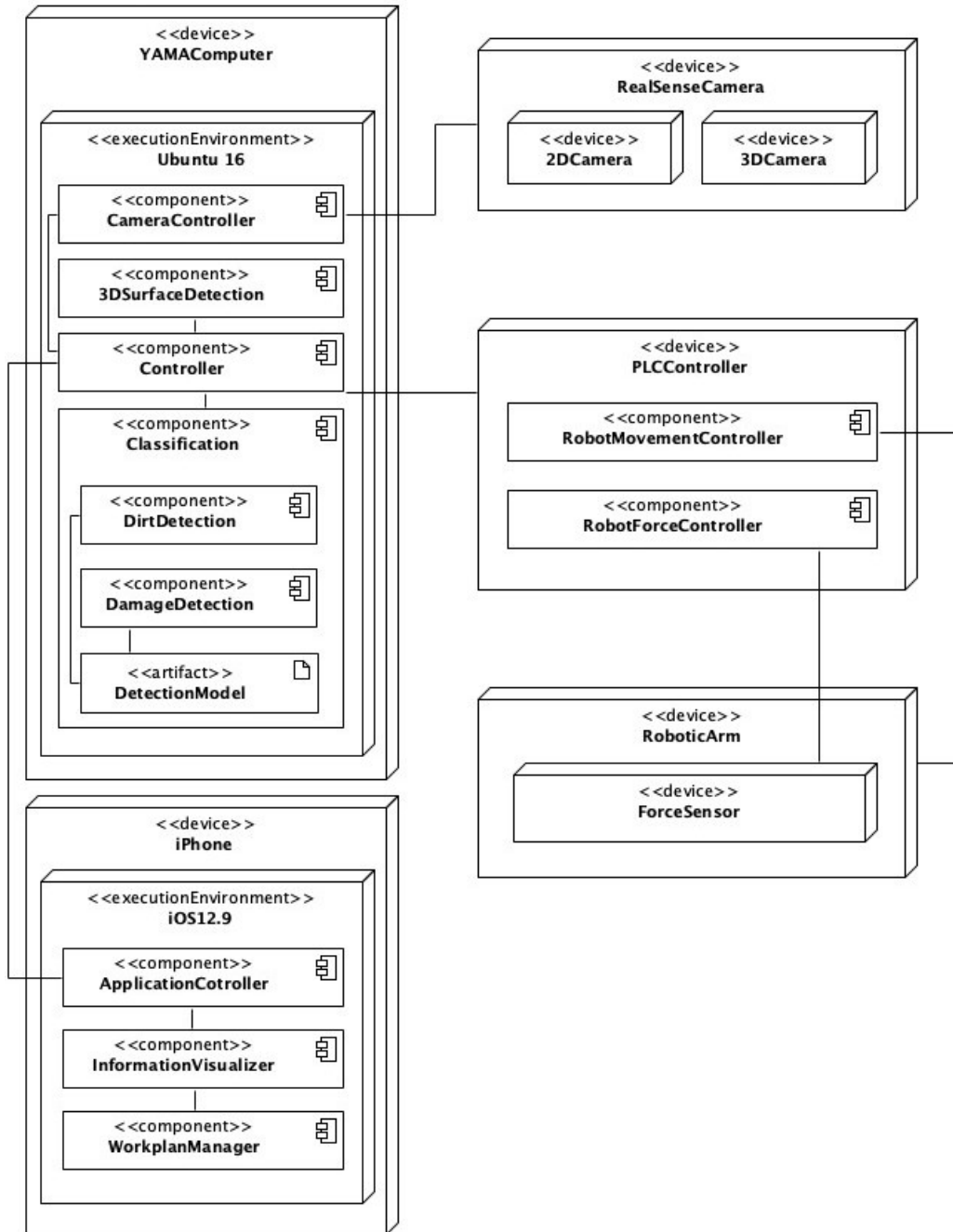


Figure 8.12: SUMA hardware software mapping (UML deployment diagram).

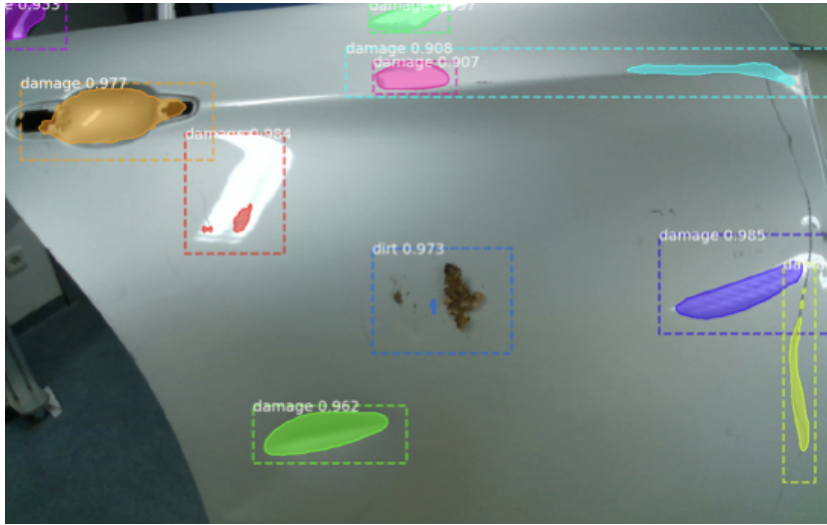


Figure 8.13: Example of segmentation results from the trained Mask R-CNN model.

8.2.3 Evaluation and Results

We separated the evaluation of *SUMA* into two sections: damage/dirt detection evaluation and path planning evaluation. Due to the stationary hardware platform of *SUMA* and unavailability of different damaged surfaces for processing and maintaining in our lab, we were not able to perform tests on different surfaces and different damages for path planning. Therefore, the main focus was to evaluate the damage/dirt detection. We used a test dataset of 40 images annotated with VIA tool, and measured the IoU for the detected bounding boxes and the ground truth annotations. Table 8.3 shows an overview of the results.

Table 8.3: Overview of instance segmentation results using Mask R-CNN for dirt and damage detection.

Accuracy	Precision	Recall	IoU
81%	81%	79%	51%

8.3 Benchmark Preparation

There is no public available reference to compare the results obtained in sections 8.1.3 and 8.2.3 with the current manual process. We performed interviews with 6 people from MTU Hanover⁸ and collected information on the process and

⁸<https://www.mtu.de/maintenance/>

Table 8.4: Three manual experiments performed to obtain benchmark as a basis for comparison in *SPARCS*.

#	# of Fasteners	# of Misclassified Items	Classification Time (Minutes)	Bin Picking Time (Minutes)
1	297	4	252	123
2	414	5	266	116
3	341	3	218	113
Total	1052	12	736	352

challenges of the manual fastener sorting process (including damage identification, classification and bin picking). We also observed the overhauling process at Schindler Handhabetechnik⁹. Due to the company non-disclosure agreements, the results of these interviews cannot be shared.

We performed the manual classification and bin picking three times, and collected the data for comparison. First, we manually classified a boxes with 297, 414 and 341 fasteners (the images of the bolts in the last box with 341 were used for *SPARCS* dataset. See Figure 8.8) and double-checked them to measure the accuracy. Table 8.4 shows the details of each experiment.

Only four, five and three fasteners were misclassified in each box. Nevertheless, the double-checking process must be performed for all the fasteners when they are checked for damages¹⁰. Therefore, we decided to use 100% accuracy, precision and recall for the manual classification. The whole manual classification process took 252, 266 and 218 minutes for each box. For placing the fasteners in their compartments, we needed 123, 116 and 113 minutes (for each box) to read the catalogs and fill in the compartments accordingly. Therefore, we needed on average about 58 seconds for classification and bin picking per fastener.

Similarly to *SPARCS*, there is no available reference to compare the segmentation results in *SUMA* with the manual damage/dirt detection. We performed 10 experiments, in which we spotted the damages/dirts on a car door surface (see Figure 8.11) and computed their coordination using moving the robotic arm to the spot prior to process the surface. To manually find the coordinations, we moved the robotic arm to the top left corner of a hypothetical bounding box around the damage/dirt, saved the location, moved the arm to the lower right corner of the hypothetical box and saved the location. For the first spot, it needed more time in compare with the next spots, due to conveying the robotic arm from the de-

⁹<https://www.schindler-handhabe.de>

¹⁰The double checking is automatically performed during the manual bin picking process, in which the technicians ensure the similarity of the fasteners prior to placing them inside the compartments.

Table 8.5: 10 manual experiments performed to obtain benchmark as a basis for comparison in *SUMA*.

#	# of Damages/Dirts	Duration (Seconds)
1	1	23
2	1	24
3	1	24
4	2	44
5	2	43
6	2	41
7	3	60
8	3	65
9	3	61
10	4	83
Total	22	468

fault position to the surface. On average, this process took about 21.3 seconds per detected spot. Table 8.5 shows the detailed information for the 10 experiments.

8.4 Comparison and Discussions

In *SPARCS*, the automatic approach needed half a second for bolt/not-bolt classification, less than half a second for orientation and grasp point calculation and about 8 seconds for bin picking. Comparing this with the time used in the manual approach shows that the automatic approach achieves about 6.5 times faster results.

The dataset creation and training time overheads are excluded from this comparison. To create datasets, we placed multiple instances of bolts on the conveyor and captured images from them. Afterwards, we cropped the minimum rectangle area of the detected contours in the image. The time spent for dataset creation depended on the number of available instances of fasteners (the more instances available, less time was needed to capture images due to having multiple samples of a fastener in a single image). After capturing images, there are a set of pre-processing step to create the final images for the datasets (see Figure 8.2). In total, we needed about 6 hours and 12 minutes for the whole dataset. The training time was also about 29 hours. On the other hand, in the manual approach, technicians must create 1:1 comparison-sketch for each fastener. The time to create the sketches is unknown to us. Technicians must also pass special training courses to prepare for job. This time could be varied per trainee. We conducted the manual

Table 8.6: Comparison between the manual and the automatic classification and bin picking for 341 fasteners.

	Manual	Automatic
Accuracy	100%	87.5%
Precision	100%	85.3%
Recall	100%	91.2%
Classification and Bin Picking Time needed per Fastener	About 58 seconds	Less than 9 seconds

experiments in a set of consecutive days. However, the manual classification and bin picking process is a tedious task and the performance depends heavily on being refreshed and rested. Otherwise, the 100% accuracy, precision and recall may not be achieved in the reported time.

For the automatic classification in a real environment, data creation and training phases must be performed every couple of months, when new unknown fasteners are detected. Therefore, in a real environment, the average data creation and training time might be more than the one we presented in this experiment. Table 8.6 summarizes the results of the manual and automatic classification and bin picking.

The choice of webcams as the *PPIC*'s cameras affected the accuracy of the classification methods. However, more advanced industrial cameras can result in sharper images and better results (see the results from Chapters 4, 5, 6, 7). In conclusion, the results shown Table 8.6 indicate a practical usage of the system in real industrial environment, which validates the hypotheses H_2 and H_3 in Section 1.1.

In *SUMA*, we obtained 81% accuracy of the detected damages and dirt and over 50% intersection of units for the detected bounding boxes. Most of the error in the results (false-negatives and false-positives) belongs to the dents. Detection of the dents with *PPIC* and only using 2D cameras can be challenging. The dents can result in light reflection, even if we set the lightning with an angle towards the object. Even supposing the dent does not reflect the light, detecting it in a 2D without any additional setup is not accurate. To address dents issue, additional components, such as a chessboard template can be used. Pointing out a chessboard template with an angle towards the surface, the image of symmetric cells are distorted and indicate a possible dent in the image. However, this method requires an additional segmentation model for the dents and cannot determine the depth of dents. A better solution may be using 3D images together with the CAD model of the object that its surface is damaged.

As a result, the automatic segmentation method required about 3 seconds for an image on *SUMAComputer*. Therefore, the automatic approach is faster than the manual input process obtained in benchmark (21.3 seconds).

SUMA does not necessarily require an overhauling process, i.e. the detection of damages/dirts must be performed on the component surfaces without removing and disassembly of components. The results obtained in section 8.2.3 do not validate the H_3 , since the accuracy is only 81%. However, the trained model accuracy has a correlation with the quality of data and defusing the noise, in particular the light reflection on the surface, which can be addressed by removing and disassembling components, cleaning the components prior to inspection, and using *PPIC-D* to detect the damages. However, the results in this section validates H_2 by saving the time.

8.5 Threats To Validity

Like any other research, there are threats to validity of the results. The study was conducted in a controlled environment. It is common in industry to have controlled environments to perform tasks sensitive to environmental factors such as temperature, brightness and humidity, which is also used in the current manual inspection workflow. The automation following recent trends like Industry 4.0 uses parts of current system components [Vogel-Heuser et al., 2017]. However, there are some challenges to build automation software, such as *changeability during run-time* and *specific platforms and their constraints* [Vogel-Heuser et al., 2014]. In the following, we describe the main limitations of the study.

Small Number of Systems for Validation

We used only two systems, namely *SPARCS* and *SUMA*, for validation of the hypotheses. The main reason for this small number is that it takes a considerable amount of time to build such systems (building *SPARCS* took 3 years and building *SUMA* took 1 year). Nevertheless, for more robust results, the number of systems for validation the studies must be increased.

Number of Experiments for Benchmark

To obtain a benchmark as a baseline for comparison, we performed three experiments for *SPARCS* and 10 experiments for *SUMA*. These were limited amount of experiments performed by limited number of people at specific sites. The number of experiments must be increased and experimenter bias must be reduced by conducting the experiments using different people, preferably selected by a stratified sampling method [Trost, 1986].

Type and Number of Classes in Datasets

We used 30 classes of fasteners in *SPARCS*. However, some components may have more than 50 different classes. To ensure consistent performance more data with more classes must be recorded. To use synthetic data for classification, we must rely on having access to the 3D model of the fasteners in order to generate synthetic images. These 3D models, however may not be easily accessible. In damage identification for *SUMA*, some of the damages in the dataset are manually created and might not be realistic differing from real damaged. To address this issue real damages from target location must be used.

Size of Test Datasets Used for Evaluation

We used 20 to 30 images in test datasets for evaluation of the trained models in this dissertation. The size of test dataset may not be sufficient to conclude that the methods are scalable and generalizable. For more reliable results, more data and more images for testing are required.

Number of Views

We used two views in *SPARCS*. The number of views of parts is a limitation, especially during damage identification. Additional views and mechanical setup must be used to address this issue and avoid any occlusion of the region of interest,

Risk of Overfitting

While *PPIC* ensures that we reduce noises, such as reflections and shadows, it also creates a very homogeneous environment. the data augmentation techniques fight the risk of overfitting to the specific environment, however, it is still a small set of classes and lighting conditions for the model to learn. For example, most of the model training curves showed the validation loss is increasing while the training loss is decreasing. This is a common feature of an overfitting model.

Technology Limitations

Some of the damages, such as dents, are not detectable optically by 2D the cameras. Therefore, these structural damages need other solutions or preferably 3D and CAD based solutions [Michaels, 2008], [Fugate et al., 2001] and [Ciang et al., 2008]. Moreover, the described methods in this chapter were combined with a robotic arm to automate the tasks. Although robotic manipulation has been improved in recent years [Gu et al., 2017] [Kalashnikov et al., 2018] and powerful grabbers, such as suction cups, are used to grasp the objects, yet, there

is no single grabber that can grasp all kinds of fasteners in mechanical machines. Unavailability of powerful grabbers is another limitation for this work¹¹.

¹¹In a direct discussion with professor Matthew T. Mason, he mentioned that although there are suction cups which can grasp very small objects such as needles, there is not yet a single grabber which can pick all sorts of small and big fasteners.

Chapter 9

Contributions and Future Work

In this dissertation, we investigated inspection processes and applied deep learning and computer vision techniques to present automatic processes for sorting, categorizing, and bin picking of the fasteners, and identify damages in both fasteners and components. We described a polarized backlighting supported platform with vibration conveyor belt with single or multiple cameras to automatically detect damages of fasteners in overhaul processes and recorded datasets for fastener damage detection and presented computer vision techniques to detect the damages on the component surfaces using their image. Moreover, we trained different convolutional neural networks to sort and classify the fasteners. In addition, we have examined the bin picking task in overhaul processes to automatically place small parts into containers for reuse. The proposed deep learning based approaches are now used by Schindler Handhabetechnik to build a demonstrator, which will be deployed at MTU Aero Engines.

This chapter concludes the dissertation. Section 9.1 summarizes the main contributions and Section 9.2 outlines the future work direction of this work.

9.1 Contributions

This dissertation contributes to the application of deep learning for inspection in industrial overhaul processes to decrease the error, and save cost and time. In particular, the following list the major contributions of this dissertation.

A Research Platform for Parts Image Capturing

We presented *PPIC*, a research platform to capture images of parts in inspection processes. *PPIC* supports single-view and multi-view applications and can be configured based on the requirements of the different tasks. *PPIC* can be configured for four inspection tasks in this study, namely, sorting, single-view and

multi-view categorization, fasteners damage identification, and surface damage identification.

Fasteners Damage Identification

We implemented a benchmark of supervised and unsupervised deep learning methods used in the damage identification literature, and evaluate them on two datasets to identify the damages of the fasteners and surfaces. The deep learning models have been selected from the best practices in research community and they are representative of different approaches to building deep learning models. The best supervised learning method acquired 99% accuracy on the test dataset. However, the major contribution in this dissertation is the use of the unsupervised learning to learn the notion of normality for fasteners. We obtained 84% accuracy for fasteners and 83% for the surfaces (both on the test datasets). To the best of our knowledge, no one else has achieved better results for industrial fasteners using an unsupervised method.

Sorting, Categorization and Bin picking for Fasteners

For the application domain of fasteners we have developed sorting, categorization and bin picking methods. The sorting was accomplished by using siamese networks to detect the similarity of the fasteners. Using this similarity the fasteners are grouped prior to categorization and bin picking. The automatic fasteners similarity detection methods obtained over 99% accuracy for 20 different fasteners. Using the results of the trained siamese model, we are able to sort the all fasteners (even the ones which were not seen by the model during training). Moreover, different convolutional neural networks were trained to categorize the fasteners. We achieved 99.4% accuracy with the single-view classification for 34 bolts and washers, and 100% accuracy using the multi-view classification for 8 nuts.

In addition, we developed a processing pipeline that overhaul plants can use for an automatic bin picking approach after classification for placing the fasteners in a compartment. The results of a preliminary evaluation were presented, in which the approach found 70% of pickable objects in an image, using only 10 annotated images for training.

Finally, we investigated the use of synthetic data in combination with real images and showed its advantage to obtain comparable results with approaches that use only real image.

Datasets for Benchmarking

We have recorded eight datasets for different inspection tasks in overhaul processes and made them available to the research community for benchmarking and further use and analysis [Taheritanjani,]. These datasets can be used industry-wide for automation of industrial overhauling workflows. The datasets include:

- 2019 images from 12 different bolts for damage identification of fasteners, with their fine-grained annotations.
- 102 images from two type of materials for damage identification on surfaces, with their fine-grained annotations and 355 images of the intact surfaces of same two materials for novelty detection.
- 1000 images of 20 bolts for similarity detection of fasteners.
- 1000 images from 20 bolts and 140 images from 14 washers for fine-grained categorization of fasteners using a single-view approach.
- 800 multi-view images from 8 different nuts, for fine-grained categorization using multi-view approaches.
- 1500 images of six screws and over 6000 synthetic images obtained from their 3D model for fine-grained categorization using synthetic data.
- 36 images from three different type of cranks, for automatic bin picking of fasteners.
- 5650 images from two views, for automatic classification of fasteners (used in *SPARCS*).

See appendix C for the access URL, descriptions, and a comparison with other available datasets.

9.2 Future Work

This dissertation has shown the possibility and the applicability of using deep learning based approaches for inspection tasks in overhaul processes. However, the ability to flexibly adapt to changing requirements is one design principles of modern industrial applications [Vogel-Heuser and Hess, 2016], i.e. industry 4.0. Future work can investigate the constraints of a controlled environment used in this work and the possibility of widening them.

Another possible improvement to the work presented in this dissertation is the application of deep learning using 3D images and unsupervised techniques, which are active research areas and we could greatly benefit from their recent advances. In addition, the platform described in Chapter 3 and applications described in Chapters 4, 5, 6 and 7 can be made available for other researchers using platform as a service (PaaS) and software as a service (SaaS). The following describes possible improvements:

- **Damage identification:** To improve the results of this section, more collected data is required. With more data and also more different damages the generalizability of the models will improve. Another improvement would be the data collection in a more industrial controlled environment reducing noise and therefore making the supervised as well as the unsupervised classification more robust. With more environmental constraints (for example illumination or rotation) the model performance can possibly be improved. Another point is that the parts must be automatically rotated to detect damages on all sides of the part. Furthermore the current algorithms have to be scaled to a larger number of different fasteners and integrated in a overhauling control system. From a research and machine learning perspective, 3D methods for object detection and 3D autoencoders can be used to detect damages which might also be effective depending on the 3D scan quality. This also requires a 3D dataset which could be partly obtained from reference 3D models of the small parts. Respective to anomaly detection Generative Adversarial Networks (GANs) like AnoGAN [Schlegl et al., 2017] can be applied to the dataset. In case of having many unlabeled training data, we can try to use them to train an unsupervised model, such as an autoencoder or a generative adversarial network, reuse the lower layers of the autoencoder or the lower layers of the GAN's discriminator, add the output layer for your task on top, and fine-tune the final network using supervised learning (i.e., with the labeled training examples) [Géron, 2019]. Although GANs are computationally challenging for high resolutions, some recent studies obtained state of the art results on high resolution images [Karras et al., 2019]. The approach of damage and anomaly detection using deep learning can be transferred to other visual inspection tasks as well as other domains like medicine. With the proposed automation not only costs can be reduced but also the performance of damage as well as anomaly detection tasks can be improved.
- **Sorting:** To fully utilize the overhauling inspection tasks with the benefits of automatic sorting, we must change the image capturing platform to support having all the fasteners in one camera view. Using one high resolution image that shows all the fasteners from one component, the automatic

sorting can be finished in one shot. However, it is challenging to build a platform that can separate all the fasteners and place them on the surface under the camera. *PPIC* that was introduced in chapter 3 must be extended to use a wider vibration plate and a wider conveyor belt. Moreover, the camera must be mounted in a higher distance to be able to capture all the instances. With such a platform, we can focus the research to use the sorting process instead of the categorization.

- **Categorizing:** We can introduce more cameras to be able to capture every feature of any fastener, regardless of orientation and position. Since we must ensure to maintain the scale feature by setting a fixed distance between the camera and the fastener, by introducing physical markers next to the fastener (for example, on the side of conveyor belt), we can automatically calibrate the cameras and scale the image to a fixed size [Koterba et al., 2005]. Moreover, to address the low amount of discriminative features in one specific view described in Sections 6.2.3 and 8.1.3, we can use the images of different similar looking fasteners in one class, detect them using a single-view classification model, and use the second-view to differentiate between the different fasteners. In this way, the similarity of second view will not affect the whole classification. Regarding synthetic data, the results indicate that, when adding new classes, the output classification accuracy is affected by the aesthetic similarity of fasteners in the dataset. A logical next step is to evaluate the CNNs after increasing the number of output classes. Moreover, a possible improvement is to evaluate the usage of deeper CNN models such as the ones presented by He et al. and Szegedy et al. [He et al., 2016] [Szegedy et al., 2016]. In addition, the total number of synthetic images in the training set has not been varied through the experiments. A possible improvement in accuracy could be achieved by training the CNN models on a bigger training set containing a larger amount of synthetic images and from multiple views as discussed earlier in Chapter 6. Finally, application of classification of 3D images of fasteners and comparing the results with the proposed multi-view techniques can be also interesting.
- **Bin Picking:** We only used three crank types for bin picking of fasteners on non-uniform backgrounds. However, the same approach can be employed for other small parts by identifying the pickability conditions. For example, we decided that cranks must be placed flat relative to camera and their inner holes must be non-occluded to consider them pickable. For bolts, they must be placed flat relative to camera, and in addition, there must be a safe distance between their contour and surrounding objects that finger-based grippers can pick them. For nuts, they must be placed head-up or head-down

without any occlusion. In addition, pickability conditions can also be addressed using keypoint detection of the objects, which has been introduced in some instance segmentation frameworks [He et al., 2017]. It is also possible to detect the keypoints for the small parts as part of the training. Study of pickability conditions for different small parts and performing keypoint detection on the small parts left for future work. In addition, the same techniques used from Wu et al. can also be employed for bin picking in overhaul processes, using the point clouds obtained from 3D cameras [Wu et al., 2019].

With the combination of robotic arms, the approaches presented in this dissertation can be extended to automate inspection process of different industrial small parts and components. These approaches could be applied to other tasks in overhauling processes, such as reassembly of the components. With training an action recognition model and using it along with the categorization models, the technicians can benefit from automatic suggestions and *next-step information* to perform the tasks.

Another interesting research direction is the combination of inspection process for electrical and mechanical components. This includes anomaly detection techniques to process the electrical devices. Using this combination, the whole machine can be inspected without separating the parts and components.

A long term evaluation in an industrial setting that shows the benefits of automatic inspection process would also be an interesting research direction. For this purpose, an integrated tool could be implemented. When combined with *PPIC* platform, interesting insights regarding feedback and actual development changes could be obtained. These changes could then be analyzed and correlated to inspection accuracy. Based on the feedback obtained from the discussion with industry partners at MTU Hanover¹, we believe that an integrated tool with the techniques presented in this dissertation could also be of interest for the industry.

¹<https://www.mtu.de/maintenance/>

Appendix A

Acronyms

AE Autoencoder

AP Average Precision

AnoGAN Unsupervised Anomaly Detection with Generative Adversarial Networks

COCO Common Objects in Context

CNN Convolutional Neural Network

FCN Fully Convolutional Network

FGVC Fine-grained Visual Categorization

FN False Negative

FP False Positive

GAN Generative Adversarial Network

GPU Graphical Processing Unit

IF Isolation Forest

IoU Intersection over Union

KS Knowledge Source

LOF Local Outlier Factor

mAP mean Average Precision

MLP Multilayer Perceptron

MRO	Maintenance, Repair and Overhaul
MSE	Mean Squared Error
MVCNN	Multi-view Convolutional Neural Network
MIRO	Multi-view Images of Rotated Objects
NN	Neural Network
OC	One-Class
PCA	Principal Component Analysis
PPIC	Platform for Parts Image Capturing
RCM	Reliability Centered Maintenance
R-CNN	Regions with CNN features
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SSD	Single Shot MultiBox Detector
SPARCS	Small Parts Classification System
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TPA	Total Productive Maintenance
VIA	VGG Image Annotator
SUMA	Surface Maintenance

Appendix B

Copyrights

As addressed in Chapter 1, parts of this dissertation have been previously published through different publishers. For all used material, the permission for reuse in this dissertation is presented. Figures B.1 and B.2 show the retrieved permission grants from the IEEE Xplore Digital Library.



RightsLink®

Home
Help
Email Support
Sign in
Create Account



Requesting permission to reuse content from an IEEE publication

Fine-Grained Visual Categorization of Fasteners in Overhaul Processes

Conference Proceedings:
2019 5th International Conference on Control, Automation and Robotics (ICCAR)

Author: Sajjad Taheritanjani

Publisher: IEEE

Date: April 2019

Copyright © 2019, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:


- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK
CLOSE

© 2020 Copyright - All Rights Reserved | [Copyright Clearance Center, Inc.](#) | [Privacy statement](#) | [Terms and Conditions](#)
Comments? We would like to hear from you. E-mail us at customercare@copyright.com


Figure B.1: Permission grant for [Taheritanjani et al., 2019a]



RightsLink®

Home
Help
Email Support
Sign in
Create Account

Automatic Damage Detection of Fasteners in Overhaul Processes



Requesting
permission
to reuse
content from
an IEEE
publication

Conference Proceedings:
2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)

Author: Sajjad Taheritanjani

Publisher: IEEE

Date: Aug. 2019

Copyright © 2019, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK
CLOSE

© 2020 Copyright - All Rights Reserved | [Copyright Clearance Center, Inc.](#) | [Privacy statement](#) | [Terms and Conditions](#)
Comments? We would like to hear from you. E-mail us at customercare@copyright.com

Figure B.2: Permission grant for [Taheritanjani et al., 2019b]

Content of the SCITEPRESS published work ([Taheritanjani et al., 2020]) is used in this dissertation based on the ACM Author Rights¹ as well as the copyright shown in Figure B.3.

¹<https://authors.acm.org/author-services/author-rights>

CONSENT TO PUBLISH and COPYRIGHT TRANSFER

For the mutual benefit and protection of Authors and Publishers, it is necessary that Authors provide formal written Consent to Publish and Transfer of Copyright before publication of the Book. The signed Consent ensures that the publisher has the Author's authorization to publish the Contribution.

Conference: **ICPRAM 2020 - 9th International Conference on Pattern Recognition Applications and Methods.**

Place/Date: **Valletta, Malta; 22 - 24 February, 2020.**

Book Title: **Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods.**

Edited by: **Maria De Marsico, Gabriella Sanniti di Baja and Ana Fred.**

Publisher: **SCITEPRESS.**

Paper number: **47.**

Title of the contribution: **2D Orientation and Grasp Point Computation for Bin Picking in Overhaul Processes.**

Author (name and address):

Sajjad Taheritajani.

Germany.

It is herein agreed that:

The copyright to the contribution identified above is transferred from the Author to the "Science and Technology Publications, Lda" (here forth known as SCITEPRESS). The copyright transfer covers the exclusive, sole, permanent, world-wide, transferable, sub licensable and unlimited right to reproduce, publish, transmit, archive, lease/lend, sell and distribute the contribution or parts thereof individually or together with other works in any language, revision and version (digital and hard), including reprints, translations, photographic reproductions, microform, audiograms, videograms, electronic form (offline, online), or any other reproductions of similar nature, including publication in the aforementioned book or any other book, as well as, the usage for advertising purposes. SCITEPRESS is also entitled to carry out editorial changes in the contribution with the sole purpose of enhancing the overall organization and form of the contribution. The Author retains the rights to publish the contribution in his/her own web site and thesis, in his/her employer's web site and to publish a substantially revised version (at least 30% new material) elsewhere, as long as it is clearly stated that the contribution was presented at ICPRAM 2020, a link to the event web site is made available there and also the presence of the corresponding DOI number. Prior versions of the contribution published on non-commercial pre-print servers like ArXiv/CoRR and HAL can remain on these servers and/or can be updated with Author's accepted version. The final published version (in pdf) cannot be used for this purpose. The Creative Commons license CC BY-NC-ND applies to everyone that wishes to use the published version.

The Author warrants that his/her contribution is original, except for such excerpts from copyrighted works as may be included with the permission of the copyright holder and author thereof, that it contains no libelous statements, and does not infringe on any copyright, trademark, patent, statutory right, or propriety right of others; and that Author will indemnify SCITEPRESS against any costs, expenses or damages for which SCITEPRESS may become liable as a result of any breach of this warranty. The Author signs for and accepts responsibility for releasing this material on behalf of any and all co-authors.

This agreement shall be governed by, and shall be construed in accordance with, the laws of Portugal. The courts of Portugal shall have the exclusive jurisdiction.

In return for these rights:

The publisher agrees to have the identified contribution published, at its own cost and expense, in the event proceedings.

The undersigned hereby gives permission to SCITEPRESS to have the above contribution published.

Date: **11 December, 2019**

Figure B.3: Permission grant for [Taheritajani et al., 2020]

Appendix C

Datasets

Access URL: <https://mediatum.ub.tum.de/1537830>

There are 8 compressed files, each represent dataset for one task:

- **Fastener Damage Identification:** consists of images of damaged and intact fasteners, with fine-grained annotations of damages to be used with supervised segmentation methods such as Mask R-CNN and also patches of the images to be used in patch autoencoders.
- **Surface Damage Identification:** consists of images of damaged and intact surfaces, with fine-grained annotations of damages to be used with supervised segmentation methods such as Mask R-CNN.
- **Sorting:** consists of images from 23 classes of bolts and washers to be used for similarity detection algorithms such as siamese networks and also for categorization and classification of the fasteners.
- **Single-View Classification:** consists of images from 20 classes of bolts and 14 classes washers to be used for single-view categorization and classification of the fasteners and also for similarity detection algorithms such as siamese networks.
- **Multi-View Classification:** consists of images from two views of 8 classes of nuts to be used for multi-view categorization and classification of the fasteners.
- **Classification using Synthetic Images:** consists of images and 3D CAD models of 6 classes of screws to be used for single-view categorization and classification of the fasteners using synthetic data.

- Bin Picking on Non-uniform Background : consists of images and fine grained annotations of 5 classes of cranks to be used for instance segmentation of the fasteners together with grasp point and orientation calculation for bin picking.
- SPARCS: consists of images from 30 classes of bolts and 4 classes washers, captured with low resolution webcam, to be used for single-view categorization and classification of the fasteners and also for similarity detection algorithms such as siamese networks.

All the fine-grained annotations for the segmentation are stored in *JSON* files which must be loaded prior to training in Mask R-CNN.

To the best of our knowledge, there are no other publicly available datasets for fasteners damage identification, sorting and classification. However, there are multiple available datasets for surface damage detection and bin picking. For a better overview, we compare our surface damage detection and bin picking datasets to other available datasets with regards to a rating schema shown in Table C.1.

Table C.1: Rating schema used to compare the datasets.

Feature	Explanation
F ₁ : Object Type	+ : industrial surfaces and fasteners ± : other industrial small parts (high intra-class and low inter-class variance) - : not industrial small parts
F ₂ : Number of Classes	The higher the better
F ₃ : Number of Images (per Class)	Usually more images are preferred
F ₄ : Resolution of Images	In higher resolution more fine-grained segmentation is possible
F ₅ : Annotations and Labels	+ : available ± : partially available (less than 50% of the data) - : not available

Table C.2 shows the comparison between the surface damage detection dataset used in this dissertation and other available datasets with regards to the rating schema.

Table C.2: Comparison between the surface damage detection dataset used in this dissertation and other available surface damage detection datasets with regards to the rating scheme described in Table C.1.

Dataset	F ₁	F ₂	F ₃	F ₄	F ₅
[Amhaz et al., 2015]	±	5	23 to 89 (damaged)	varied ranges from 220 to 1000 pixels (height and width)	±
<i>Road Anomaly Detection</i> ¹	±	4	489 to 640		+
[Tabernik et al., 2019]	±	1	52 (damaged) 347 (Intact)	500 × 1255	+
<i>Severstal: Steel Defect Detection</i> ²	+	1	12568	1600 × 256	+
[Song and Yan, 2013]	+	6	1800 (damaged)	200 × 200	+
Surface damage detection dataset used in this dissertation	+	2	60 (damaged) 170 (intact)	5184 × 3456	+

Table C.3 shows the comparison between the bin picking datasets used in this dissertation and other available datasets with regards to the rating schema.

Table C.3: Comparison between the bin picking datasets used in this dissertation and other available bin picking datasets with regards to the rating scheme described in Table C.1.

Dataset	F ₁	F ₂	F ₃	F ₄	F ₅
[Drost et al., 2017]	±	28	125	3264 × 2448	±
[Brégier et al., 2017]	±	8	122 (each image contains one to 24 fine-grained annotations)	1178 × 1018	+ (instance segmentation)
[Kleeberger et al., 2019]	±	10	varied ranges from 290 to 39260	1280 × 1024	+ (instance segmentation)
Bin picking datasets used in this dissertation	+	5	12 (each image contains four to 63 fine-grained annotations)	1280 × 1024	+ (instance segmentation)

¹<http://radprojectbismil.blogspot.com>

²<https://www.kaggle.com/c/severstal-steel-defect-detection/>

List of Figures

1.1	Overview of maintenance types.	2
1.2	Evolution of maintenance through the years.	3
1.3	Activities in overhauling of industrial machines for both corrective and planned preventive maintenance (UML activity diagram).	4
1.4	Activities in overhauling of industrial machines for condition-based preventive maintenance (UML activity diagram).	4
1.5	Composition of machine component and taxonomy of fasteners.	5
1.6	Simplified activities with focus on Inspection in overhaul processes (UML activity diagram).	6
1.7	Examples of intact and damaged bolts from a side view (a) and top view (b). The annotated damages show either scratches or dirt which could not be cleaned by washing [Taheritanjani et al., 2019b].	7
1.8	Examples of industrial parts on a workstation that are grouped based on their similarity.	8
1.9	Fasteners that were taken from part of engine and must be classified during inspection in overhaul process.	9
1.10	The structure of a sample bolt, washer and nut, displaying the length, width, shaft, pitch and diameters ([Taheritanjani et al., 2019a] and [Birkeland, 2018]).	10
1.11	Examples of compartments which contain the small parts after packaging.	11
2.1	Original Image (a). The result of contour detection (b). Minimum enclosing rectangle around the contour (c).	18
2.2	Original Image (a) and its first image moment as the centroid in pink color (b). The mass on the bolt head is more than its shaft. Therefore the centroid is not exactly on the center and is towards the head.	19
2.3	Applying PCA to an input image and finding the axes. To bigger axis represents the orientation of the bolt.	20
2.4	Artificial neuron.	21

2.5	An example of a multilayer perceptron.	22
2.6	Sigmoid activation function.	23
2.7	Tanh activation function.	23
2.8	ReLU activation function.	23
2.9	Leaky ReLU activation function.	23
2.10	Applying convolutional filters to extract features.	25
2.11	An example of feature map visualization [Zeiler and Fergus, 2014].	26
2.12	Applying pooling filters to subsample feature maps.	27
2.13	Nut augmentations	27
2.14	Visualization of a gradient descent example.	29
2.15	Overfitting	31
2.16	Inception modules for multilevel feature extraction [Szegedy et al., 2016]	33
2.17	Residual block with identity function for free gradient flow [He et al., 2016]	34
2.18	Multi view convolutional neural network	34
2.19	Example of artificial camera configurations with 12 (a), 20 (b) and 60 (c) angles [Kanezaki et al., 2018].	35
2.20	Bookshelf and dresser from different angles	36
2.21	Multi-view CNN architecture [Su et al., 2015].	37
2.22	RotationNet/Multi-view CNN results [Kanezaki et al., 2018]	38
2.23	Siamese network architecture.	39
2.24	Object detection output [Girshick et al., 2014]	39
2.25	Semantic segmentation using a FCN [Long et al., 2015]	40
2.26	Instance segmentation using Mask R-CNN [He et al., 2017]	41
2.27	One-Class SVM	43
2.28	Isolation Forest	43
2.29	Local Outlier Factor determining density with k=3 neighbors	44
3.1	An example of similar fasteners. A is identical to C except that its non-threaded shaft is one millimeter shorter. B has a two millimeter longer non-threaded shaft and a one millimeter smaller diameter than C [Taheritanjani et al., 2019a].	47
3.2	An example of two bolts that each one is a scale version of the other one. Scaling one's image is an example of the perspective issue and can result in a misclassification [Taheritanjani et al., 2019a].	48
3.3	The sketch of <i>PPIC</i> with a vibration conveyor belt. The backlighting comes from below the conveyor belt. The light green section can move the parts back to the beginning of the conveyor belt and the vibration section.	50

3.4	The sketch of <i>PPIC-A</i> with a light diffusing conveyor belt. The backlighting comes from below the conveyor belt and a single camera with 90° captures the images.	51
3.5	Image of fasteners obtained on the conveyor belt using normal ambient lighting (a) and using top camera view of <i>PPIC-A</i> with reduced shadows and reflection (b).	51
3.6	The sketch of <i>PPIC-B</i> with a single camera, light diffusing conveyor belt, and backlighting. There are illumination on top of platform and light diffusing textile to polarize the light.	52
3.7	Image of a bolt obtained using normal ambient lighting (a) and using camera view of <i>PPIC-B</i> (b).	52
3.8	The sketch of <i>PPIC-C</i> with two cameras, illumination on top, and polarized backlighting supported conveyor belt.	53
3.9	Image of a nut obtained using both camera views of <i>PPIC-C</i>	54
3.10	The sketch of <i>PPIC-D</i> with one camera on top, and one illumination source with 45° angle towards the surface.	54
3.11	Image of damages on a surface a using normal ambient lighting (a) and using <i>PPIC-D</i> with reduced reflection (b).	55
4.1	The preprocessing workflow (UML activity diagram).	60
4.2	The overview of the algorithms which are used in this study [Taheritanjani et al., 2019b].	61
4.3	An example of the heatmap visualization for a damaged fastener. Low activation is shown with cold colors (namely blue). The colder area indicates less anomalies in that part of the image [Taheritanjani et al., 2019b].	63
4.4	An example of Mask R-CNN result for the damages including their masks and bounding boxes [Taheritanjani et al., 2019b].	63
4.5	An example of the autoencoder input (first row), its reconstruction (second row), reconstruction error (third row), and square of reconstruction error (last row) for an intact and a damaged fastener [Taheritanjani et al., 2019b].	65
4.6	An example of the original image input (first row), thresholding result (second row), segmentation result using Mask R-CNN (third row), and its autoencoder reconstruction (last row) for galvanized surfaces.	72
4.7	An example of the original image input (first row), thresholding result (second row), segmentation result using Mask R-CNN (third row), and its autoencoder reconstruction (last row) for aluminum surfaces.	73

5.1	An example of the original input image that was used to test the accuracy of model on new classes fasteners.	81
5.2	The result of computed similarity of the fasteners on the input image (Figure 5.1).	83
6.1	The camera view is divided into five regions shown in blue and aqua. The red dots indicate the differences between the bolt head when the bolt is moved vertically along its shaft. The yellow dots indicate the differences in light reflection on the bolt shaft when it is moved horizontally. The more we move the bolt to the left side of the view, the less reflection is captured on its left side, and vice versa [Taheritanjani et al., 2019a].	87
6.2	Twenty bolts and fourteen washers which were used for the classification.	88
6.3	Overview of steps executed before saving the data on the disk (UML activity diagram).	89
6.4	Overview of preprocessing steps executed before feeding the data to the train process (UML activity diagram).	89
6.5	An example fastener (on the left) and its 3D model created by 3D scanner (on the right). The 3D model is rendered without texture for clarity.	101
6.6	Examples of images for each class that uses in the experiments. For each pair, the image on the left is a sample real image, while the image on the right is a sample synthetic image.	102
6.7	An example synthetic scene where a screw lies on a horizontal plane. The isometric view of the scene (a). The side view (b). . . .	103
7.1	Examples of the images captured for training a classifier, using <i>PPIC-A</i>	109
7.2	Overview of preprocessing steps executed prior to train an occluded/not-occluded classifier (UML activity diagram).	109
7.3	Example of fasteners that are placed next to each other (a) and their contour obtained from adaptive thresholding algorithm (b). .	110
7.4	Examples of orientation (green axis) and grasp point (pink circle) calculations for two bolts.	111
7.5	The datasets contain three different cranks (a). Example image (b) [Taheritanjani et al., 2020].	114

7.6	Example annotated image in the dataset. While only the outer contour of unpickable cranks are annotated, we also annotated two inner holes of pickable instances, and their outer contour. The pickable and unpickable cranks were labeled as different classes [Taheritanjani et al., 2020].	114
7.7	Overview of the required methods for fasteners bin picking in overhaul processes. The computation steps to calculate the orientation of the pickable cranks, using Mask R-CNN or PCA, and the grasp point of the pickable cranks, using Mask R-CNN or Image Moment. The mask filtering algorithm (marked in yellow) must be calculated only once, for both the orientation and the grasp point computations.	115
7.8	Example original input image (a), the visualization of the pickable cranks after filtering the masks, using only Mask R-CNN (b), and the visualization of all of the found pickable cranks using Mask R-CNN (c). Using the results from (c), we apply PCA and Image Moment to find the orientation (green and blue pivots) and the grasp point (yellow circle) for the lever on the lower right corner of the input image (d) [Taheritanjani et al., 2020].	116
8.1	<i>SPARCS</i> use cases (UML use case diagram).	123
8.2	Activities in <i>CreateDatasetForComponent</i> scenario of <i>SPARCS</i> (UML activity diagram).	125
8.3	Simplified activities in <i>SortBolts</i> scenario of <i>SPARCS</i> (UML activity diagram).	126
8.4	The realization of <i>PPIC-A</i> with a rotating glass table covered with opal foil and supported with backlighting, used in the first iteration of <i>SPARCS</i>	127
8.5	The realization of <i>PPIC-A</i> with a vibration plate and LED backlighting, used in the second iteration of <i>SPARCS</i>	128
8.6	The realization of <i>PPIC</i> with a vibration conveyor belt with light diffusing textile and backlighting, used in current solution of <i>SPARCS</i>	129
8.7	<i>SPARCS</i> hardware software mapping (UML deployment diagram).	130
8.8	Overview of the 30 bolts used in <i>SPARCS</i> dataset. Some bolts may look distorted due the image projection prior to dataset capturing.	131
8.9	Images of the heads of three bolts (top) and their respective side view (bottom). The similarity of the heads reduces the accuracy of multi-view model after aggregating the results into an overall classification.	133
8.10	<i>SUMA</i> use cases (UML use case diagram).	135

8.11	The hardware platform used in <i>SUMA</i> . The robotic arm is equipped with a 3D camera and sees the surface in front of it.	137
8.12	<i>SUMA</i> hardware software mapping (UML deployment diagram).	138
8.13	Example of segmentation results from the trained Mask R-CNN model.	139
B.1	Permission grant for [Taheritanjani et al., 2019a]	156
B.2	Permission grant for [Taheritanjani et al., 2019b]	157
B.3	Permission grant for [Taheritanjani et al., 2020]	159

List of Tables

1.1	Rating schema used to compare the results and state of the art in this dissertation.	14
2.1	Model results on the ImageNet 2012 classification challenge . . .	32
4.1	Fasteners used in the dataset.	59
4.2	Knowledge sources that are used for damage detection with their input and output sources. Feature Vector: lower dimensional representation, Area Damaged: percentage of the fastener which is damaged, Numeric Anomaly: numeric value representing how anomalous a fastener is according to a damage detection algorithm, Model Output: output of a model determining whether the fastener is damaged or intact, Result: output of the blackboard representing the final decision.	62
4.3	Supervised classification performance results on test dataset . . .	64
4.4	Unsupervised classification performance results on test dataset. . .	66
4.5	Comparison between the fastener damage identification approaches used in this dissertation and the state of the art results in damage identification with regards to the rating scheme.	67
4.6	Surfaces used in the dataset (102 images with 15 damages per image on average and their fine-grained annotations and 355 images of the intact surfaces for novelty detection).	68
4.7	Parameters and settings used to train Mask R-CNN model, pre-trained with COCO dataset.	70
4.8	Parameters and settings used to train autoencoder for anomaly detection.	70
4.9	The performance results on test dataset.	71
5.1	Fasteners used in the dataset (scaled to 0.5 of the original size). . .	78
5.2	Overview of parameters and settings used to train models.	80

5.3	Overview of the training results for sorting using Siamese networks, using 20 classes, 200×200 input image size, 1000 test pairs, and 40000 iterations.	80
5.4	Results of the model accuracy on new classes of fasteners, averaged in test images.	82
6.1	Overview of the training results, using InceptionV3 and Keras on top of TensorFlow, optimizer=SGD (Stochastic Gradient Descent), learning rate=0.0001 and momentum=0.9. The accuracy is reported on the test dataset which is 10% of the whole data. . .	90
6.2	Eight nuts which are captured in the dataset.	93
6.3	Overview of parameters and settings used to train models.	96
6.4	Classification accuracy on test dataset.	97
6.5	Average computation time per nut during inference in ms.	97
6.6	MultiCNN individual network's test accuracy. <i>Network1</i> uses the 90° view images and <i>Network2</i> uses the 45° view images.	97
6.7	The number of images per class for the specified data split.	103
6.8	Overview of parameters and settings used to train models.	104
6.9	Class-wise classification accuracy of a VGG16 networks.	104
6.10	Class-wise classification accuracy of a VGG19 networks.	104
6.11	Comparison between the fasteners categorization approaches used in this dissertation and the state of the art results in single-view/multi-view image classification with regards to the rating scheme.	105
7.1	Parameters and settings used to train pickable/non-pickable classifier.	110
7.2	Overview of classification, orientation and grasp point calculation results.	112
7.3	Parameters and settings used to train Mask R-CNN model.	115
7.4	Overview of segmentation results.	119
7.5	Overview of orientation and grasp point calculation results.	119
7.6	Comparison between the orientation and grasp point computation approaches for bin picking used in this dissertation and the state of the art results in the orientation and grasp point calculation for bin picking with regards to the rating scheme.	121
8.1	Overview of bolt/not-bolt model, and grasp point and orientation calculation results for 30 classes of bolts.	132
8.2	Overview of classification results for 30 classes of bolts.	132
8.3	Overview of instance segmentation results using Mask R-CNN for dirt and damage detection.	139

8.4	Three manual experiments performed to obtain benchmark as a basis for comparison in <i>SPARCS</i>	140
8.5	10 manual experiments performed to obtain benchmark as a basis for comparison in <i>SUMA</i>	141
8.6	Comparison between the manual and the automatic classification and bin picking for 341 fasteners.	142
C.1	Rating schema used to compare the datasets.	161
C.2	Comparison between the surface damage detection dataset used in this dissertation and other available surface damage detection datasets with regards to the rating scheme described in Table C.1.	162
C.3	Comparison between the bin picking datasets used in this dissertation and other available bin picking datasets with regards to the rating scheme described in Table C.1.	162

Bibliography

- [Abdelraouf, 2018] Abdelraouf, A. (2018). Using synthetic data for classification of small parts. Master's thesis, Technische Universität München, Germany.
- [Ahuja and Khamba, 2008] Ahuja, I. P. S. and Khamba, J. S. (2008). Total productive maintenance: literature review and directions. *International Journal of Quality & Reliability Management*, 25(7):709–756.
- [Akata et al., 2015] Akata, Z., Reed, S., Walter, D., Lee, H., and Schiele, B. (2015). Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936.
- [Amari et al., 1996] Amari, S.-i., Cichocki, A., and Yang, H. H. (1996). A new learning algorithm for blind signal separation. In *Advances in neural information processing systems*, pages 757–763.
- [Amhaz et al., 2015] Amhaz, R., Chambon, S., Idier, J., and Baltazart, V. (2015). Automatic crack detection on 2d pavement images: An algorithm based on minimal path selection, accepted to iee trans. *Intell. Transp. Syst.*
- [Aytekin et al., 2015] Aytekin, Ç., Rezaeitabar, Y., Dogru, S., and Ulusoy, I. (2015). Railway fastener inspection by real-time machine vision. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(7):1101–1107.
- [Azoff, 1974] Azoff, E. A. (1974). Borel measurability in linear algebra. *Proceedings of the American Mathematical Society*, 42(2):346–350.
- [Baur et al., 2018] Baur, C., Wiestler, B., Albarqouni, S., and Navab, N. (2018). Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. *arXiv preprint arXiv:1804.04488*.
- [Bellman, 2013] Bellman, R. (2013). *Dynamic programming*. Courier Corporation.
- [Bertinetto et al., 2016] Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. (2016). Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer.

- [Birkeland, 2018] Birkeland, R. S. (2018). A multi-view cnn approach to classify bolts and nuts in overhauling processes. Master’s thesis, Technische Universität München, Germany.
- [Biver et al., 2012] Biver, S., Fuqua, P., and Hunter, F. (2012). *Light science and magic: An introduction to photographic lighting*. Routledge.
- [Brachmann et al., 2016] Brachmann, E., Michel, F., Krull, A., Ying Yang, M., Gumhold, S., et al. (2016). Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3364–3372.
- [Brégier et al., 2017] Brégier, R., Devernay, F., Leyrit, L., and Crowley, J. L. (2017). Symmetry aware evaluation of 3d object detection and pose estimation in scenes of many parts in bulk. In *The IEEE International Conference on Computer Vision (ICCV)*.
- [Breunig et al., 2000] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM.
- [Bruegge and Dutoit, 1999] Bruegge, B. and Dutoit, A. A. (1999). *Object-oriented software engineering; conquering complex and changing systems*. Prentice Hall PTR.
- [Cha et al., 2017] Cha, Y.-J., Choi, W., and Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32(5):361–378.
- [Cha et al., 2018] Cha, Y.-J., Choi, W., Suh, G., Mahmoudkhani, S., and Büyüköztürk, O. (2018). Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*.
- [Chai, 2015] Chai, Y. (2015). *Advances in fine-grained visual categorization*. PhD thesis, Oxford University, UK.
- [Chalapathy et al., 2018] Chalapathy, R., Menon, A. K., and Chawla, S. (2018). Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*.
- [Chan et al., 2015] Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., and Ma, Y. (2015). Pcanet: A simple deep learning baseline for image classification. *IEEE transactions on image processing*, 24(12):5017–5032.
- [Chang et al., 2015] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. (2015). Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.
- [Chen and Deng,] Chen, L. and Deng, J. Research on surface defects detection of stainless steel spoon based on machine vision. In *2018 Chinese Automation Congress (CAC)*, pages 1096–1101. IEEE.

- [Chen et al., 2017] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.
- [Chen et al., 2018] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818.
- [Chen et al., 2019] Chen, N., Sun, J., Wang, X., Huang, Y., Li, Y., and Guo, C. (2019). Research on surface defect detection and grinding path planning of steel plate based on machine vision. In *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1748–1753. IEEE.
- [Ciang et al., 2008] Ciang, C. C., Lee, J.-R., and Bang, H.-J. (2008). Structural health monitoring for a wind turbine system: a review of damage detection methods. *Measurement Science and Technology*, 19(12):122001.
- [Cubuk et al., 2018] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.
- [Danielczuk et al., 2018] Danielczuk, M., Mahler, J., Correa, C., and Goldberg, K. (2018). Linear push policies to increase grasp access for robot bin picking. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 1249–1256. IEEE.
- [Danielczuk et al., 2019] Danielczuk, M., Matl, M., Gupta, S., Li, A., Lee, A., Mahler, J., and Goldberg, K. (2019). Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7283–7290. IEEE.
- [Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- [Dieter Schraft and Ledermann, 2003] Dieter Schraft, R. and Ledermann, T. (2003). Intelligent picking of chaotically stored objects. *Assembly Automation*, 23(1):38–42.
- [Do et al., 2018] Do, T.-T., Cai, M., Pham, T., and Reid, I. (2018). Deep-6dpose: Recovering 6d object pose from a single rgb image. *arXiv preprint arXiv:1802.10367*.
- [Drost et al., 2017] Drost, B., Ulrich, M., Bergmann, P., Hartinger, P., and Steger, C. (2017). Introducing mvtec itodd-a dataset for 3d object recognition in industry. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2200–2208.

- [Dutta and Zisserman, 2019] Dutta, A. and Zisserman, A. (2019). The via annotation software for images, audio and video. *arXiv preprint arXiv:1904.10699*, 5.
- [D’Avella et al., 2020] D’Avella, S., Tripicchio, P., and Avizzano, C. A. (2020). A study on picking objects in cluttered environments: Exploiting depth features for a custom low-cost universal jamming gripper. *Robotics and Computer-Integrated Manufacturing*, 63:101888.
- [Endrenyi et al., 2001] Endrenyi, J., Aboresheid, S., Allan, R., Anders, G., Asgarpoor, S., Billinton, R., Chowdhury, N., Dialynas, E., Fipper, M., Fletcher, R., et al. (2001). The present status of maintenance strategies and the impact of maintenance on reliability. *IEEE Transactions on power systems*, 16(4):638–646.
- [Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.
- [Feng et al., 2013] Feng, H., Jiang, Z., Xie, F., Yang, P., Shi, J., and Chen, L. (2013). Automatic fastener classification and defect detection in vision-based railway inspection systems. *IEEE transactions on instrumentation and measurement*, 63(4):877–888.
- [Ferguson et al., 2018] Ferguson, M., Ak, R., Lee, Y.-T. T., and Law, K. H. (2018). Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning. *arXiv preprint arXiv:1808.02518*.
- [Fugate et al., 2001] Fugate, M. L., Sohn, H., and Farrar, C. R. (2001). Vibration-based damage detection using statistical process control. *Mechanical Systems and Signal Processing*, 15(4):707–721.
- [Georgakis et al., 2017] Georgakis, G., Mousavian, A., Berg, A. C., and Kosecka, J. (2017). Synthesizing training data for object detection in indoor scenes. *arXiv preprint arXiv:1702.07836*.
- [Géron, 2019] Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems*. ” O’Reilly Media, Inc.”.
- [Giben et al., 2015] Giben, X., Patel, V. M., and Chellappa, R. (2015). Material classification and semantic segmentation of railway track images with deep convolutional neural networks. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 621–625. IEEE.
- [Gibert et al., 2015] Gibert, X., Patel, V. M., and Chellappa, R. (2015). Robust fastener detection for autonomous visual railway track inspection. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 694–701. IEEE.
- [Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

- [Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.
- [Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.
- [Grau, 2003] Grau, O. (2003). *Virtual Art: from illusion to immersion*. MIT press.
- [Gu et al., 2017] Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE.
- [Gupta et al., 2014] Gupta, M., Müller, J., and Sukhatme, G. S. (2014). Using manipulation primitives for object sorting in cluttered environments. *IEEE transactions on Automation Science and Engineering*, 12(2):608–614.
- [He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.
- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [Hema et al., 2007] Hema, C. R., Paulraj, M., Nagarajan, R., and Sazali, Y. (2007). Segmentation and location computation of bin objects. *International Journal of Advanced Robotic Systems*, 4(1):9.
- [Hinton, 1992] Hinton, G. E. (1992). How neural networks learn from experience. *Scientific American*, 267(3):144–151.
- [Holland, 2008] Holland, S. M. (2008). Principal components analysis (pca). *Department of Geology, University of Georgia, Athens, GA*, pages 30602–2501.
- [Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.
- [Huang et al., 2008] Huang, G. B., Mattar, M., Berg, T., and Learned-Miller, E. (2008). Labeled faces in the wild: A database for studying face recognition in unconstrained environments.
- [Jardine and Tsang, 2005] Jardine, A. K. and Tsang, A. H. (2005). *Maintenance, replacement, and reliability: theory and applications*. CRC press.
- [Jolliffe, 2011] Jolliffe, I. (2011). *Principal component analysis*. Springer.

- [Kalashnikov et al., 2018] Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. (2018). Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*.
- [Kanezaki et al., 2018] Kanezaki, A., Matsushita, Y., and Nishida, Y. (2018). Rotation-net: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [Karakasis et al., 2015] Karakasis, E. G., Amanatiadis, A., Gasteratos, A., and Chatzichristofis, S. A. (2015). Image moment invariants as local features for content based image retrieval using the bag-of-visual-words model. *Pattern Recognition Letters*, 55:22–27.
- [Karras et al., 2019] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2019). Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*.
- [Khosla et al., 2011] Khosla, A., Jayadevaprakash, N., Yao, B., and Li, F.-F. (2011). Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, volume 2.
- [Kim et al., 2012] Kim, K., Kim, J., Kang, S., Kim, J., and Lee, J. (2012). Vision-based bin picking system for industrial robotics applications. In *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 515–516. IEEE.
- [Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- [Kleeberger et al., 2019] Kleeberger, K., Landgraf, C., and Huber, M. F. (2019). Large-scale 6d object pose estimation dataset for industrial bin-picking. *arXiv preprint arXiv:1912.12125*.
- [Koch et al., 2015] Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2.
- [Koterba et al., 2005] Koterba, S. C., Baker, S., Matthews, I., Hu, C., Xiao, J., Cohn, J., and Kanade, T. (2005). Multi-view aam fitting and camera calibration. In *Proc. International Conference on Computer Vision*, volume 1, pages 511 – 518.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

- [Lampe et al., 2004] Lampe, M., Strassner, M., and Fleisch, E. (2004). A ubiquitous computing environment for aircraft maintenance. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1586–1592. ACM.
- [Laptev et al., 2016] Laptev, D., Savinov, N., Buhmann, J. M., and Pollefeys, M. (2016). Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 289–297.
- [Latorella and Prabhu, 2000] Latorella, K. A. and Prabhu, P. V. (2000). A review of human error in aviation maintenance and inspection. *International Journal of industrial ergonomics*, 26(2):133–161.
- [Levine et al., 2018] Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436.
- [Li et al., 2014] Li, M., Zhang, T., Chen, Y., and Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM.
- [Li et al., 2002] Li, Q., Wang, M., and Gu, W. (2002). Computer vision based system for apple surface defect detection. *Computers and electronics in agriculture*, 36(2-3):215–223.
- [Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.
- [Liu et al., 2008] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE.
- [Liu and Kang, 2017] Liu, K. and Kang, G. (2017). Multiview convolutional neural networks for lung nodule classification. *International Journal of Imaging Systems and Technology*, 27(1):12–22.
- [Liu et al., 2012] Liu, M.-Y., Tuzel, O., Veeraraghavan, A., Taguchi, Y., Marks, T. K., and Chellappa, R. (2012). Fast object localization and pose estimation in heavy clutter for robotic bin picking. *The International Journal of Robotics Research*, 31(8):951–973.
- [Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

- [Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.
- [Mahler et al., 2017] Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Ojea, J. A., and Goldberg, K. (2017). Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*.
- [Maji et al., 2013] Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. (2013). Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.
- [Marx and Graeber, 1994] Marx, D. A. and Graeber, R. C. (1994). Human error in aircraft maintenance. *Aviation psychology in practice*, pages 87–104.
- [Masci et al., 2012] Masci, J., Meier, U., Ciresan, D., Schmidhuber, J., and Fricout, G. (2012). Steel defect classification with max-pooling convolutional neural networks. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–6. IEEE.
- [McCarthy et al., 2013] McCarthy, C., Feng, D., and Barnes, N. (2013). Augmenting intensity to enhance scene structure in prosthetic vision. In *2013 IEEE international conference on multimedia and expo workshops (ICMEW)*, pages 1–6. IEEE.
- [Mery et al., 2015] Mery, D., Rizzo, V., Zscherpel, U., Mondragón, G., Lillo, I., Zuccar, I., Lobel, H., and Carrasco, M. (2015). Gdxd: The database of x-ray images for nondestructive testing. *Journal of Nondestructive Evaluation*, 34(4):42.
- [Michaels, 2008] Michaels, J. E. (2008). Detection, localization and characterization of damage in plates with an in situ array of spatially distributed ultrasonic sensors. *Smart Materials and Structures*, 17(3):035035.
- [Mobley, 2002] Mobley, R. K. (2002). *An introduction to predictive maintenance*. Elsevier.
- [Moubray, 2001] Moubray, J. (2001). *Reliability-centered maintenance*. Industrial Press Inc.
- [Nilsback and Zisserman, 2006] Nilsback, M.-E. and Zisserman, A. (2006). A visual vocabulary for flower classification. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1447–1454. IEEE.
- [Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528.
- [Palzkill and Verl, 2012] Palzkill, M. and Verl, A. (2012). Object pose detection in industrial environment. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–5. VDE.

- [Pan and Yang, 2009] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.
- [Park et al., 2016] Park, J.-K., Kwon, B.-K., Park, J.-H., and Kang, D.-J. (2016). Machine learning-based imaging system for surface defect inspection. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 3(3):303–310.
- [Parkhi et al., 2012] Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. (2012). Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE.
- [Peng et al., 2015] Peng, X., Sun, B., Ali, K., and Saenko, K. (2015). Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1278–1286.
- [ping Tian et al., 2013] ping Tian, D. et al. (2013). A review on image feature extraction and representation techniques. *International Journal of Multimedia and Ubiquitous Engineering*, 8(4):385–396.
- [Qi et al., 2017] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.
- [Rajpura et al., 2017] Rajpura, P. S., Bojinov, H., and Hegde, R. S. (2017). Object detection using deep cnns trained on synthetic images. *arXiv preprint arXiv:1706.06782*.
- [Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- [Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.
- [Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.
- [Russakovsky et al., 2015a] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015a). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.
- [Russakovsky et al., 2015b] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015b). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.
- [Saenko et al., 2010] Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer.

- [Sarkar et al., 2017] Sarkar, K., Varanasi, K., Stricker, D., Sarkar, K., Varanasi, K., and Stricker, D. (2017). Trained 3d models for cnn based object recognition. In *VISI-GRAPP (5: VISAPP)*, pages 130–137.
- [Schlegl et al., 2017] Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer.
- [Schölkopf et al., 2000] Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., and Platt, J. C. (2000). Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588.
- [Schwarz et al., 2018] Schwarz, M., Milan, A., Periyasamy, A. S., and Behnke, S. (2018). Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter. *The International Journal of Robotics Research*, 37(4-5):437–451.
- [Schönfeld, 2018] Schönfeld, R. (2018). Automatic detection of damaged small parts during overhauling processes. Master’s thesis, Technische Universität München, Germany.
- [Selvaraju et al., 2017] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626.
- [Senthikumar et al., 2014] Senthikumar, M., Palanisamy, V., and Jaya, J. (2014). Metal surface defect detection using iterative thresholding technique. In *Second International Conference on Current Trends In Engineering and Technology-ICCTET 2014*, pages 561–564. IEEE.
- [Simonyan et al., 2014] Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1573–1585.
- [Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [Singh et al., 2014] Singh, A., Sha, J., Narayan, K. S., Achim, T., and Abbeel, P. (2014). Bigbird: A large-scale 3d database of object instances. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 509–516. IEEE.
- [Sironi et al., 2018] Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., and Benosman, R. (2018). Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1731–1740.

- [Smith et al., 2017] Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.
- [Song and Yan, 2013] Song, K. and Yan, Y. (2013). A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Applied Surface Science*, 285:858–864.
- [Spenrath et al., 2013] Spenrath, F., Palzkill, M., Pott, A., and Verl, A. (2013). Object recognition: Bin-picking for industrial use. In *IEEE ISR 2013*, pages 1–3. IEEE.
- [Stanchev et al., 2003] Stanchev, P., Green Jr, D., and Dimitrov, B. (2003). High level color similarity retrieval.
- [Su et al., 2015] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953.
- [Sudakov et al., 2008] Sudakov, S., Barinova, O., Velizhev, A., and Konushin, A. (2008). Semantic segmentation of road images based on cascade classifiers. In *Proceedings of the ISPRS XXI Congress*, pages 601–604.
- [Szegedy et al., 2017] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12.
- [Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.
- [Szegedy et al., 2016] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.
- [Szegedy et al., 2013] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.
- [Tabernik et al., 2019] Tabernik, D., Šela, S., Skvarč, J., and Skočaj, D. (2019). Segmentation-Based Deep-Learning Approach for Surface-Defect Detection. *Journal of Intelligent Manufacturing*.
- [Taheritanjani,] Taheritanjani, S. Overhaulinginspection.
- [Taheritanjani et al., 2019a] Taheritanjani, S., Haladjian, J., and Bruegge, B. (2019a). Fine-grained visual categorization of fasteners in overhaul processes. In *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, pages 241–248. IEEE.

- [Taheritanjani et al., 2020] Taheritanjani, S., Haladjian, J., Neumaier, T., Hodaie, Z., and Bruegge, B. (2020). 2d orientation and grasp point computation for bin picking in overhaul processes.
- [Taheritanjani et al., 2019b] Taheritanjani, S., Schoenfeld, R., and Bruegge, B. (2019b). Automatic damage detection of fasteners in overhaul processes. pages 241–248.
- [Taylor et al., 2011] Taylor, G. W., Spiro, I., Bregler, C., and Fergus, R. (2011). Learning invariance through imitation. In *CVPR 2011*, pages 2729–2736. IEEE.
- [Taylor and Stone, 2009] Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685.
- [Teer and Salem, 1977] Teer, D. and Salem, F. (1977). The formation of low friction wear-resistant surfaces on titanium by ion plating. *Thin Solid Films*, 45(3):583–589.
- [Trost, 1986] Trost, J. E. (1986). Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies. *Qualitative sociology*, 9(1):54–57.
- [Tsai et al., 2007] Tsai, M.-F., Liu, T.-Y., Qin, T., Chen, H.-H., and Ma, W.-Y. (2007). Frank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390. ACM.
- [Uijlings et al., 2013] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.
- [Usman and Rajpoot, 2017] Usman, K. and Rajpoot, K. (2017). Brain tumor classification from multi-modality mri using wavelets and machine learning. *Pattern Analysis and Applications*, 20(3):871–881.
- [Vinzenz, 2020] Vinzenz, L. M. (2020). *Automatic Damage Detection on Metallic Surfaces*. Bachelor’s thesis, Technische Universität München, Germany.
- [Vogel-Heuser et al., 2017] Vogel-Heuser, B., Bauernhansl, T., and Ten Hompel, M. (2017). Handbuch industrie 4.0 bd. 4. *Allgemeine Grundlagen*, 2.
- [Vogel-Heuser et al., 2014] Vogel-Heuser, B., Diedrich, C., Fay, A., Jeschke, S., Kowalewski, S., Wollschlaeger, M., et al. (2014). Challenges for software engineering in automation. *Journal of Software Engineering and Applications*, 2014.
- [Vogel-Heuser and Hess, 2016] Vogel-Heuser, B. and Hess, D. (2016). Guest editorial industry 4.0—prerequisites and visions. *IEEE Transactions on Automation Science and Engineering*, 13(2):411–413.

- [Wan et al., 2013] Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066.
- [Wang et al., 2017] Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., and Tang, X. (2017). Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.
- [Wang et al., 2014] Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393.
- [Wang et al., 2018] Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., and Cottrell, G. (2018). Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 1451–1460. IEEE.
- [Wei et al., 2018] Wei, Q., Ren, Y., Hou, R., Shi, B., Lo, J. Y., and Carin, L. (2018). Anomaly detection for medical images based on a one-class classification. In *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575, page 105751M. International Society for Optics and Photonics.
- [Wu et al., 2019] Wu, Y., Marks, T., Cherian, A., Chen, S., Feng, C., Wang, G., and Sullivan, A. (2019). Unsupervised joint 3d object model learning and 6d pose estimation for depth-based instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0.
- [Wu et al., 2015] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.
- [Xuan et al., 2017] Xuan, Q., Fang, B., Liu, Y., Wang, J., Zhang, J., Zheng, Y., and Bao, G. (2017). Automatic pearl classification machine based on a multistream convolutional neural network. *IEEE Transactions on Industrial Electronics*, 65(8):6538–6547.
- [Yang et al., 2012] Yang, S., Bo, L., Wang, J., and Shapiro, L. G. (2012). Unsupervised template learning for fine-grained object recognition. In *Advances in neural information processing systems*, pages 3122–3130.
- [Yangping et al., 2018] Yangping, W., Shaowei, X., Zhengping, Z., Yue, S., and Zhenghai, Z. (2018). Real-time defect detection method for printed images based on grayscale and gradient differences. *Journal of Engineering Science & Technology Review*, 11(1).
- [Yin, 2011] Yin, R. K. (2011). *Applications of case study research*. sage.

- [Yu, 2020] Yu, H. (2020). Sorting fasteners based on their similarity using siamese networks. Master's thesis, Technische Universität München, Germany.
- [Zeiler, 2012] Zeiler, M. D. (2012). Adadelata: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.
- [Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.
- [Zhang and Lu, 2004] Zhang, D. and Lu, G. (2004). Review of shape representation and description techniques. *Pattern recognition*, 37(1):1–19.
- [Zhu et al., 2016] Zhu, Z., Wang, X., Bai, S., Yao, C., and Bai, X. (2016). Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, 204:41–50.
- [Zilly et al., 2015] Zilly, J. G., Buhmann, J. M., and Mahapatra, D. (2015). Boosting convolutional filters with entropy sampling for optic cup and disc image segmentation from fundus images. In *International workshop on machine learning in medical imaging*, pages 136–143. Springer.