OSNABRÜCK UNIVERSITY

STUDY PROJECT

# Computer Vision Based Asparagus Classification

*Authors:*
Maren Born,
Michael Gerstenberger,
Katharina Groß,
Richard Ruppel,
Sophia Schulze-Weddige,
Malin Spaniol,
Josefine Zerbe

*Supervisors:*
Dr. Ulf Krumnack
M.Sc. Axel Schaffland

*The study report is submitted in partial fulfillment of the requirements for the degree of Master of Science*

*in the*

Research Group Computer Vision
Institute of Cognitive Science

April 9, 2020

OSNABRÜCK UNIVERSITY

# *Abstract*

School of Human Sciences
Institute of Cognitive Science

Master of Science

**Computer Vision Based Asparagus Classification**

by Maren Born, Michael Gerstenberger, Katharina Groß, Richard Ruppel,
Sophia Schulze-Weddige, Malin Spaniol, Josefine Zerbe

TODO: 0.1 Structure of the project:

Some introductory sentences about the project and its contributors.

TODO: 0.2 Structure of the report:

Short overview of every chapter.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

An essential step in the commercial asparagus cultivation is the sorting of the harvested stalks into different quality classes. Depending on size, shape and colour, a label is assigned to the individual spears which determines the class of the asparagus. Nowadays, this task is usually performed automatically, with modern asparagus sorting machines achieving a throughput of up to twelve spears per second (https://www.heringinternational.com/fileadmin/media/archive1/downloads/presseberichte/2015-09-23_FAZ_-_Verlagsspezial_Industrierregion_S%C3%BCdwestfalen_-_Textilbeton_f%C3%BCr_Dior.pdf). However, the accuracy of such machines can be unsatisfying. Manual resorting is usually necessary, and thereby causes substantial effort.

The aim of this study project was to investigate how techniques from computer vision, both classical and deep learning based, can be applied to improve classification results for the asparagus sorting machines. The metric for the sorting corresponds to the quality of an asparagus. The quality in turn is defined based on a number of features that represent differences (or even flaws) in colour, shape, or texture. Hence, improving sorting algorithms for the classification of asparagus requires reliable means of measuring these features. As such, we were interested in the question how the quality-defining features of asparagus spears can be estimated using state of the art computer vision and machine learning techniques.
Besides the systematic engagement with feature detection methods from the field of machine learning, the study project was also used as a training ground for project management. The hands-on experience on a long-term project gave the chance to learn how to distribute work and how to organize ourselves in a larger team.
The project was supported by a local asparagus farm and a company specializing in asparagus sorting machines. They provided training data and expertise on the classification of the spears. The idea to apply new machine learning approaches to a commercial problem of the agricultural industry seemed promising and challenging at the same time. As the data received from the farm was unlabelled, a larger focus on the preprocessing of the recorded image data became inevitable. Further, the variance in quality classes and the subjective view of human sorting behaviour toughened the perspective on a practical application into the actual sorting machine. Nevertheless, the original goal of studying different techniques in computer vision and testing their usability in a restricted hardware setting for asparagus classification gave valuable insights into the practical application of previously only theoretically assessed knowledge.
Over the course of one year, the team members worked on the implementation of different approaches to tackle the issue of image classification. After intense engagement with the subject, the methodologies and the project outcome were documented and critically reviewed. On the following pages, the different stages of the project together with the results of the applied computer vision approaches are described in detail. The report chapters are mostly in chronological order, each with

the focus on a different stage of the study project throughout the year.

In the first chapter, the idea of the project is presented together with an introduction to the current standards of image classification in machine learning as well as a general background on the quality classes, the sorting process, and classification of asparagus in the agricultural industry. Further, the results of the project are summarized in short at the end of the chapter, to provide a comparison of the expected outcome at the start of the project with the results that were achieved after the year.
The second chapter comprises the organizational background to the project. This includes task management and teamwork, as well as a thorough evaluation of our planning abilities as a group. Further, the process of data collection and a first inspection of the sorting machine are elaborated, with an emphasis on the first issues of unlabelled data. Unfortunately, the research for scientific literature, specifically regarding our topic of agricultural application of deep learning approaches, proved mostly disappointing. Nevertheless, it gave an insight into practical possibilities and a rough guideline for our endeavour. The collected literature that was found to be close to our topic is reported at the end of the second chapter.
In the third chapter, the preprocessing phase is captured with the building of the data set at its end section. The first classical machine learning techniques for automatic feature extraction on image data are explored, followed by the process of manual feature extraction with the usage of a labelling application. The resulting outcome is reported, evaluated, and the labelled image data is transformed into a data set for the further training of the chosen models.
The fourth chapter constitutes the different approaches that were chosen to tackle the issue of image classification. An emphasis was put on the application of deep learning techniques for the classification of the asparagus. As the collected data includes labelled and unlabelled samples, methods from the range of supervised learning to unsupervised learning were tested.
The fifth chapter contains the results of the project. The outcome of each approach is described and compared to other methods as well as to the original classification accuracy of the sorting machine at the asparagus farm. Further, all approaches are critically reviewed and discussed with an evaluation of their practical application in the food industry. The overall outcome of the project is judged as a scientific study and as a team management experience.
In the sixth chapter our findings are set into a broader perspective. The project and its results are summarized on a scientific basis as well as on the organizational level while future prospects of the project are assessed.

Before delving any deeper into the specific context of software development for the application in classification tasks, however, a thorough insight into the idea of the study project is given in the following chapter. Further, a rough background on the addressed topics of machine learning and the sorting of asparagus are provided. In 1.1 *The project*, the objective of the study project is introduced. The next section, 1.2 *Background on computer vision-based classification tasks*, gives an overview on the machine learning techniques that explore how image classification can be implemented and how these approaches can provide a solution to our issue. In 1.3 *Background on sorting asparagus*, the process of asparagus classification in the commercial food industry is illustrated and the different quality classes of asparagus are defined. The first chapter concludes with 1.4 *Expected outcome vs actual outcome*, where the results of the study project are presented on a broad level and compared to the expected outcome.

## 1.1 The project

The objective of the study project was to find both, conventional and neurally-inspired computer vision based approaches which can be tested for their practical application in the commercial sorting of white asparagus. The different methods were implemented based on the image data that was received from the automatic sorting machine Autoselect ATS II employed by the asparagus farm "Gut Holsterfeld". The aim was to check whether approaches from the fields of machine learning and computer vision can be applied to improve the classification behaviour of the asparagus sorting machine and, thus, if they can be used in a more industrial than scientific setting regarding the specific problem of asparagus classification. The initial intention to directly implement new software into the machine was put in the back to make room for intensive research on the different approaches and on fine-tuning the received data to build a practical data set to train neural networks. The idea for the project was formed by one of the students of the study group. She was in contact with the asparagus farm "Gut Holsterfeld" and had received note of the unsatisfying classification performance of its sorting machine. The practical application to a real world problem sparked the interest of the group and the general curiosity towards computer vision, in particular neural networks, further inspired to deal with the sorting issue in a deep learning context.
All project members were students in the field of Cognitive Science at the University of Osnabrück. The study project is part of the Master Program in Cognitive Science at the University of Osnabrück. It was supervised by Dr. Ulf Krumnack (https://www.ikw.uni-osnabrueck.de/en/research_groups/computer_vision/people/krumnack_ulf.html) and Axel Schaffland, M.Sc. (https://www.ikw.uni-osnabrueck.de/en/research_groups/computer_vision/people/schaffland_axel.html).
The intention of the study project is to confirm the ability of its participants to independently formulate and solve an unknown problem from the scientific context of one subject area using the methods and terms of a previously theoretically learned subject (Universität Osnabrück, 2019b) (Universität Osnabrück, 2019a). This includes the documentation and presentation of the results, the methodologies as well as the reflection on the work process. Within the scope of the project, for example, the development of software, analysis and interpretation of statistical data material are practiced. A further aspect of the study project is to deepen the communicative and decision-making competence of its participants (Universität Osnabrück, 2019a). The idea is to train independent project work in groups of students under conditions that are common for research projects in science or industry.
As the project took part in the scope of an examination for the Cognitive Science master program at the University of Osnabrück, most of the work was developed at the university, that is, at the Institut für Kognitionswissenschaften (IKW). The project was realized as a cooperation with a local asparagus farm "Gut Holsterfeld", at Rheine (https://www.gut-holsterfeld.de/). Additional image data was received from the asparagus farm Querdel (https://www.querdel.de/). Further cooperation existed with the mechanical engineering company HMF Hermeler Maschinenbau GmbH (www.hmf-hermeler.de) that developed the sorting machine Autoselect and provided valuable expertise on the sorting issue.
The documentation to the project can be found at (https://asparagus.readthedocs.io/en/latest/), all associated software is stored in the Github repository CogSciUOS/asparagus (https://github.com/CogSciUOS/asparagus) and at the institute internal storing system (PATH).

In the next section, an introduction to the field of computer vision based approaches

for the classification of data is given. It provides a theoretical background on the broad selection of different techniques that can deal with the objective of this project, namely the improvement of sorting commercial asparagus into quality classes with a machine.

## 1.2 Background on computer vision based classification tasks

In the following chapter, we are going to describe the current standard of computer-based image classification. Our main focus will be on artificial neural networks and classical computer vision techniques. We will give a broad overview of relevant topics and underline their importance for our project. Computer vision is a field of computer science that aims to automatically extract high-level understanding from image data and provide appropriate output. It is closely linked with machine learning, which describes the ability of a system to learn and improve from experience rather than being specifically programmed. Machine learning is frequently used to solve computer vision tasks.

Image classification is one of the main subfields of computer vision which gained a lot of attention in the scientific as well as the economic world. Besides the classical computer vision techniques that use algorithmic approaches to determine patterns, edges, and other points of interest that can help to classify images, artificial intelligence was introduced to the field in the 1960s. Since then, more and more creative and complex artificial neural networks were introduced to solve numerous classification tasks including letter, face, and street sign recognition among others (Mirończuk and Protasiewicz, 2018) (Balaban, 2015) (Stallkamp et al., 2011).
Some experts in the field claim that artificial neural networks revolutionized the field of image classification, yielding better results than ever before (He et al., 2016a) (Krizhevsky, Sutskever, and Hinton, 2012a). More and more challenges, for example ImageNet, were introduced and computer scientists all over the world implemented creative solutions for the proposed problems. Additionally, influential companies like Google had a deep interest in finding solutions for image-based classification problems and pushed the research on this and related topics even further. In the era of optimization, computer-based classification became indispensable in many industries and also found its way into agriculture which is the field of interest in this study project. In the following, a short introduction in both classical computer vision techniques and artificial neural networks is given which will span a bridge to our current classification problem.

Neural networks are used for image classification in many domains. In contrast to the algorithmic approaches, it is not determined by the programmer how and what exactly the model learns. A large amount of data is provided to the model, which then extracts relevant information to learn the classification task. But what is relevant to the model is not necessarily relevant or interpretable to a human observer. This is one of the biggest disadvantages compared to the classical computer vision approaches, which are understandable and, therefore, interpretable and adjustable. Another problem that comes along with this is that the bias for those approaches often does not lie in the code itself but in the data, which is by far more difficult to detect and surpass. For this reason, many see artificial neural networks as a black box, which may yield great results but can never be fully understood. Recent advances try to tackle that issue by researching artificial intelligence systematically aiming at making it explainable (Tjoa and Guan, 2019) (Gilpin et al., 2018).
In image classification, usually convolutional neural networks (CNNs) are used,

which are inspired by the visual cortex of the brain. The idea is that highly specialized components or, in the case of CNNs, filters learn a very specific task, which is similar to the receptive fields of neurons in the visual cortex (Hubel and Wiesel, 1962). These components can then be combined to high-level features, which can then in turn be combined to objects that can be used for classification. In CNNs this concept is implemented by several successive convolutional layers in which one or more filters are slid over the input generating so-called feature maps. Each unit in one feature map looks for the same feature but in different locations of the input. In recent years, CNNs became so good that they outperform humans in many classification tasks (Russakovsky et al., 2015)(http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/).

Although machine learning exhibits very promising results and a lot of research and literature is available on the topic, many branches of industry still rely on traditional computer vision techniques in their implementation of image classification. This also applies to the asparagus sorting paradigm. To the best of our knowledge, no asparagus sorting machine is currently on the market that uses artificial intelligence for its classification algorithm. This brings us to the second topic of this chapter, namely the classical computer vision algorithms.
Many classical computer vision algorithms aim to detect and describe points of interest in the input images that can be generalized to features. For these features, low-level attributes such as rapid changes in colour or luminance can be used. In contrast to the features learned with the help of machine learning, these features are not specific to the training data set and therefore do not depend on it being well-constructed. Further, they are usually created in a way that is interpretable by humans which makes them very flexible and easily adaptable to specific use cases. This is why in some cases, traditional computer vision techniques can solve a problem much more efficiently and in fewer lines of code than machine learning approaches.

In summary, both approaches have interesting implications for computer-based image classification tasks and provide us with promising techniques for our problem of asparagus classification. While we relied mostly on machine learning for the classification task itself, traditional computer vision algorithms were used to detect important features from the images, which not only helped us in the labelling process but also can be used for some of the binary classification tasks of the hand-labelled features.

## 1.3 Background on sorting asparagus

In this section, the different sorting classes for asparagus are discussed with a focus on the classification of the 13 quality classes as characterized by the asparagus farm "Gut Holsterfeld" and implemented for our project aim. Two additional sources of information for the following issue were the owner of the asparagus farm "Gut holsterfeld", Mr. Silvan Schulze-Weddige, and the CEO of the engineering company "HMF Hermeler Maschinenbau", Mr. Thomas Hermeler.

While asparagus accounts for a fifth of the area used for vegetable cultivation in Germany, the harvesting season of white asparagus only spans over a period of 2 months, beginning in April and usually ending on the 24th of June (Statistisches Bundesamt (Destatis), 2017) (Weber and Quinckhardt, 2018). During this period, the asparagus is harvested, classified, and sold in accord with a price range that is

defined by the quality class of the asparagus.
In the European Union, there is a uniform system for the sorting of asparagus into quality classes (Europäische Komission, 1999) (United Nations Economic Commission for Europe (UNECE), 2017) (https://mlr.baden-wuerttemberg.de/de/unser-service/presse-und-oeffentlichkeitsarbeit/pressemitteilung/pid/nationale-handelsklassen-f
//www.bzfe.de/inhalt/spargel-kennzeichnung-5876.html). However, supply and demand usually determine the number and accuracy of these categories. One of the first defining features is the colour of the asparagus which comprises four categories: white, violet, violet-green, and green (Europäische Komission, 1999). For this project, only the first two colours were of relevance. Further distinction is made between class Extra, class I, and class II, where the class Extra defines the product as perfect quality, class I defines it as good quality, and class II includes products that do not qualify for the other classes but satisfy the minimum requirements for commercial distribution (Europäische Komission, 1999). A last distinction is made in the characteristics length and width. White and violet asparagus may not exceed 22 cm in length. The minimal length of the spear is above 17 cm for long asparagus, 12-17 cm for short asparagus, 12 cm for class II, and less than 12 cm for fractured spears with intact heads. Additionally, there is some level of tolerance accepted for the quality classes. In class Extra, 5% of class I asparagus is tolerated. Class I allows for 10% wrongly sorted asparagus of class II. In class II 10% of spears are accepted that do not suffice for the minimum requirements of asparagus trade. The last quality difference that has to be fulfilled on a national basis is that there are no more than 15% of hollow spears tolerated in one package or bundle.
Depending on how carefully the individual farmer further categorizes the asparagus, additional classes can be established. Regional differences to those classes make the challenge for the manufacturers of sorting machines even more complicated. The classes as stated in this report depend solely on the views of the farm Gut Holsterfeld and do not necessarily apply to any other classification system of the asparagus industry.
In the Table 1.1, 14 quality classes are shortly described, of which 13 classes were relevant to this project. The classification tree in Fig 1.1 illustrates the decision process that underlies the categorization of the relevant 13 asparagus classes.
One of the challenges of asparagus classification with a machine lies in the human sorting error. While the human sorter can distinguish between colours or between shapes like bent or flowering, a machine works even more precisely. The machine can assess exact calculations about the length or the width of a spear. Thus, a threshold has to be defined if a spear is sorted into a certain class. However, the data on which the machine calculates its features to characterize an asparagus spear was previously labelled by a human. The subtle differences in colour might have escaped the sorter and hence the machine will sort a spear into the category Rost while the spear would be judged as category I A Anna by the human sorter. Since the human sorting behaviour is subjective, the same asparagus can be categorized differently by two independent sorters. Furthermore, an asparagus sorted twice by the same person at different points in time might not end up in the same category than before.
A second problem for the sorting machine poses the distinction between different colours. The machine might not be able to distinguish whether the asparagus was, e.g. very rusty or simply very dirty in certain cases.
A third factor for classification difficulties is caused by the restricted view of the product. An asparagus can look perfectly shaped from one angle but might be damaged on the side that is not exposed to the camera of the sorting machine.

| Label | Description |
|---|---|
| I A Anna | In this context, quality class Extra is represented by I A Anna, I A Bona, and I A Clara. I A is defined as asparagus that is perfectly straight and white. There are no large pressure marks, no rust, no violet colour, and no curvature. The identification label Anna marks that the width (diameter) of the spear is in a range of 20 to 26mm. |
| I A Bona | Class Extra asparagus and the spear is of width 18 to 20 mm. |
| I A Clara | Asparagus with a width of 16 to 18 mm and of quality class Extra. |
| I A Krumme | The asparagus fulfills all criteria for class I A while it shows a slight curvature. |
| I A Violett | The asparagus is of a violet complexion, while it complies with all guidelines for quality class I A. |
| II A | The asparagus is both curved and violet. |
| II B | The asparagus is curved, rusty, and/or in any other way damaged or classified as defective product. |
| Rost | The asparagus shows traces of rust or has rusty parts. |
| Dicke | The asparagus exceeds the norm of 26 mm in width. |
| Hohle | The asparagus is hollow from the inside. |
| Blume | The head region of the asparagus is about to bloom or it shows clear outlines of a flower. |
| Suppe | The asparagus has a width of less than 16 mm. |
| Bruch | Any asparagus that is below a length of 21 cm. However, another distinction in this category is made between asparagus that has an intact head and a length of at least 100 mm (Kerze), an asparagus without head (Bruch), and an asparagus head alone (Köpfchen). |
| Keule | The upper end of the asparagus is thicker than the lower part. The shape is similar to a club, hence the name of the class. This category was of no concern to the project because no image data of this category could be recorded. |

TABLE 1.1: In this table, 14 quality classes for asparagus categorization are listed and described, according to the asparagus farm "Gut Holsterfeld". Except for the class Keule, all 13 quality classes were used as the goal label for the asparagus classification.

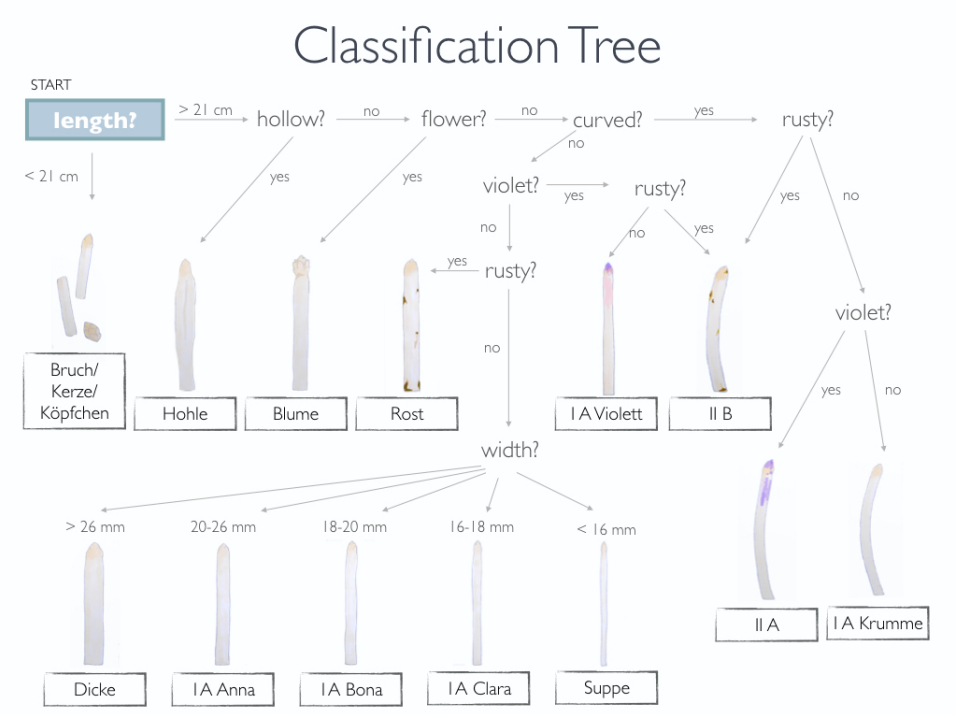



FIGURE 1.1: The classification tree shows how each asparagus is attributed with a label of its quality class. It follows the sorting rules of the asparagus farm "Gut Holsterfeld". Starting from the upper left corner of the image, mostly binary decisions are made until a label is reached. The width of the asparagus is further subdivided at the end. Any further damaged or defective asparagus automatically belongs in category II B.

Another complication poses the question of demand and supply. During some seasons there is more asparagus of a certain class and less of another class is available. The farmer usually arranges the number of spears belonging to a quality class according to the seasonal conditions. For example, during a season with less high quality asparagus of classes I A, the sorting threshold will shift. Spears that are usually sorted into, e.g. a lower class will now belong to a higher class. The machine has to be flexible to accord for this change of preferences.
These challenges are not impossible to overcome but they make it more difficult to find a solution to the sorting task and compromises might be necessary.

To summarize this subchapter, there are 13 quality classes of asparagus relevant for this project. The spears are labelled according to features that influence the price of a spear. Further, the challenge for a sorting machine will not only be to sort for these criteria but to provide a flexible and individual solution in accord with the farmer.

## 1.4 Expected outcome vs. actual outcome of the project

Based on the literature review, we aimed to improve the current sorting performance of the ATS II machine at the local asparagus farm "Spargelhof Gut Holsterfeld". In this report, we investigated techniques from computer vision, both classical and deep learning based approaches. We expected to reach a result which is able to classify asparagus images into 13 different classes better than the current

standard. For the initial performance of the sorting machine, no reliable accuracy of correctly sorted asparagus pieces is available, but between three and six workers are employed to re-sort wrongly classified asparagus pieces. The farmer himself assumes an accuracy of not more than 70%.

An advantage of our project is that it was directly supported by the local asparagus farm, providing training data and allowing us to evaluate our proposed solutions in a real environment during the asparagus harvesting season 2020.

However, there was a misunderstanding between us and the supporting asparagus farm about the kind of data we need. The existing images were too few, and also unlabelled. Therefore, we spent the first two and a half months with data acquisition instead of starting with preprocessing as we originally planned. Throughout the harvesting season 2019, we continuously went to the asparagus farm and collected unlabelled asparagus images during the normal harvesting procedure (see also 2.3). For 12.962 images (for more detail see 2.3), we collected labelled data by taking images of pre-sorted asparagus pieces. Pre-sorted in this context means that the asparagus pieces were sorted by the sorting machine and if needed re-sorted manually by professional workers. We would have wanted to collect more data this way, but this was not possible, as the asparagus quality suffered from it (for more detail see 2.3).
The number of labelled images is insufficient to learn classes using deep learning approaches (Russakovsky et al., 2013) (Russakovsky and Fei-Fei, 2010) (https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/
Therefore, we spent six months preprocessing and labelling the data manually. Preprocessing involved: organizing the large number of images, renaming the files, so that the three images of one asparagus piece can be accessed together, and performing automatic feature extractions (ref to preprocessing).To label the images by hand, we wrote a custom application (reference hand-label-assistant). The final labelled data set contains over 10.000 (13.319) labelled asparagus pieces. Next, we worked on numerous classical and deep learning approaches (ref to chap 4). We reached several very interesting results which will be discussed in Chapter 4. Although we developed an end-to-end prototype, we did not deploy it onto the sorting machine for the harvesting season 2020. Even though some of our results are highly promising, it is therefore difficult to compare it to the current performance of the machine.

*Hier noch ein Absatz mit den Inhalten aus der Conclusion – sobald diese steht.*

# Chapter 2

# Data acquisition and organization

The value and purpose of a study project does not only lie in the implementation and realization of a long-term scientific study - in our case, computer vision-based label prediction to improve an agricultural sorting machine. The intention is also to learn how to organize, schedule, and distribute tasks for a team over the period of one year.

To accomplish the objective of the project, a working structure had to be established. The project members had to learn each other's strengths and weaknesses to form a robust and efficient unit. Self-organization and team building became key to the success of the project. Thus, the secondary focus of the report lies on the management of the study group.
The process of team building and structuring is captured in the first half of this chapter. A timeline of the different working stages can be traced in a roadmap created at the start and updated at the end of the year. It reveals how well the team succeeded in predicting the duration of the various tasks.
Further, the communication between the members is addressed which includes regular meetings, agreements on different issues, or the working platforms that were used. Additionally, the teamwork is evaluated which comprises task distribution, the democratic working structure, and the structural integrity of the team.
After the organizational part of the project, the second half of this chapter is concerned with the collection of the data and the literature research. The sorting machine is described as well as the process of saving and retrieving the recorded images. It is followed by a summary of the literature which was thought to be relevant to our topic, regarding the visual detection of agricultural products or images with low variance with machine learning approaches. There was no specific paper, however, which could be used as a basis for our project.

The first section of this chapter, 2.1 *Roadmap of the project*, gives an overview of the time management of the study group. It is followed by the chapter 2.2 *Organization of the study group*, in which communication and teamwork are assessed. In 2.3 *Data collection*, the acquisition of the data from the sorting machine is described in more detail. The last section, 2.4 *Literature research*, presents the retrieved literature concerning the issue of asparagus classification and agricultural product classification with machine learning based approaches.

## 2.1 Roadmap of the project

At the beginning of the project, a roadmap was created to structure the year into different working stages as well as to have an overview of the tasks and problems that needed to be addressed. Near the end of the year, this map was evaluated and updated to mirror the actual time spent at each project stage and to check how

accurately the project had been planned from the start. If there were great discrepancies between estimation and reality, the map helped to acknowledge the wrong judgement of time and which working stages had taken longer than others.

The following timetables in Figure 2.1 give a broad outline of the major stages of the project. In the left figure, it was estimated how much time for a specific phase is needed, whereas in the right figure the time spent for the stage is given. Both figures are structured to display the year in a circle, starting in April of 2019, and running clockwise through the year until April 2020. The months are represented by the inner circle while the outer circle marks the different working stages.
The project comprised four to five major stages. The following stage descriptions are shortened for the purpose of simplicity. A more detailed representation of the single tasks attributed to each stage can be found in the ensuing roadmaps in Fig. 2.2 and Fig. 2.3. The project started with the Data Collection and the Organization of the study project. During the first stage, the images were recorded with the sorting machine, while the major planning and research for the project took place. In the second phase, most of the Preprocessing happened, that is, preparing and labelling the image data. This phase is split into two phases for the right figure displaying the actual time, 'Preprocessing' and 'Labelling'. The classification stage includes the time spent on the machine learning approaches which were implemented and trained on the asparagus data. In the last stage, the approaches were evaluated and their results were compared. It is noteworthy that the different stages overlapped to a certain degree, because minor tasks of a previous stage were still in progress while tasks from the subsequent stage had already started. For the purpose of this figure the start and end time is displayed as a hard boundary.

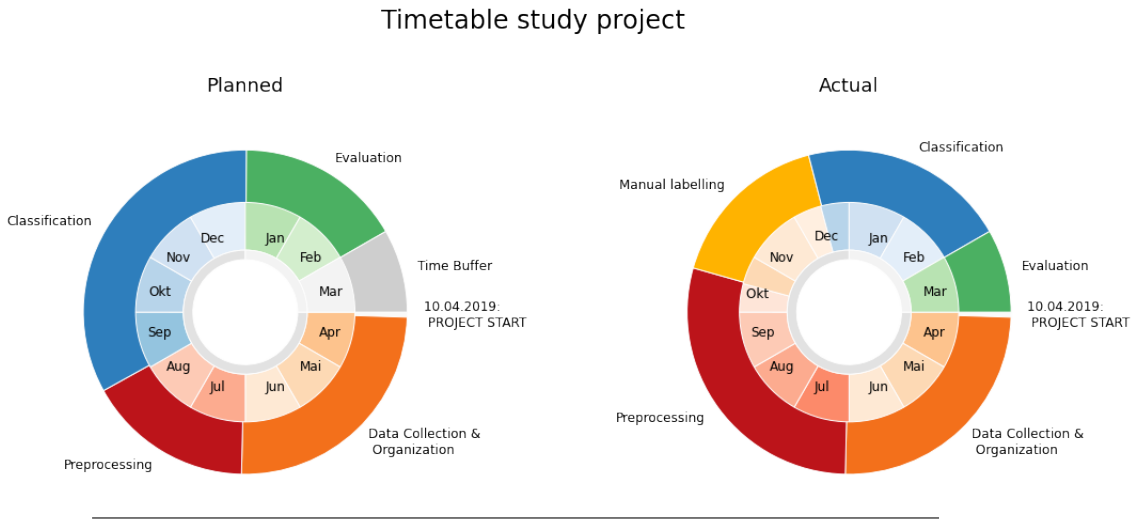



FIGURE 2.1: On the left side is a timetable with the estimated time of the study project during the year from April 2019 to April 2020. To the right, the timetable displays how the time was spent.

When comparing both figures, some distinctions can be recognized. The figure to the left shows that - while it was noted that the Preprocessing step of labelling the data would take some time - it was not estimated to take longer than until the end of August. Compared to the Preprocessing step in the right figure, it is observed that the phase continued until October. Furthermore, the time for labelling a sufficient amount of images was underestimated. In the right figure, the labelling

process received its own stage to highlight the fact that it demanded much more effort and time compared to the initial estimation. This exchange required that the time buffer and parts of the time calculated for the later stages of classification and evaluation had to compensate for the time loss. The time difference can best be seen when comparing the respective colours (that is, the brighter colours of red, orange and yellow to the darker colours of blue and green). While the main focus of this project was supposed to be the application of different machine learning techniques to classify the data, the data generation and preprocessing posed to be the most time-consuming stages.
As a result, both timetables differ in that the year was more optimistically planned than realized. A major factor was the lack of experience of the participants concerning not only the conduction of a larger project with many co-workers but also the general implementation of the preprocessing stage for machine learning classification. Another factor influencing the shifted timeline was the working structure of the team. It was re-evaluated during October to better fit the needs of the group members and avoid distributing so-called 'bottleneck' tasks (that is, tasks that are decisive for the continuation of the next steps) to single members only. All in all, a valuable lesson learned during the project was to not underestimate a preprocessing phase.

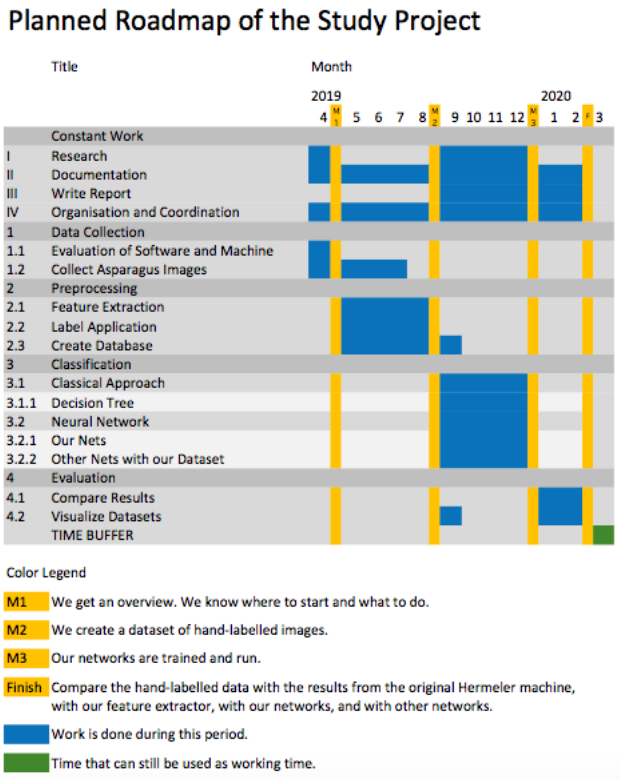



FIGURE 2.2: The figure shows the planned roadmap of the study project. It reveals how the time needed for each task was estimated in the beginning of the project.

In Figure 2.2 and Figure 2.3, the stage specific tasks can be seen in more detail. Again, both figures display the estimated time and the actual time, respectively. The headlines serve as the division into the major stages except for the first heading, 'Constant Work', which shows the tasks that demanded continuous attention and effort throughout the year. The duration of tasks is represented in blue, while

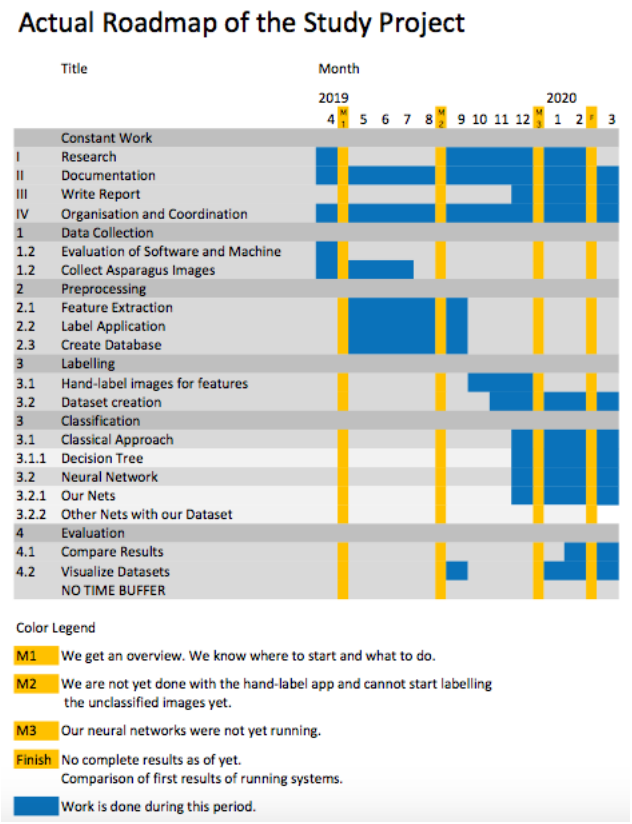



FIGURE 2.3: A roadmap that shows how the time was spent.

the yellow lines mark milestones that are explained in their legends.

Comparing both roadmaps, the shift of focus is seen more clearly and can be associated with the duration of single tasks. Especially the prominent drift of the classification stage mirrors the fact that the time estimation is worth improving. The roadmap helped to better assess the time needed for task completion. In the following chapter, the management of the work distribution and the communication are taken into focus.

## 2.2 Organisation of the study group

In this chapter, we will describe how we have organized ourselves to work together on the project. For this we will elaborate on the tools we used for communication and organization as well as on the structure of our group work and our teamwork in general.

### 2.2.1 Communication

Our main way of communication were frequent meetings in which we discussed our process work period and distributed tasks for the next one. In addition to those meetings, we used different platforms, which worked with varying degrees of success. After a presentation about possible strategies of cooperation, we initially decided to use Asana, GitHub and Telegram to facilitate the communication outside of our meetings. The different means of communication will be described and evaluated in the following abstract.

Regular meetings have been the most helpful in organizing our project. During the meetings, which usually lingered for one or two hours, there was a discussion leader and a protocol writer. The protocols were saved for review in our GitHub project. The meetings were characterized by long discussions about the best approach for the following steps and the possibilities to tackle the next challenge. Our supervisors were almost always present at the meetings, they brought in their expertise and gave us the opportunity to ask concrete questions. During the first half of the project, tasks were always distributed at the end of each meeting and in the next meeting the progress of the work was discussed. The communication was changed in the second half of the project, where we started to use a schedule that described the different tasks and deadlines in detail, and to gather for co-working. The organizational meetings were continued and used to discuss the current status with Ulf and Axel. Everyone reported about his area of responsibility. This helped us to spend less time discussing and debating and more time working on our tasks. Telegram is a cloud-based instant messaging service for the use on smartphones, tablets and computers. Since the first meeting, we had a constant conversation in a group chat on Telegram, in which we arranged us for the weekly meetings, planned handovers of keys/transponders for the room or informed each other about the status of the project. The group chat also created space for mutual motivation when needed. The majority of important information was exchanged via telegram.
Asana is used to distribute tasks and to communicate projects successfully. Many integrations of other applications, such as Slack, can help to achieve this. We used a board view where we listed tasks in different sections. But this function alone did not help us much in our project. The tasks were easier to distribute in direct consultation at physical meetings and demonstrated or discussed after completion. So it happened that asana was not used enough to be helpful. If we had relied on communication with Slack or other agreed services or applications, it might have made more sense, but asana alone was proven to be inefficient in our use case.
GitHub is a web-based popular plattform using a version control system using Git that helps developers store and manage their code, and track and control changes to their project. During the project, we learned how to use it. Initially, a presentation was given by Katharina on how to use Git, because there are a few rules to follow to keep it straightforward. Git allowed us to work from anywhere, which was very helpful for us. We also automatically created a documentation with Sphinx. This means that by using the GitHub project in the right style, the protocols, the work schedules, the manuals, and our code comments are automatically described in our documentation (https://asparagus.readthedocs.io/en/latest/).

### 2.2.2 Teamwork

This subchapter outlines the team forming, the team members and the cooperation in our study project. It starts by introducing the team members and their previous experiences. This is followed by a description of the practical aspects of teamwork, the work structure, and the distribution of project-relevant tasks.

First, the team members and their respective backgrounds are illustrated before delving into further work-related task distributions. The project was an initiative of one of the students and, hence, a large part of the project members are friends that were inspired by her ideas. Other students joined the project after its public announcement to complete the team. Thus, the group consisted of members with varying degrees of knowledge about each other, which had an influence on the dynamics of the teamwork. The team was initially made up by Malin Spaniol,

Maren Born, Katharina Groß, Josefine Zerbe, Michael Gerstenberger, Richard Ruppel, Sophia Schulze-Weddige, Luana Vaduva, Thomas Klein and Subir Das. None of the members had yet worked together as such on a project of this scope. During the course of the project, three members left the team for various reasons. Thomas left in July due to a change in his study program. This was an unfortunate occurrence because he posed a valuable source of knowledge in the field of computer vision. Further, Luana and Subir left in October to pursue different study projects.

By the start of the project, all except two of the members were in their master's degree in Cognitive Science at the University of Osnabrück. The members brought a wide variety of backgrounds into the team through different bachelor programs or different majors in the broader field of cognitive science. In the beginning of the project, the team members had little to no experience in the application of computer vision or neural networks. The motivation of most students was to pursue new and interesting tasks in these fields. Four students had a theoretical background in computer vision, six students had gained some experience with neural networks through the course 'ANN with Tensorflow', taught at the University of Osnabrück, while some had also taken machine learning classes during their study program. None of the members had prior knowledge about project management or task organization on a broader level. Git was previously only used by three students so far, but none of them were experts on its usage. Further, no participant had any experience with the Grid system of the IKW, not to mention running jobs on different machines. Thus, the project started with 10 members, where each of the participants brought a different level of experience in the most relevant fields of machine learning, that is computer vision and neural networks, into the team.

After introducing the team members and their backgrounds, this section continues with elaborating the structural organization of the team and the work distribution. As none of the members had any previous experience with team formation, in the beginning, the team lacked some structure and a clear distribution of single roles inside the team. One reason for this was the harmonious atmosphere between the single members. Many of the team members had a friendly relationship already from the start of the project. Also, further tasks at the beginning, like the trips to the asparagus farm, strengthened the team spirit and the social interactions positively. So we started with a very dynamic structuring of tasks by making every decision democratically. As described in more detail in the next chapter, we first used the joint meetings to distribute the tasks. As an example, we had to prepare a schedule for organizing the data collection and already started with preprocessing tasks. These two main activities were mostly done in smaller teams of two to three people. Rather few tasks were done alone. In our meetings, we often formulated possible next tasks, even for the distant future, but as the project progressed we did not assign them to a specific person or working group.
In August, we restructured our own organization. On the one hand, this was due to the fact that the tasks changed and thus a new structure was more appropriate. On the other hand, also the strength of the individual team members were a reason for the restructuring. Some team members had less programming experience than others and therefore had difficulties realizing certain tasks at the same time and with the same precision than others. Even if they had good ideas in terms of concept, it was not possible for them to implement these ideas quickly enough so that they could be included into the project. Nevertheless, there was still the possibility to grow in his new assignments. In addition, the democratical self-organization

and difference in programming expertise led to a distortion in the time management of the group and some tasks were not finished in time. To integrate more of the strengths that the single team members brought and to tackle the issue of time management, it was decided to write a roadmap that distributed the work more appropriately, gave an overview of the tasks that still had to be done and how much time was left to do them.
Further, common working hours on campus were introduced as well as a division of responsibilities which were work-related and related to the supervision of tasks. The common working hours ensured that questions and decisions that arose could be discussed directly. This was especially helpful when different tasks overlapped and required communication and agreement. The supervision of the work was divided into manager roles, which means that the work was split into different main fields where each member was responsible for managing their assigned area, distributing tasks and keeping an overview of the relevant work inside their working field. Furthermore, one knew who to turn to for questions and when in need of discussion or feedback. The meetings became more effective due to the new structure, and there was less discussion concerning task distribution. As we distributed roles, we were also more responsive to the strengths and weaknesses of the group members. Therefore one can say that the team structure and distribution of work changed over the course of the project. The strengths of single members were used more efficiently and the supervision of working areas led to a more structured supervision and manageable task distribution.

In the course of the project and to sum up the focus of this chapter, namely the organisation of our study group, it can be said that we had a harmonious team with two different working structures. The first one cultivated at the start of the project and the other one at the end of the project, respectively, which happened due to the tasks amount and the optimisation of our teamwork. We have all learned that teamwork is not a trivial issue, even for members who already know each other well.

## 2.3 Data collection

In this section, the process of recording and assembling the data is described, together with information about the asparagus sorting machine 'Autoselect'. For the data collection, the project group had to drive to the asparagus farm "Gut Holsterfeld" at Rheine. First expectations to receive only labelled data were disappointed. The farmer was not able to provide the amount of labelled data that was needed for the project. As the asparagus season was about to start, the first two months of the project were mainly dedicated to data collection.

First, the asparagus sorting machine at Gut Holsterfeld is described before reporting about the process of labelled and unlabelled data collection. The machine 'Autoselect ATS II' (2003) is designed for sorting white asparagus and green asparagus (HMF Hermeler Maschinenbau GmbH, 2015). The sorting is based on the parameters width, length, shape, curvature, rust, and colour. A total of 30 criteria for classifying an asparagus spear are used to describe these parameters. The calculation of the single features is based on a classical analytical approach. For example, the feature colour is composed of 8 sub-parameters. Each spear is reviewed at different areas (the head of the asparagus, the area below the head, and the stem, respectively) and judged for its hue in percentage. The values are compared and,

(A) (B) (C)

FIGURE 2.4: Example pictures of the quality class I A Anna. In picture (A), the asparagus spear is to the right, in (B), it is in the middle, and in (C) it is to the left.

according to a threshold, the spear is sorted into a colour category. For all parameters, there is a minimal threshold and a maximal threshold. As another example, the parameter width is calculated at three points at the asparagus (top, middle, and bottom part). From these three values, an average value is calculated that decides in which category the asparagus is sorted. If an asparagus exceeds the maximal threshold, it is not recognized and cannot be sorted accordingly. Thus, all parameters have to have an upper and a lower threshold, including parameters like shape, curvature, and colour. When evaluating which thresholds to choose, it is recommended to check that most asparagus spears tend to be in between the average value and the maximal threshold, with a larger tendency to accumulate around the average value. All parameters can be freely chosen by the farmer and can this way be fitted to the needs of the respective asparagus farm.
According to the manual, the number of classifications (quality classes) is selectable but it is based on the number of output trays available to the machine. The machine Autoselect at "Gut Holsterfeld" has 16 trays. The program of the Autoselect processes the sorting parameters from the best class to the worst class. The user can define the order of parameters. The manufacturer suggests to first sort for length and width, then sort for colours parameters, and analyze the shape parameters last. The sorting process of the software is performed in a certain order according to the quality, with asparagus of the best quality classes being collected in the first trays. Before the first use of the machine, all parameters are selected after the first sort has run through the machine. Then, the user can fit the thresholds accordingly. The asparagus is arranged on a conveyor belt that runs it through the recording section of the machine. Here, a camera takes three pictures per asparagus as seen in Figure 2.4. Small wheels on the conveyor belt rotate the asparagus in the meantime so that it can be photographed from several positions. In the best case, on each image a different side of the asparagus is recorded. The conveyor belt transports the spear further and it is sorted into a tray depending on the chosen quality class by the machine.
The accuracy of the sorting machine is described to be as good as 90% best case by the manufacturer, while the farm reported it to be best around 70%, with re-sorting being necessary by professional sorters. Especially categories like Blume or Hohle were considered to be inconsistent - if no better than chance - by both, manufacturer and farmer. Further information could not be given on the software of the machine. A meeting with a representative of the engineering company HMF that

manufactured the sorting machine was arranged (=www.hmf-hermeler.de). Unfortunately, the software itself was not available to HMF as it was produced by another specialized company. During the meeting, it was discovered that the quality of the camera might be highly relevant for the accuracy of the sorting machine. Luminance and resolution are decisive for the quality of the image. Thus, technical restrictions for the sorting process had to be expected, also for our idea of implementing techniques based on machine learning.

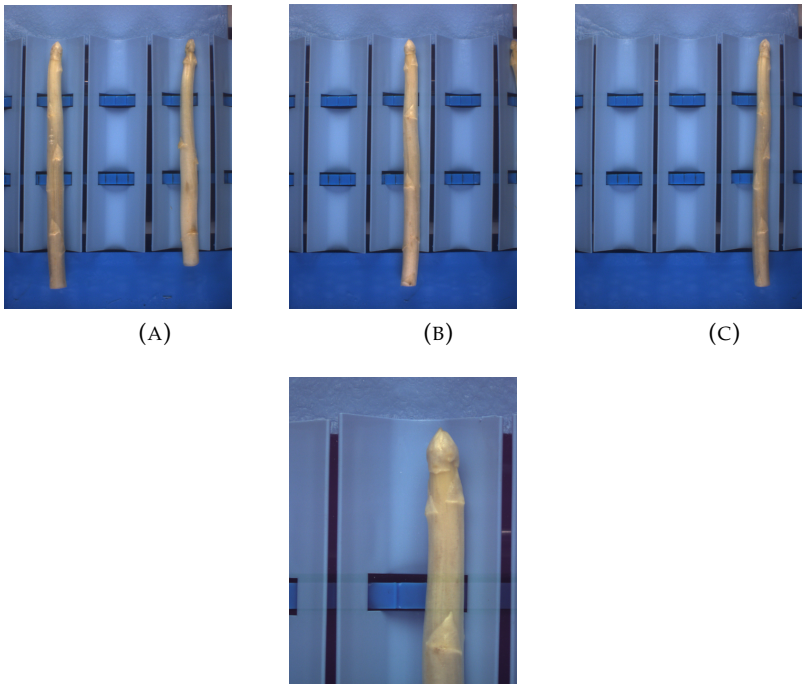
In the following it is described how the image data was collected. It was possible to save images with the Autoselect, however, the storage space on the machine was very limited. Whenever its C storage was full, the machine stopped and the sorting process was interrupted. This caused not only problems to the storing of images but also to the asparagus farm who had to rely on the machine for the sorting. Further, the selection of images to be saved was restricted to only 1000 images. Any number chosen above 1000 led to the machine saving every image until the storage was full and the machine stopped (i.e., when instructing to save 1001 images, the machine would ignore the number and save images up to its breaking point).
One solution to the image selection problem was the installation of the Teamviewer software on the machine. For this, the machine had first to be connected to the internet at the farm, where only restricted internet access was possible. After the Teamviewer installation, the process of image collection could be started remotely. This work was very ineffective and time-consuming because the process had to be restarted every 1000th picture. Further, there was still the problem of limited storage. The data could not be directly transmitted to another computer because the internet connection at the farm was not stable enough. Therefore, an external storage was connected to the machine. Further, automatic transfer of the images to the external harddrive was not possible but the images had to be transferred manually, again via Teamviewer. A solution to the problem was the installation of an automatic file moving service, for which the requirements are described below.
A program was needed that transfers the images to a new saving destination to resolve the problem of limited storage and manual transfer. It had to run in the background and should not disturb the workflow during the sorting process. The operating system on which the sorting machine runs is Windows. After some internet research on background processes and programs, the decision was made to use a service, that is, a system process running independently of any program. As Windows was used, the development was done with the DOTNET framework in the programming language C. The package provided is called Topshelf (https://github.com/Topshelf/Topshelf). Topshelf is a service hosting framework for building Windows services using .NET (https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview). Like this, it is possible to develop a console application in the development phase, compile it as a service, and install it later via the console. Previously, it was not possible to debug services. The function of the service is based on the FileSystemWatcher object (https://docs.microsoft.com/en-us/dotnet/api/system.io.filesystemwatcher?view=netframework-4.8) from the System.IO namespace. In the main program, a list of files in the source folder is kept. Files that are older than one hour are moved to the target folder on the external drive. The selected files are moved by a function that is called, when an event is triggered. The event is triggered by the FileSystemWatcher after subscribing to different flags. After a short time the service was adjusted. An hourly waiting time was then introduced via Teamviewer sessions because the program of the sorting

machine itself still accessed the image number indefinitely and therefore image collection had to be started every 1000th image.

Another issue concerning the storage space for collecting the image data was that the external harddrive ran full after 2 to 3 sorting days. The project members split in groups of 2 and exchanged the harddrive two times a week. The collected images were then transferred to storage capacities of the university with the support of the project's supervisors.
A last problem that had to be considered was that of unlabelled data. Before the project had started, the farm had not collected any labelled data and we could not rely on the labels that the machine attributed to each asparagus. A solution was to send the already sorted asparagus a second time through the machine. Like this it was assumed that we can not only receive labelled data but also acquire the parameters calculated by the machine. Unfortunately, it was not possible to extract the parameters from the machine. Another issue was posed by the fact that a second sorting is not good for the quality of the asparagus spears. Running them through the machine damaged some spears or led to the exclusion of certain spears from a good quality class into a worse quality class. Further, at least one project member had to be involved in the re-sorting and, thus, had to be present at the farm. The sessions of exchanging the external harddrive and collecting labelled image data were then combined.

An example image of the received data can be seen in Figure 2.4. There are three pictures per spear. The image resolution is around 10403 $\times$ 4 1376 pixel per image, with an RGB colour space.
All in all, the number of collected, unlabelled data is above XX images, thus, above XX asparagus spears. Of the labelled data, we collected 1005 images with the label I A Anna, 908 images for I A Bona, 513 images of I A Clara, 936 images of I A Krumme, 1514 images of I A Violett, 1051 images of II A, 1468 images of II B, 1169 images of Rost, 749 images of Dicke, 775 images of Hohle, 1717 images of Blume, 1157 images of Suppe, and at least 309 labelled as Bruch. The image number does not represent the number of different asparagus spears, as each spear is represented by three distinct images.

Most image data, that is XX image samples, was received from the asparagus farm "Gut Holsterfeld". Additionally, a few images could be recorded at another asparagus farm, "Querdel's Hof", in Emsbüren (https://www.querdel.de/). The farm sorts the asparagus with an updated version of the Autoselect machine at "Gut Holsterfeld", that is, it uses the same software but other hardware. The difference of both machines is purely in hardware, in particular the resolution of the camera was improved and a second camera was installed that focuses on the head region of the asparagus. At "Querdel's Hof", we collected 20616 images in total, 76 from the class 'normal', 152 from the class 'violet/flower', and 20388 unlabelled images. Of the data, an asparagus spear is represented by four images: three images show the same asparagus from different perspectives and a fourth image depicting solely the head region. An example image can be seen in Figure 2.5. No internet connection could be established to the farm, thus, no further images were collected. Moreover, the class labels of asparagus spears as well as the data format of the images from "Querdel's Hof" are different to the data from the farm "Gut Holsterfeld", due to the additional head image. Therefore a combination of both data sets was not convenient. A further reason for no more collected data at the second farm was that the objective behind the study project was primarily the implementation of new



(A) (B) (C)

(D)

FIGURE 2.5: Example pictures of the asparagus farm "Querdel's Hof". In picture (A), the asparagus spear is to the right, in (B), it is in the middle, and in (C) it is to the left. A fourth picture (D) is taken with a second camera with focus on the head

.

software regarding the machine type Autoselect at the farm "Gut Holsterfeld".

To summarize this section, of the XX images collected in total, mostly unlabelled data was collected from the sorting machine Autoselect at the asparagus farm Gut Holsterfeld. Of the labelled data, each class of asparagus is represented with at least XX images (corresponding to XX spears).

## 2.4 Literature research

In the classification of food products, there seems to be a large field of different possibilities to apply classical machine learning, and also artificial neural network approaches for classification tasks on image data (Bhargava and Bansal, 2018) (Brosnan and Sun, 2002).
Although, none of the papers that were found could be used as a blueprint to the asparagus classification project, some of them helped to get a better idea of how to proceed with the project, how to give structure to our preprocessing phase, or evaluate the machine learning methods that were already used on other food classification tasks (Mery, Pedreschi, and Soto, 2013) (Bhargava and Bansal, 2018). For example, some of the literature was concerned with the evaluation of food products but not with differentiating between as many classes as 13 (Diaz et al., 2004) (Kılıç et al., 2007). Often, the variance in the food products used was either too high (Zhang and Wu, 2012) or too low (Kılıç et al., 2007) (Al Ohali, 2011). One paper even evaluates the sorting of asparagus, however, only on a small dataset with three categories of green asparagus (Donis-González and Guyer, 2016). Further papers discovered on food classification were not detailed enough in their approaches

and, thus, not sharing any information needed for our task (Pedreschi, Mery, and Marique, 2016). Other papers were mainly interested in the implementation of a certain toolbox (Mery, Pedreschi, and Soto, 2013).
Besides the search for food related classification tasks, further research was done for evaluating the use of a semi-supervised learning approach. As the available data was only sparsely labelled an idea was to engage with approaches relying on semi-supervised learning (Olivier, Bernhard, and Alexander, 2006) (Zhu, 2005). Details about the regarding literature can be found in the respective chapter 4.3. There, a semi-supervised approach is used in the implementation of a variational aoutencoder. In regards to deep learning-based approaches, classical neural networks - such as AlexNet, VGG16/VGG19, GoogleNet, Capsule Networks, DenseNet, ResNet or Network in Network (NIN) - were assessed and presented for better understanding of the range of possible pre-trained networks and ideas for network structures (Krizhevsky, Sutskever, and Hinton, 2012b) (Simonyan and Zisserman, 2014) (Szegedy et al., 2015) (Sabour, Frosst, and Hinton, 2017) (Huang et al., 2017) (He et al., 2016b) (Lin, Chen, and Yan, 2013). Also, classical computer vision approaches were considered, like multiclass support-vector-machines (Prakash et al., 2012).
Another subject of extensive research posed the evaluation of our manual labelling of the unlabelled data (see chapter 3.4). For this, different methods were compared and an agreement measure was chosen (McHugh, 2012). Especially for later task distribution during the project year, literature research was mainly done in smaller groups of one or two and only discussed with the group if it was relevant to the team.

In conclusion, the time given for further research in a group was efficiently used. The design of a structured framework for collecting research literature could not be established. Except for the first few weeks into the project, there was no unified literature research during the project. Rather, every member consulted literature on their own for the specific tasks and problems everyone had to face. Like this, most of the papers that were used as inspirations or basis to certain tasks and approaches can be found throughout the report.

# Chapter 3

# The dataset

Before implementing any approach that allows to predict a label to an asparagus, the recorded image data has to be preprocessed and arranged in a sensible format making it accessible (or even simpler to work with) for any computer algorithm. It will be described in detail how the raw images were processed into a data set and how we obtained labels for the supervised learning approach. Further, the sorting criteria that were decided on and the evaluation of the label agreement will be summarized. In the end, the classical computer vision approaches that were used to detect certain features from the image data without a machine learning process will be outlined.

As a first step, the images were loaded and saved on the university internal store. The images were checked for any general errors and ordered in sets of 10000 images in the single folders on the Grid. This was followed by different preprocessing steps: the three images of one spear were gathered at one location and the background was removed together with any other unnecessary information in the images. Most of the collected data was not labelled such that a solution to the issue of how to attribute a fitting label to each spear was needed. This proved to be a labour-intensive task for the accumulated amount of data. The first attempt to classify the unlabelled data consisted of automatically extracting the single features in the image which are responsible for a spear's label. The idea was to use classical machine learning approaches to obtain the individual features. For example, one can define a threshold for the RGB values in the images to detect colour-based features like rust. As it was not possible to completely extract all features from the pictures (especially the detection of a flower posed great difficulties), it was decided to label the images by hand. For that purpose, first all implementable feature extraction functions were combined in an application to aid the hand-labelling process.
The hand-label application has a graphical user interface. With 'yes' and 'no' buttons, the user clicks through the different feature categories and confirms the presence or absence of a feature. Some of the automatic feature extraction scripts classified an image poorly or even incorrect on their own but could still be used as an assistance to the user of the app.
For manually labelling all images with the application, every member of the group had to look through a batch of asparagus images and note whether each feature is present in the image.
The manual label process bore its own difficulties as critical images led to indecisiveness in the feature detection and a variance in the overall labelling agreement. The sorting agreement was therefore derived and used as a measurement of the overall quality of the manual sorting process.
Finally, the data set was built. To have a unified version of all images and their respective labels, a reading function was created.

In this chapter, the different preparatory steps for the recorded data are described, including the creation of a data set which is usable for any machine learning or computer vision approach to analyse the image data. In 3.1 *Preprocessing*, the data is assessed and simplified for any further processing. The second section, 3.2 *Automatic feature extraction*, deals with the creation of feature scripts that were researched and implemented for an automatic recognition of single features. The results were combined in an application which is described in detail in 3.3 *Manual labelling app*. In 3.4 *Manual labelling*, the process of hand-labelling the images with the application is described, followed by a section analyzing the results and comparing the overall agreement of the labellers. The last section, 3.5 *The asparagus data set*, concludes with the creation of the final data set, used for the later training of the neuronal networks and other approaches to detect the label of a spear from its three images.

## 3.1 Preprocessing

An important and often underestimated step of any machine learning project is preprocessing. After collecting the data, it is not yet in the right format and needs to be altered for the particular purpose that one has in mind.

Inside the sorting machine we are working with, the asparagus pieces are transported on a conveyor belt with multiple compartments. Whenever an asparagus piece is in the center, a picture of the three center compartments is taken. That means, each asparagus can be found in three pictures, one in each of the three positions – left, center and right. In our case, the first step of the preprocessing was to combine the three perspectives of each asparagus piece. The image names can be used to combine the three relevant images and determine in which position the asparagus was captured. This way the images can be cut into three pieces and renamed in a way that makes it clear which images belong together. Each asparagus gets a unique identification number and the three perspectives are denoted with "a" for left, "b" for middle, and "c" for right. For example, the image 42_b.png is the center image of the asparagus piece with the identification number 42.
The second step was to remove the background to trim as much of the unnecessary information as possible. As the conveyor belt is blue, there is a high contrast to the bright asparagus pieces, aiding the background removal. First, the asparagus piece is masked using the hue of the HSV representation of each image. Second, very dark regions are marked as background by thresholding the value component. This is particularly important for the automatic feature extraction (cf. Chapter 4). Additionally, the asparagus was rotated upwards to reduce the variance in angle. The rotation angle is achieved by binarizing the asparagus piece image into foreground and background pixels, calculating the centerline as the mean pixel location along the vertical axis and fitting linear regression to the centerline pixels. The estimates for the angles are retrieved using arcus tangens and the images are rotated accordingly. By doing that, any distortions like reflections or single noisy pixels are set to zero and only the asparagus piece itself remains. The only faulty areas that can be left over with this approach are areas that are connected to the asparagus. The position of the asparagus within each cutted image was further improved by centering the fragments of the image around the center of mass. This way the asparagus is always in the middle. If the new cutting line falls out of the image boundaries, the missing area is filled with zeros.

It should be noted that, although it is an advantage to have several perspectives on the asparagus, as some features might only be visible from a certain side, it also

bears some problems. Namely, the lighting and reflection are slightly different, which can alter the colour values, and the images can be a little distorted. Hence, the asparagus might appear wider or smaller depending on the perspective. This makes it harder for classical computer vision algorithms to determine some features accurately.

Another data set was computed by converting the images to colour palette images. An appropriate palette and the respective mapping of RGB values to palette colours was determined using clustering in colour space. First, a set of RGB tuples is collected by adding pixel values for the depiction of 10.000 asparagus pieces. Second, the resulting list of RGB tuples is converted to an image and third, a palette is determined using a clustering algorithm that determines the position of cluster centers while maximizing the maximal coverage. The resulting cluster centers can be displayed as a list of RGB values and represent the colour pallette (https://github.com/python-pillow/Pillow/blob/55e5b7de6c41b0386660b0bee7784ac04f412f4b/src/libImaging/Quant.c, https://www.sciencedirect.com/science/article/pii/S1026309811002100). Each image of the downscaled data set was transformed to the palette representation. Visual inspection showed little quality loss such that it can be assumed that the relevant information for image classification is well preserved.

Several additional data sets were computed based on the data without background. This holds for downscaled versions as well as for a version that contains the asparagus heads only. To compute the latter, the images were padded to avoid sampling outside of the valid image boundaries and the uppermost foreground row was detected. Subsequently the center pixel was determined and the image was cropped such that this uppermost central pixel of the asparagus is the center of the uppermost row of the snippet. The resulting partial images of asparagus heads are rotated using the centerline regression approach described above. The approach has proven reliable and the resulting depictions were used to train a dedicated network for head related features (see 4.1.2).

## 3.2 Automatic feature extraction

Beside the main aim of this study project, to use machine learning algorithms to classify asparagus automatically, also classical computer vision methods were implemented for comparison. We used these methods to write algorithms that take the images as an input and return a prediction for the different features as an output. The goal was to predict the features reliable enough to deduce the corresponding class from them.

According to the producer of the used sorting machine, the softwares currently on the market use very similar methods, which gives us reason to believe that the classical methods might not be adequate to yield a better classification than the current status quo. Nonetheless, it is interesting to see, which features are harder to detect and which are easier.
The results vary greatly between the different feature extraction methods. While the functions to detect the width and length and whether the asparagus is violet or bended were good enough to be integrated into the hand label app to help us label, other features turned out to be more difficult.

In the following, the different automatic feature extraction methods will be described, alongside with the results that were achieved and future steps that could be taken to improve the results further. For each feature detection method, the images with removed background were used.

### 3.2.1 Length

The length detection uses a pixel-based approach, it counts the number of rows from the highest to the lowest pixel that is not a background pixel and therefore not zero. The asparagus was rotated upwards, as described in chapter 3.1, in order to improve the results, as the rows between the highest and the lowest pixel are counted and not the pixels themselves. This technique is a simplification, which does not represent curved asparagus very well, because it will have a shorter count than it would have if the pixels were counted along the asparagus piece. But in reality, there are not a lot of asparagus pieces close to the decision boundary between broken and a whole piece. Usually, the asparagus is picked a few centimeters longer than necessary and then cut to the desired length. The only asparagus shorter than that length are the ones that break while in the sorting machine. And if they break they generally break closer to the center of the asparagus rather than at the ends of it. Therefore, the difference in length detection does not matter for our classification.

All in all, the length detection yields good results that are very helpful for the label app. The next step would be to train a decision boundary that determines which number of pixels should be the threshold to differentiate between broken and not broken. At first, we tried to calculate this threshold by finding a conversion factor from pixel to millimeter, as we know the cut off in millimeters. But this approach appeared to be more difficult than anticipated, because the conversion factor varies in the different image positions. This problem only became apparent after the asparagus season had ended, for which reason we could not reproduce the camera calibrations in retrospective in order to take well-measured images, for example from a chessboard pattern. Accordingly, the threshold needs to be deduced from the data or learned with a machine learning approach.

### 3.2.2 Width

The width detection uses a very similar approach as the length detection as it takes the pixel count from the left-most to the right-most pixel in a certain row as a width measure. But in contrast to the length, the width was measured at several different rows from which the mean width was taken. For the label app we decided that three measurements are sufficient, because asparagus spears usually do not show drastic changes in width at several positions. Hence, if there are larger changes at all, three evenly distributed positions should capture those changes well.

The algorithm operates as follows: Firstly, the images are binarized into foreground and background, which means setting all pixels that are not zero, and therefore not background, to one. After that, the upper-most foreground pixel is detected and the length is calculated with the length detection function as described above. The length of the asparagus is used to divide it into even parts. This is done by determining a start pixel and dividing the remaining rows that contain foreground pixels by the number of positions one wants to measure at. This way several rows are selected in which the number of foreground pixels is counted. One can interpret each row as a cross-section of the asparagus, therefore the number of foreground

pixels is a direct measure for the width. Then, the mean of these counts is calculated and used as the final width value. As the head of the asparagus can be of varying form and does not represent the width of the whole asparagus well, it is excluded from the measure. This is done by selecting a start pixel below the head area instead of naively choosing the uppermost pixel. To be precise, the start pixel is chosen 200 pixels below the uppermost pixel in order to bypass the head area with certainty. As described in the length detection, also in the case of the width detection curved asparagus pieces might lead to slightly different outcomes than the true values, but again this difference is regarded as irrelevant in our case.

### 3.2.3 Rust

The rust detection finds all pixels that fall in the range of RGB values that correspond to the color brown. Those pixels are counted and normalized by the maximal number of possible pixels that could be rusty, namely the number of foreground pixels. But as it is impossible that the whole asparagus is rusty and hence that all the foreground pixels fall into the relevant range of RGB values, this normalization yields small numbers as results. To give an example, an output value of 0.13 is already considered moderately rusty. The lower and upper bound for the RGB values are set to [50,42,30] and [220,220,55], respectively. That means, all pixels that have a red value between 50 and 220, a green value between 42 and 220 and blue value between 30 and 55 are considered as rust.

Visual inspection shows that the rust detection algorithm works well to detect rusty areas and barely misses any rusty parts. But it is difficult to set a threshold for the number of pixels needed to be classified as rusty, because many pixels with a brown color that are distributed over the whole piece are not supposed to be classified as rust. Only clusters of brown pixels are a reliable indicator for rust. To make this intuition more clear, it could be the case that two asparagus have the same number of brown pixels, but in one case they are all connected building a cluster and in the other case they are evenly distributed over the whole asparagus piece. That would mean that, although both yield the same output value, only one of them should be considered as rusty. However, it should be mentioned that this is merely an artificial example to display one draw-back of the implementation. In reality, it is very unlikely that an asparagus has a large number of brown pixels that are not in fact rust.

Nevertheless, it remains unsolved to set a robust threshold that works well on the whole dataset. One problem that cannot be solved algorithmically, is dirt in the sorting machine. If the machine is not cleaned thoroughly and regularly, dirt can be falsely classified as rust as it often falls in the same color range. Another problem can be a change of lighting when taking the images. Both issues can be controlled for easily, but have to be communicated well to the farmers.

### 3.2.4 Violet

Especially when exposed to light asparagus pieces tend to change in colour. An initial shift from the desired white appearance to a slightly rose or violet tone can be observed first. This change in colour, that is considered a flaw according to german quality standards, typically affects the head of asparagus pieces. However in some cases a non-uniform distribution of more or less violet areas can be observed. Moreover, it has shown in practice that colour impression is highly subjective (Luo, 2000). This manifested in discussions of edge cases during labeling.

Effects of meta contrast potentially even affect the colour impression on an individual level (Reeves, 1981). The lack of a formal definition for violet asparagus pieces also poses challenges to approaches of measuring this feature.

In a simple approach to measure whether an asparagus piece is violet or not, colour hues are evaluated. More precisely, this strategy is based on evaluating histograms of colour hues that are calculated for foreground pixels of the asparagus images after converting them to the HSV colour space. Pale pixels are removed from the selection by thresholding based on the value component of the HSV representation. Finding the optimal threshold was proven difficult and corresponds to the above-mentioned question of what it means for an asparagus spear to be violet. A value of 0.3 was considered a good compromise. All three perspectives were taken into account to compute one histogram per asparagus piece. A score was calculated by summing up the number of pixels that lie within a violet colour range. A second threshold was used as the decision boundary for violet detection. The direct and intuitive feedback in the label app showed the relation between varying thresholds and the prediction. It has shown that lowering thresholds also means the feature extractor becomes more sensitive at the price of a reduced specificity. Best overall matches (accuracies) with the subjective perception were found for very low thresholds. In many cases, however, measurements based on this definition of "violet" did not match the attributed labels.

Hence, another sparse descriptor was derived from the input images. However, instead of thresholding pale values and calculating the histograms of colour hues this approach relies directly on the colours that are present in the depiction of an asparagus piece. As the 24 bit representations contain a vast amount of colour information in relation to the number of pixels it is, however, unfeasible to use these as an input. Instead the colour palette images can be used. Histograms of pallete images can serve as the basis to define the feature "violet" in a way that captures more of the initial colour information while being simple and understandable enough to allow for customizations by users of sorting algorithms or machines. As a consensus regarding such an explicit definition is arguably hard to achieve and somewhat arbitrary the descriptor was used to learn implicit definitions of the feature by examples (see XX).

It has been shown that explicit ways of directly measuring, in how far an asparagus piece is violet can be implemented but heavily depend on the definition of this feature. As colour perception is highly subjective across and even within subjects (Reeves, 1981), machine learning approaches that are trained on human labelled data appear to be promising. Using them can help generalizing a definition for the degree to which an asparagus piece is violet.

### 3.2.5 Curvature

Multiple curvature scores can easily be computed based on regression fits to the centerline of an asparagus piece. For example, the parameters of linear or polynomial regression can be interpreted as a description of the curvature of an asparagus spear. However, the question remains if a scalar descriptor that matches with the subjective definition can be derived.

Deriving sparse descriptions is based on a two stage approach. In the first stage, the centerline of an asparagus piece is computed because it is considered to be a good description of the curvature of asparagus pieces. Preprocessed images are

used as the input where the background is removed and replaced by black pixels. Each asparagus piece is approximately oriented vertically. This means that also for curved pieces the head relies within the top center of the image (see XX). The centerline is computed by binarizing the image into foreground and background and computing the mean of pixel locations along the vertical axis (i.e. for each row). The resulting binary representation shows a single pixel line. It serves as the input to the second stage of curvature estimation.

In the second stage, curves are fit to the pixel locations of the centerline. For the simplemost score, linear regression is employed and the sum of squared errors is thresholded and interpreted as a curvature score. This score is small for perfectly straight asparagus pieces and increasingly large for bent ones. As an S-shaped piece is arguably perceived as bent even when the overall deviations from the center line are small a second descriptor was computed as the ratio between the error of a linear fit and polynomial regression of degree three. Thresholding values and employing a voting scheme (e.g. at least one value indicates curvature) for the results for all three perspectives yields a rule to measure curvature. However, it has again proven difficult to set thresholds appropriately to reliably capture the visual impression. Hence, another sparse representation was calculated by fitting linear regression to each of six segments of each depiction of an asparagus piece. A multilayer perceptron (MLP) was trained on the resulting 18 angles per piece (see XX).

Calculating a score for curvature is fast and efficient. While the respective approach is suitable to define curvature it does not necessarily meet up with the subjective perception of asparagus curvature. Just like histograms of palette images, curvature scores are the results of feature engineering: The use of extensive domain knowledge to filter relevant features (Zheng and Casari, 2018). They can serve as an input to a machine learning approach that maps this sparse representation to the target categories (see XX).

### 3.2.6 Flower

The implementation of the flower detection function turned out to be difficult to realize. Several approaches have been tested, but none of them generated sufficiently good results. Two main notions were tried. On the one hand, the shape of the head was used as an indicator for a flower. The idea was that asparagus pieces with a flowery head exhibit a less closed head shape. In other words, the head looks less round and has no smooth outline, but shows fringes. On the other hand, the structure within the head was examined. Supposedly, asparagus with flowery heads exhibit more edges and lines in the head area. In both cases, it was challenging to find a way to discriminate between asparagus with and without flowery heads. One reason for that, is the poor resolution of the camera, that is installed in the sorting machine. With a pixel to millimeter ratio of around one to four, it is even difficult to detect flowers with the human eye. Likewise, the current software in the machine struggles greatly with the classification of this feature as well. There are newer versions of the machine available on the market that have an additional camera which solely takes images of the head of the asparagus piece. This way the inspection of the head improves considerably and the detection of flowery heads should be facilitated (see chapter 2.3).

## 3.3 The hand-label app: A GUI for labelling asparagus

Providing a sufficiently large data set that contains information about target categories is one of the major non-algorithmic challenges in the application of machine learning for classification tasks (Al-Rawi and Karatzas, 2018). In many cases, however, the respective labels are missing. Missing labels are especially problematic if traditional supervised learning methods such as feed forward CNNs are employed: if the variance in the input images is high, a large number of samples is required (Russakovsky et al., 2015). The options to reduce the variance and hence the need for a large labeled dataset are limited. One may rely on preprocessing to reduce the variance. In addition, strategies on the algorithmic domain such as the use of pretrained networks for transfer learning (https://www.learnopencv.com/keras-tutorial-transfer-learning-using-pre-trained-models/) or semi-supervised learning as well as manual feature engineering by means of traditional computer vision or machine learning (see 4.1.1) may help to retrieve sparse representations without the requirement for training on labelled data sets. As such these approaches promise to reduce the requirement for very large labelled data sets. This does not mean, however, that labelled samples are irrelevant. Without a sufficient number of labelled samples also these approaches fail. In addition, labelled data is essential for the evaluation of machine learning algorithms. Without labels no quality metrics such as accuracies, sensitivity and specificity can be calculated. If target labels are unknown it is hence inevitable to manually generate them.

Annotating labels manually is a common practice, however it requires plenty of effort. "Dataset annotation and/or labeling is a difficult, confusing and time consuming task; and even after labeling, it is difficult to assess a dataset for label correctness, whether manually or automatically" (Al-Rawi and Karatzas, 2018). Human performance may be acknowledged as the baseline or "gold standard" that image classifiers are evaluated by (Footnote: For example the performance of GoogLeNet is compared to human level performance using the ImageNet dataset (Russakovsky et al., 2015)). Hence in many scenarios data is labelled by human coders such that machine learning algorithms can be fitted on the training subset of the resulting hand labelled data sets and evaluated on a test subset. This holds especially for image classification tasks (Russakovsky et al., 2015). In the present case some features were considered to be reliably measurable by means of computer vision (e.g. the width or the length). For features such as a flowering asparagus head or the evaluation whether or not a piece is affected by the rust fungus this has proven to be difficult. Considering the amount of data that could potentially be labelled, a custom interface is required that allows for efficient attribution of labels: For this purpose an app with graphical user interface is developed that allows for efficient attribution of labels.

The hand label app comprises two user interfaces. A startup window that allows for a preview of asparagus images and the attributed labels (represented by Ui_Asparator) as well as the main labeling interface (Ui_LabelDialog). Using the labeling interface is possible only after the user selects the source folder of images and specifies or loads a file that contains the attributed labels. This ensures that the input and output file path are set. A dictionary that maps indices to images can be parsed from filenames and the minimum index is determined. As such the label dialog and the respective controller class always resides in a valid initial state. For labeling, the user answers questions (with yes or no) that are displayed alongside the images that depict each asparagus piece from three perspectives. To facilitate this process the arrow keys can be used. The result is saved and the next question

will automatically appear upon answering.
In addition, automatic feature detection can be selected for specific features. The result is displayed and saved to a file. The user is not asked to answer questions that target named features and attribute the respective label. This flexible approach was chosen as it was initially unclear and disputed in how far automatic feature extraction yields results that meet up with the individual perception. It also allowed to improve automatic feature extraction methods and to develop a direct intuition for the relation with the data. On top of that it has proven to be useful for debugging automatic feature extraction methods that failed for some images.

The development of the app was accompanied by three major challenges. First handling a large data set of several hundred gigabytes that is accessible in a network drive. Second, changing requirements that resulted from group decision processes with respect to automatic feature extraction as well as from unforeseen necessities in (parallel) preprocessing and made substantial changes of the initial architecture and the reimplementation of parts of the code necessary. The third challenge was the related question of the handling of internal states of the app. The latter may be further explained.

Internal states of the app were handled such that it is possible for the user to navigate into invalid states for which no images are available. The reason for this choice may be explained shortly. Note that preprocessing was done such that each asparagus piece has a unique Integer-ID and a specifier for perspectives a, b and c in it's filename. While generally IDs are in a continuous range from zero to n some indices are missing. As preprocessing jobs were scheduled to run in parallel and preprocessing failed for few corrupted files it has proven almost inevitable to end up with few missing indices although a dictionary of input filename and output filename was passed to each grid job. In addition, the large amount of data did not allow to save all files in a single directory. In summary this means that one could not simply iterate over asparagus IDs (represented by the state of a spin box in the user interface), determine the file path and display the related images. Instead parsing filenames from a slow network drive is necessary which requires limiting the number of selected pieces. As GUI elements such as spin boxes and keyboard controls allow for setting an integer, and it was a requirement that this integer relates to the asparagus ID, one ends up with the following situation. Either one prevents that the asparagus ID is set or incremented freely to a value that does not exist or one allows to navigate into an invalid state. It was decided that it is the most simple approach to allow the app to have an inconsistent state where the current ID is invalid because there is no respective data (Footnote: The earlier approach showed to have several implementation specific drawbacks. Note for example that upon entering multiple digits in an input field an event is triggered multiple times. Upon entering the value 10 for the asparaus ID one ends up with the value being set to 1 before being set to 10 where 1 relates to a potentially missing ID. This means the user could not freely enter IDs when setting them to specific values is impossible.). Hence, all cascades of methods of the app including preprocessing functions that require the respective images as an input were adjusted such that they can handle this case.

The app is implemented using the PyQt5 framework while coarsely following the model, view controller principle. No separate database and the respective model class is used. The data model is implicitly parsed from the filenames and administered in attributes of the view. The labels are administered as a Pandas DataFrame and serialized as a csv-file. Upon state change (i.e. index increment) images are

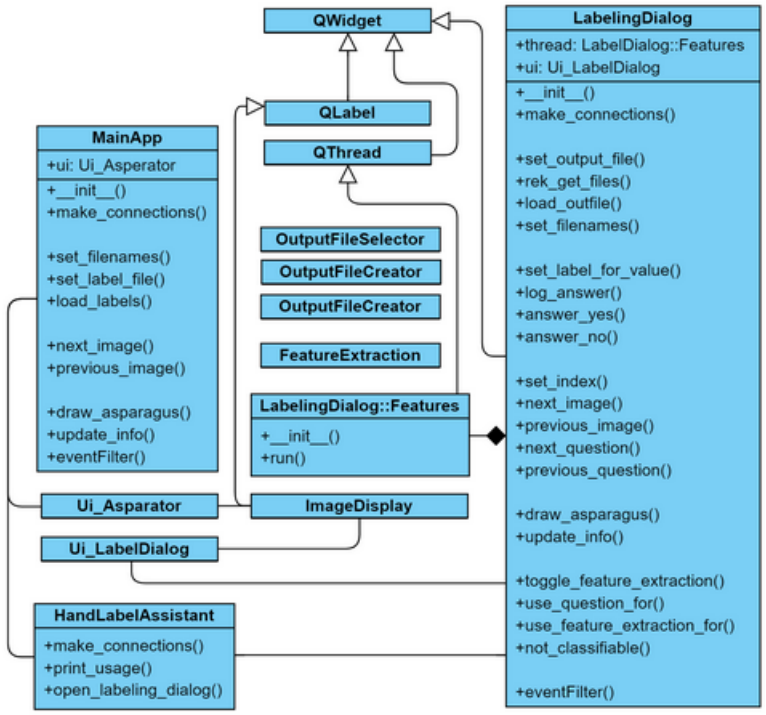



FIGURE 3.1: ??

loaded from the network drive that was mounted on the level of the operating system. The views are designed using QtDesigner. Four manually coded classes are essential for the architecture of the app: The class HandLabelAssistant in which the PyQt5 App is instantiated, the controllers MainApp and LabelingDialog as well as Features which is member class of the latter, of type QThread and represents the API to the automatic feature extraction methods in FeatureExtraction. Ui_Asparator, UiLabelDialog and classes for file dialogs represent the views. A class ImageDisplay is required to display images with the correct aspect ratio. The figure below shows the UML class diagram alongside methods and attributes that are considered relevant to understand the architecture of the app.
Developing a custom app for the labeling process required substantial time resources. However, it was found that existing solutions did not meet the specific requirements. Our custom hand label app allowed us to attribute labels for more than 10.000 labels in a manageable amount of time. Details of the labeling process are described in the next section.

## 3.4 Manual labelling

In this section of chapter 3, the process of manually labelling the data with the help of the hand-labelling app is laid out. It includes the sorting criteria which allocate each spear to a single asparagus class, the practice of manually labelling the preprocessed data for its features, the outcome of the labelling process, and the approach to measure the agreement of the manual labelling are described in more detail.

The images were classified by all members of the group. As none of the group participants are experts of labelling asparagus, a general guideline for the labelling had to be established which is elaborated in detail in the following subsections. The guideline is written in accordance with the owner of the asparagus farm of

"Gut Holsterfeld", Mr. Silvan Schulze-Weddige. He was consulted in all questions regarding the sorting of the asparagus.
General challenges in the manual sorting in front of a computer screen, including the respective image quality, and especially the variance in the agreement of the project members were expected from the start. As the task relies on the subjective view of single humans, opinions about the affiliation to one asparagus class can diverge in critical or vague cases. This is demonstrated in that the seasonal workers who usually sort the asparagus by hand do not agree on every case. Further, a classification session with Mr. Schulze-Weddige was held in which the project members learned to distinguish the different features that decide the class of an asparagus spear. During the session, again it emerged that some images containing critical spears - that is, where the features are not apparent - were hard to classify even by the expert.
To tackle the issue and to have an overview of the general agreement of the sorting between group members of the project, a measurement unit was researched and applied, namely the Kappa Agreement. It was applied on a smaller subset of presorted asparagus pieces in the beginning of the manual labelling and a second time at the end of the manual sorting process. The Kappa Agreement was used to assess the degree of accordance in sorting between the single members and monitor how the sorting agreement developed during the manual labelling process.

The first subsection, 3.4.1 *Sorting criteria*, expands on the guideline used for categorizing each spear of asparagus according to its features (into a single class). In the second subsection, 3.4.2 *Sorting outcome*, the process of the manual sorting is shortly described and the achieved results are presented. The ensuing third subsection, 3.4.3 *Agreement Measures*, and fourth subsection, 3.4.4 *Reliability*, focuses on the Kappa Agreement, which we chose as an evaluation measurement for the accord of our sorting, and the results of applying it are discussed.

### 3.4.1 Sorting criteria

The class of an asparagus spear is decided by several factors, ranging from its shape to its colour. Put together, these single features give us the label for the spear. As it was decided to label the images for their features, the features were checked by the labellers. The images were displayed with the hand-labelling app and the features are divided as follows: 'fractured', 'hollow', 'flower', 'rusty body', 'rusty head', 'bent', 'violet', 'very thick', 'medium thick', 'thick', 'thin', 'very thin'. Further, images that could not be sorted thoroughly were sorted as 'not classifiable' (e.g. when the spear is cut off the image).
To some extent, the manual feature categorization was influenced by the price that each category of asparagus earns. High quality pieces (i.e. 1A Anna) should be thoroughly examined and sorted more conservatively.
In the following text, the different criteria for manually labelling the data images for their corresponding features are described.

**Fractured**
An asparagus spear includes the feature 'fractured' when it is broken, or if it does in any other way not fulfill the required length of 210 mm. First, the feature was labelled manually but the procedure was abandoned shortly after because it can be derived more precisely from the computer-vision based feature extraction 'length', which is automatically calculated within the labelling app. It is then discerned between a fractured piece with a head and a fractured piece without a head. Whenever a spear was fractured but with the head part still attached, it was sorted like

a usual, intact asparagus. However, if the head part of the spear was damaged or missing completely, no other features were chosen but the asparagus was labelled as 'not classifiable'.

**Hollow**
The feature 'hollow' indicates that the spear is hollow inside. This might be expressed by a bulgy center and a line running vertically along the spear's body. Another, more distinct indicator is when the piece looks like two spears fused together, forming a single asparagus. A hollow asparagus can be confused with a very thick asparagus.
The feature can be easily checked when you have physical access to the asparagus. If the asparagus is actually hollow, it will have a hole at its bottom which you can see when turning the spear around. Unfortunately, this cannot be done when only looking at images from the side. The feature hollow sometimes even occurs without showing a clear line or obvious bulge at its center. Therefore, there is a risk of wrong classification.

**Flower**
The asparagus piece is sorted as 'flower' when the head part forms a recognizable flower. For a blooming asparagus, a jagged pattern (slightly resembling a crown) can be seen at its head region. Also, the tip of the head part can be an indicator if it resembles a pointed cap and the petals are visible. When a spear is in full bloom, it is clearly observable. However, the distinction between a spear with clear-cut but closed petals and a spear which just started to develop a flower can be quite difficult.
The feature was sorted less strict in uncertain cases where the flower is not clearly distinguishable. One argument is that the flower does not develop further after the sorting process (e.g., as it happens with the feature 'violet'). Another reason for a less conservative approach regarding the feature 'flower' was to lessen any agreement errors in between the manual sorters. It was decided to sort a spear without a rather clear flower as not having the feature 'flower'.

**Rusty body**
If a spear has rust on its body, it is visible as a dark brown colour. It often starts at the tips of the leaves or at the bottom part. In severe cases it spreads over the whole asparagus piece. The colour is not light brown but of a dark shade. The colour is not to be confused with bruises, pressure marks, or a slightly yellow complexion (which can occur in a mature asparagus) which are of no further concern to us. The feature was sorted more strictly to facilitate the decision process during the manual sorting. Therefore, 'rust' was set to be present even when only the tip of a leaf showed a dark spot. Other brownish bruises were not classified as rust.
We decided to split the feature 'rust' into the sub-features 'rusty body' and 'rusty head'. In the case of rust being only on the body, it might still be removable by peeling. If there is rust on, or very close to the head, however, it is not removable.

**Rusty head**
If there is a dark spot on or close to the head region recognizable as rust, it is captured by this feature. The head part is usually distinct in shape and colour from the body part of the asparagus.
In principle, the same guidelines applying to 'rusty body' can be transferred here, with an explicit focus on the top part (until around 1 cm below the collar) of the asparagus. Rust at the head can be confused with shadow caused by the petals and

the luminance of the sorting machine. Also, it might be that the head is actually violet and not rusty. As rust at the top part of the spear cannot be removed without damaging the head, it is more decisive for the later categorization into a price class, if the head has rust.

**Bent**
A piece is categorized as bent, if the shape of the asparagus is curved and not straight. Further, a spear counts as bent whenever it changes the growing direction - that is, it resembles an S curve. If it is only slightly round but can otherwise be thought of as straight and fitting next to other straight pieces without standing out, it is sorted as not being bent. If the spear looks close to the same on all three pictures, it might indicate that it is heavily bent and therefore cannot be turned on the machine's conveyor belt.
Like the feature 'flower', though, the feature bent has a broader range of shapes where the piece is not obviously deformed but also not completely straight. In an indecisive case, it was sorted less conservative. exception holds for S-shaped pieces, which always count as bent.

**Violet**
The feature 'violet 'indicates whether an asparagus piece is of violet colour. The shift of colour from white to violet occurs most often around the head region - either at the tip of the head or just below the collar of the head region. However, the piece can also be violet on the whole body.
Here, it is crucial to sort thoroughly because the spear will darken further after the sorting. Thus, even a slightly pink spear was sorted as being violet.

**Thickness and length**
The thickness and the length were features we did not consider recognizable by view alone (except for extreme cases). Therefore, both features were measured automatically with scripts written and implemented into the manual sorting app. The division into different categories of thickness can be inferred by the overall thickness of the spear.

**Not classifiable**
Whenever an image was damaged, the spear was unrecognizable, the head part of the spear was severed, two separate spears were present on one picture, the spear was cut off by the image, or other unusual circumstances on the image occured, it fell into the category of not being classifiable. The image was not further checked for its features, i.e. as bent, violet, etc., but only marked as being not classifiable.

### 3.4.2 Sorting outcome

In this section, the process and the results of the manual sorting are presented. During the overall sorting period no major problems occurred that led to any breaks or even the abortion of the labelling process.

Whenever an image was manually sorted in our hand-label application, the chosen labels were saved in a csv-file. As csv files are plain text files, all stored values are strings arranged in a table. The first line of the table serves as the heading, attributing each entry in the first column to a feature. As seen in the figure 3.2, the features are noted in the first column, with the first entry being the image ID. Every feature is separated by a semicolon and can be of value 0.0, 1.0, or empty. Hence, whenever a feature was present in an image, the value was set to 1.0, while,

if the feature was not present, it was set to 0.0. For the "not classifiable" labelled images, no value was given to any other feature. In such cases, all other manually selectable labels are left empty. The image path to every of the three images (a, b, and c) for one spear was also saved in the label file in a separate column. After the labelling process, the single csv-files with the labels were merged into one large combined_new.csv file which was later used for the classification of the data with the different computer-vision approaches. It can be found in the study project's Github repository under XX.

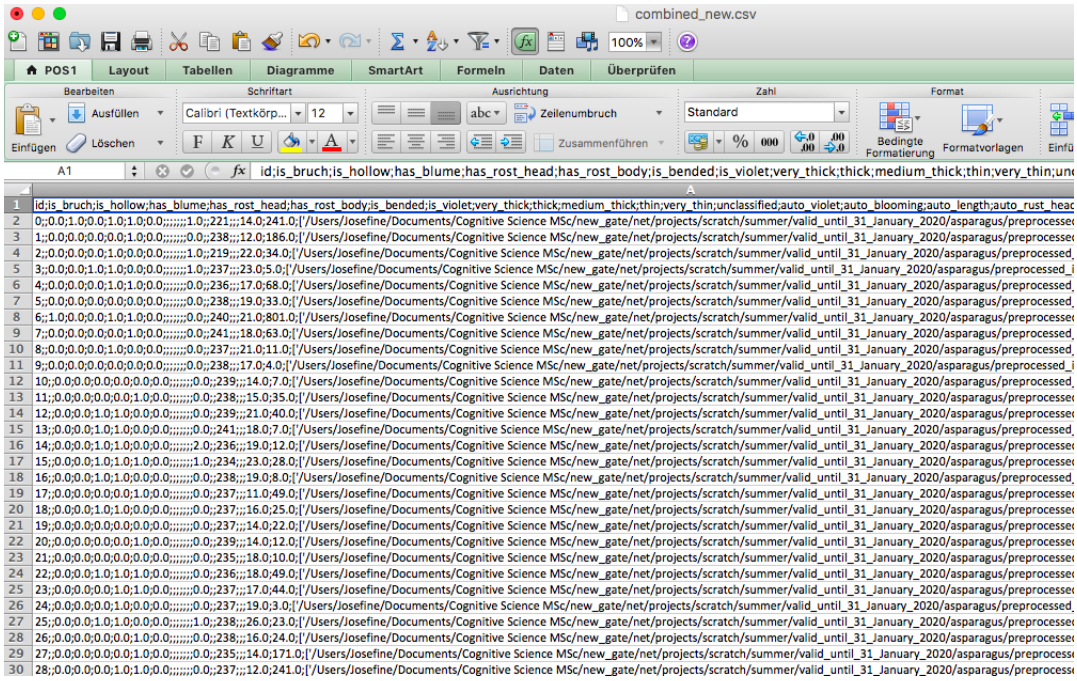



FIGURE 3.2: The feature labels extracted by the manual sorting process were saved in a csv file. This image shows the beginning of combined_new.csv in which all label files were later combined to one file.

The manual sorting lasted over the period of November 2019 to January 2020. A session usually consisted of 500 images, with an asparagus spear being viewed in three positions from the same perspective. A session of sorting 500 images took around two to four hours. One minor factor sometimes influencing the time spent for sorting were difficulties with the external connection to the IKW store, as all images are stored on the university servers and it was sometimes worked from home. Another factor was the large number of pictures, with some spears obviously incorporating certain features, while others needed more careful examination. The average time spent for sorting one spear (i.e. noting all features present) was around 27 - 48 seconds.

All in all, 13319 triples of images were labelled for their features. There is a large variance in the presence of the features in the data. Some features were occurring more often than others. Of the acquired 13319 images, the individual features are represented as follows: 3.5% fractured asparagus pieces, 3.3% hollow, 12.9% flower, 14.7% rusty head, 45.5% rusty body, 40% bent, 7.9% violet, and 2.1% not classifiable images. Further categories that were not manually assessed but calculated from the automatic thickness detection included 4% very thick (i.e., above 26

mm), 29.1% thick (20 - 26 mm), 18,7% medium (18 - 20 mm), 17.9% thin (16 - 18 mm), 30.3% very thin (below 16 mm). A further feature calculated automatically was the length of each spear. All images that were not classifiable are excluded from the calculation of the thickness in the stated numbers.
As can be seen, many features are only sparsely present in the data. This poses an imbalance that is relevant for later classification tasks and the usage of the data set. Further, every member of the group participated in the sorting but not everybody sorted the same number of images. Due to this circumstance, the sorting bias of certain members is more present in the data than of others.
The manual sorting was stopped when the amount of classified data exceeded 13000 samples. That is, the less present features (hollow, violet, and flower) were in advance assumed to have a presence of around 8%. In order to have at least 1000 images with such a feature present, a threshold of 13000 labelled samples was set in advance. The manual sorting was put to a halt when this threshold was reached.

As different people were sorting the asparagus and because the decision boundary for sorting a spear is flexible/subjective to the human eye, a measurement was needed that explores how well the members agreed in the labelling process. Thus, at the beginning and at the end of the manual labelling, images were taken and scrutinized for their features. This was done to train the labellers but also to have a test set for the agreement measure. Every category was therefore sorted by at least two labellers separately and their agreement was compared. More details about the method that was chosen to assess the group's sorting accuracy can be found in the following section 3.4.4. *Reliability*.

### 3.4.3 Agreement Measures

When different annotators label data, it is indispensable to verify the degree of agreement among raters. In order to judge how consistently the data is labeled, several statistical methods (inter-rater-reliability) can be applied.

For the current purpose, different agreement measures, all implemented by scikit learn, were used. The first was Cohen's Kappa. It was chosen, as it is seen as a more robust measure than a simple agreement percentage, such as a measure of true positive and true negatives, which was traditionally used for those cases (Cohen, 1960). It is more robust, as the rate of agreement occurring by chance is included in the calculation. This method is applicable to compare the rating of two raters on a classification problem. This measure of agreement always lies between -1.0 and 1.0, inclusively. The higher the Kappa value, the higher the agreement. Values around 0 indicate no agreement and negative values indicate negative agreement, so to say systematic disagreement. Values between 0.41-0.6 are seen as moderate agreement, 0.61-0.8 as substantial agreement, and everything above as almost perfect agreement. Everything below 0.4 is interpreted as not acceptable (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/).

Another statistical method used to measure agreement is the F1 score. The F1 Score is used for the evaluation of binary classification. The F1 score relies both on the precision, as well as the recall of a test's accuracy. A F1 score value lies between 0 and 1, the higher the F1 score, the higher the agreement.

Lastly, we looked at the accuracy measure. For a normalized accuracy score, the values lie between 0 and 1, and the best performance is 1. This measure returns

the fraction of correctly classified samples. It is a less robust measure than Cohen's Kappa score.

### 3.4.4 Reliability

In order to evaluate the degree of agreement of our data, we made agreement measures at two points in time (Footnote: for our API documentation see: https://asparagus.readthedocs.io/en/latest/api/measure_agreement.html). The first time, six different annotators labelled images out of each asparagus group (13 class labels). We ensured that always two different annotators labelled the same set of images. Results were not as good as hoped for. The Kappa scores varied strongly between groups and features from -0.03 to 0.76, while the accuracy scores ranged from 0.49 to 1. It seemed untypical to us, that the agreement scores were so low, even though the raters give the same label to many of the asparagus pieces. This is a known problem (Powers, 2012) (Sim and Wright, 2005) (Feinstein and Cicchetti, 1990)(https://www.sheffield.ac.uk/polopoly_fs/1.404095!/file/RSS_Poster_Laura_Flight_Final.pdf).

One reason for our results could be that we compared the agreement group-wise. This means that we already knew in advance that for certain features, we have almost only 1s or almost only 0s for all images of one group. For Kappa scores, if the distribution of 0s and 1s is not balanced, disagreement of the underrepresented value is punished more heavily (Footnote: see also: https://stats.stackexchange.com/questions/47973/strange-values-of-cohens-kappa). Therefore, we decided to repeat our agreement measure and do it this time feature-wise on non-labeled images, so that the annotators could not anticipate a specific group label. In order to better understand the reliability of our data, we also decided to look at the accuracy score and the F1 score, additionally. Before we did so, we labeled another 50 images all together, clarified classification boundaries again and discussed unclear images.

The second time, 50 images were labeled by 4 annotators. The agreements were measured annotator-pair-wise, and then averaged. The results in the Cohen's Kappa score vary between and within features, and also between annotator pairs. However, the results are much better than the first time. The highest aggregated kappa score over all annotator pairs is reached for the feature flower (0.79), then hollow (0.79), violet (0.76), rusty head (0.72), bent (0.55) and lastly for rusty body (0.47). For the features flower, rusty head and violet, the interquartile range (IQR) is quite small, whereas the IQR for hollow, bent and rusty body is much larger (see Figure 3.3 and XX - both figures).

The agreement scores accuracy and F1 yielded very similar results. Results were slightly better than the Kappa scores, in total and for each feature. The highest accuracy score is reached for the feature hollow (0.96), then flower (0.93), then violet (0.92), then rusty head (0.86), then bent (0.75) and then rusty body (0.74). The order is the same for the F1 scores. The median F1 scores lie between (0.71 and 0.97).

All in all, one can therefore say that the agreement scores indicate moderate up to substantial agreement. Agreement is highest for the features flower, violet and rusty head.

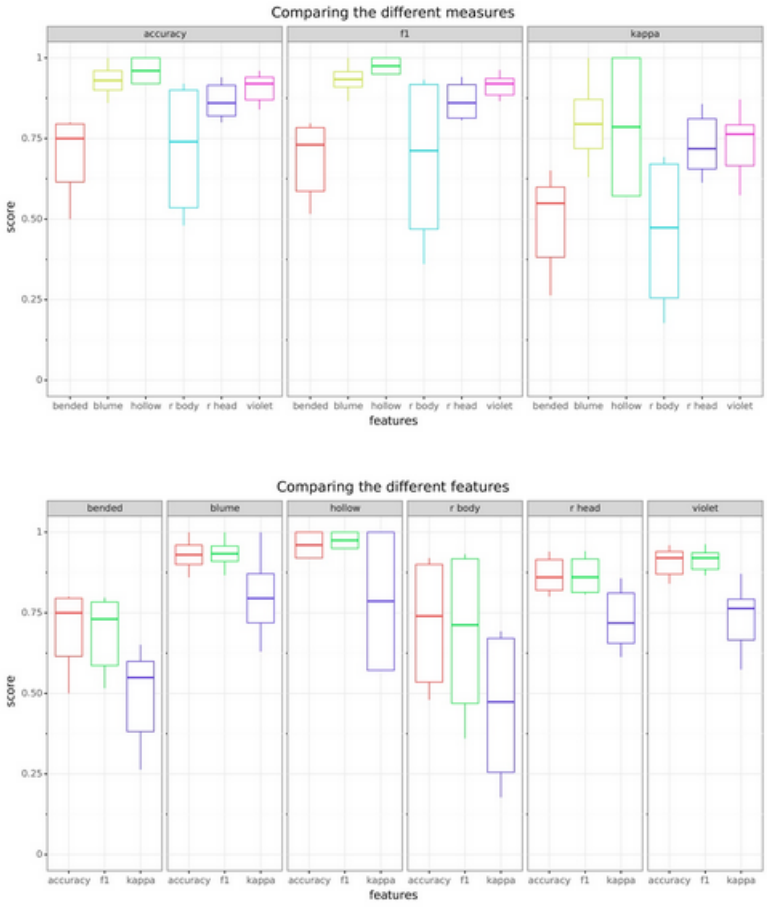



FIGURE 3.3: The upper figure shows the agreement measures accuracy, F1 and Cohen's Kappa, separately for each manually labelled feature. Shown are the box-plots, so the middle line indicates the median, the box indicates the IQR. All scores are aggregated scores over all annotator pairs. The lower figure shows each feature separately. For each feature, the corresponding accuracy, F1 and Cohen's Kappa score is given. Shown are the box-plots, so the middle line indicates the median, the box indicates the IQR. All scores are aggregated scores over all annotator pairs.

## 3.5 The asparagus dataset

It was described how the data was collected and stored on the IKW storage. After saving, the images were pre-processed and saved in a selected structure. Additionally a CSV file with labels for classifications was created. In this subchapter it is described how the images were processed to be able to use them with models in the future.After giving a theoretical overview of the different possibilities, an assessment follows of what our ideal would have been in the project best practice. In addition, a description is given which of the existing methods we have used in practice. These methods are compared and followed by a recommendation for further work with the data set.
One question arising in regards to modeling a dataset was: what is the best way to provide a model with the image input and corresponding information? There is more than one answer to this question, which makes it more difficult than initially expected. Building the input pipeline in a machine learning project is always long and painful and can take as much time as building the models themselves. First,

it is described which common methods are used to prepare the data. We introduce the methods of simply using raw data, saving a NumpayArrays/Tenors or a TFRecord file and the tf.data API and the tf.data.data set API. The most straigthforward variant is to build the model first, then read in the data ,and load the samples one after the other. APIs such as OpenCV, or Imeagio help with this. Here,the possibilities of Python are an advantage. By far the shortest time needed to implement is to load the images from the network memory, our folder structure, and then make them available to the network. However, the runtime is slow, and further processes (described below) are not efficient. Additionally, there may be an overflow and large amounts of data do not fit into the memory.

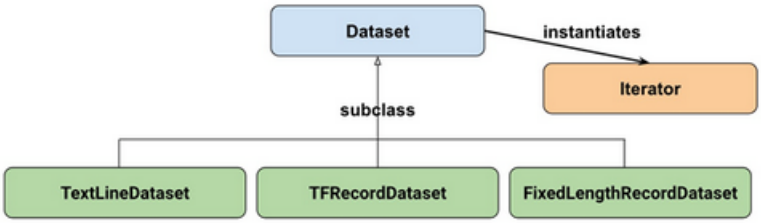



FIGURE 3.4: ??

```
# Use interleave() and prefetch() to read many files concurrently.
files = Dataset.list_files("*.tfrecord")
dataset = files.interleave(lambda x: TFRecordDataset(x).prefetch(100),
                           cycle_length=8)

# Use num_parallel_calls to parallelize map().
dataset = dataset.map(lambda record: tf.parse_single_example(record, …),
                      num_parallel_calls=64)

dataset = dataset.shuffle(10000)
dataset = dataset.repeat(100)
dataset = dataset.batch(128)

# Use prefetch() to overlap the producer and consumer.
dataset = dataset.prefetch(1)
```

FIGURE 3.5: ??

Speichern eines Numpy Arrays/Tensors: SOPHIA
Additionally, we created a single numpy data set for the labelled images that could be easily stored and loaded for training. Firstly, all the identification numbers from the label files were selected and the corresponding images were found and copied to a separate folder. This folder contains all the images for which we have labels. Secondly, the images were grouped by their identification number, which means the three perspectives of each asparagus spear were combined. We decided to downsample the images for creating this data set to facilitate the training process and reduce memory consumption. Initially, each image was downsampled by a factor of six, that means every 6th pixel was used in the reduced image, but this factor can be easily changed and a new data set will be created for the individual needs of each approach. After that, they were transformed to a numpy array and concatenated to a single file. The images were either concatenated horizontally, so that in the resulting image the three perspectives appear to lay next to each other, or vertically, so that the images are stacked one after another. Finally, all the concatenated arrays were combined to a single four dimensional data set. The first dimension depicts the number of labelled asparagus spears, the second and third

dimension represent the height and the width of the images, respectively. Further, the fourth dimension represents the depth, which is three for the horizontally concatenated images, namely the RGB values, and nine for the vertically stacked images.

In the following, Tensorflow's own binary storage format TFRecord is introduced. This approach facilitates the mix and match of data sets and network architectures. Especially because we have huge amounts of data XXXXXX TB, it will have a significant impact on our import pipeline and, therefore, on the total training time. The file format is optimized for images and text data sets. These are stored in tuples which always consist of file and label. The binary files from 100MB to 200MB in size not only take up less space in memory, but can also be read more efficiently. This makes even more difference in our case because the data is stored in the network and not on an SSD hard disk on the PC. The serialized file format allows the data to be streamed efficiently through the network. Another advantage is that the file is transportable over several systems, regardless of the model you want to train with it.

Working with TFRecord also simplifies the next steps of .... For example, the division into train set, test set and validation set in a folder structure is no longer necessary and also the mixing of data is no longer needed. These functions are possible with the data format dynamically in any position.

how to build a TFRecord? (In brief)
Any data in TFRecord has to be stored as either a list of bytes, a list of float, or a list of int64 only. Each of these created data list entities has to be wrapped by a Feature class. Next, each of the features is stored in a key value pair with the key corresponding to the title being allocated to each feature. These titles are going to be used later when extracting the data from TFRecord. The dictionary created is passed as input to Features class. Lastly, the features object is passed as input to Example class. Then this example class object is appended into the TFRecord.

After the TFRecord files were built, it was time to read the data efficiently and create the dataset. With Tensorflow release, 1.4 tf.data was introduced. With the tf.data API it became possible to create complex input pipelines from simple, reusable parts.

Tf.data even makes it possible to edit large data sets. The preferred pipeline for our asparagus project is to apply complex transformations to the images and combine them into stacks for training, testing and validating in arbitrary ratios.

These "like lazy lists" are not really a new idea. The principle can be found in most mainstream languages like C's LINQ or Java 8's streams.

Tf.data provides two interfaces. Firstly, a dataset can be created from a source in several ways. Secondly, a dataset can be changed, e.g. by using different labels or just by transformations like mapping, repeating, batching, or many others. These dozens of transformations can be combined, where a classical order is described in the following.

Order of the operations
To summarize, one good order for the different transformations is:

1. create the dataset
2. shuffle (with a big enough buffer size) 3, repeat
3. map with the actual work (preprocessing, augmentation...) using multiple parallel calls
4. batch
5. prefetch

Besides the described functional transformations of the input pipeline under tf.data.dataset, there is an iterator which gives sequential access to the elements in the dataset. The iterator stays at the current position and gives you the possibility to call the next element as a tuple of testors. With initializable iterators, you have the possibility to run through the dataset several times. In addition, you can pass different parameters to start the call. This is especially handy when searching for parameters.

In summary there are two advantages. On the one hand, there is the possibility to build a dataset with different data sources. On the other hand, there are many functional transformations and an illerator with sequential access to the data.

Derek Murray recapitulates: "If you compare a tf.data pipeline to the equivalent queue-based pipeline, we use a similar structure of queues and threads, but importantly there are no Python queue runner threads on the critical path, and so the tf.data pipeline isn't constrained by the Global Interpreter Lock and it scales to much higher throughputs."
(https://docs.google.com/presentation/d/16kHNtQslt-yuJ3w8GIx-eEH6t_AvFeQOchqGRFpAD7U/edit#slide=id.g254d08e080_0_10)

At the beginning of the development of the dataset, there was the misunderstanding that one of the possibilities was to create a single object which represents the dataset. The most promising one seemed to add directly to the TFDS.
https://www.tensorflow.org/datasets/beam_datasets

This would enable the user to get the dataset directly from tensorflow api. For this, we would have needed to publish the data, which would have been no problem. The documentation was not that far advanced at that time, so we spent a lot of time on it. Especially the large amount of data was a problem. Questions like: "How do we deal with the fact that only a part is labelled" or "How should we enter the labels: each as a feature, in a list, or as several features?". It would have been faster and more helpful for the development of the networks if we had continued to search and integrated the TFRecord files with the tf.data api in our pipeline. So we needed a lot of time and effort that was ultimately lost.

# Chapter 4

# Classification

Given the structure of our dataset, namely image data with a sub dataset with corresponding class labels, and a sub dataset with corresponding feature labels, different machine learning and computer vision methods were chosen, to tackle the problem of image classification.

Image classification refers to the method of identifying to which category an image belongs to, according to its visual information. Classification problems can be divided into three different types: binary, multi-class and multi-label. Whereas a binary classification only distinguishes between two different classes and therefore classifies an image into one of the two classes, a multi-class classification distinguishes between multiple exclusive classes. A multi-label classification also works for multiple classes. However in the latter, a single image can belong to none, one, several or all of the classes. Those classes can be seen as a feature vector for each image. Each feature can be present or not, independently of the other features. ( vielleicht hier zitieren, seite muss noch rausgesucht werden (Har-Peled, Roth, and Zimak, 2003))

Each of those classification methods comes with certain advantages and implications. There exist many classification methods which have been developed for binary classification problems, but less methods are suited for multi-label or multi-class classification. Therefore, the latter often work as a combination of binary classifiers. Moreover, multi-class and multi-label classification have the difficulty of sparser labels.

Binary classification can be used to decide whether a certain feature is present at one certain asparagus piece, or not. This is helpful for a first inspection of the data, but does not enable a full classification of one image into one of 13 classes, which are currently sorted at the Spargelhof Gut Hosterfeld. Multi-class classification solves this problem. It is fairly easy to apply this classification type on the pre labelled images, but increasingly difficult for the semi-supervised and unsupervised approaches. While it enables a clear identification of class belonging, it does not enable to train variability within classes. As the class id results from a combination of the presence of certain features, and the absence of others, it is therefore also reasonable to go for a multi-label classification approach.

Moreover, there are different methods on how to approach image classification. Those can be divided into three main groups: supervised learning, semi supervised learning and unsupervised learning. In addition to classical computer vision-based approaches, our study group investigated several neural network approaches.

During our group work, algorithms of all three different classification types (binary, multiclass and multilabel) as well as of all three learning types (supervised,

semi-supervised and unsupervised) were applied for different working steps and different purposes.

In the long run, an integrated model was aimed which predicts all features of a single asparagus piece, and from which an additional class can be inferred. However, as intermediate steps towards that goal, the focus was to optimize models on identifying the presence of single features. Besides that, we only investigated a few multi-label classification tasks.

The following chapter aims to give a general background of the different approaches chosen for our image classification problem, as well as a detailed overview of the concrete implementations of the models and the mechanisms of their hyperparameters. All algorithms were implemented using Python.

## 4.1 Supervised learning

In machine learning, there are different approaches for an application to be trained on a set of data (Géron, 2019) (Bishop, 2006). Depending on the level of supervision that the system receives during the training phase, the learning process is grouped into one of four major categories (Géron, 2019). One of these categories is supervised learning. For supervised learning approaches, the training data includes not only the input but also its corresponding target labels, which is why only labelled data is used. The objective is to find a mapping between target (x) and label (y) when a set of both (x,y) is provided as training data to the application (Olivier, Bernhard, and Alexander, 2006). An advantage of supervised learning is that the problem is well defined and the model can be evaluated in respect to its performance on a labelled data set (Daumé III, 2012) (Olivier, Bernhard, and Alexander, 2006). In other words, the labels can be used as a direct basis for the model optimizing function during training.
Supervised learning spans over a large set of different methods, from decision trees and random forests, to support vector machines (SVM), to neural networks (Caruana and Niculescu-Mizil, 2006) (Géron, 2019).
A classic task for supervised learning systems is the classification of received data and mapping it to one of a finite number of categories (Bishop, 2006). The disadvantage of supervised learning is the effort of receiving enough labelled data. It can be challenging to obtain fully labelled data because labelling experts are needed to classify the data which is usually a time consuming and expensive task (Zhu, 2005) (Figueroa et al., 2012).

In the following subchapters, different supervised learning methods were chosen to solve the classification task either (a) by using the labelled data that we collected by running pre-sorted samples a second time through the sorting machine, or (b) by using the data that was manually labelled for features as described in chapter 3. In 4.1.1 *Prediction based on feature engineering*, an approach using a multi-layer perceptron (MLP) is described for feature classification. In the second section 4.1.2 *A dedicated network for head-related features*, a convolutional neural network is used to train solely on the head image data for the features flower and rusty head. The third section, 4.1.3 *Single-label classification*, is concerned with labelling the input images with a convolutional neural network (CNN) in a binary setup for their designated features. In 4.1.4 *Multi-label classification*, a neural network is trained on the data to label it not only for one feature but all features at the same time. In the last

section 4.1.5 *From feature to label*, a random forest approach is described to map the features of the image data to their class label.

### 4.1.1 Prediction based on feature engineering

Besides approaches that directly use images as an input one may use high level feature engineering to retrieve sparse representations that contain relevant information in a condensed form and apply classical machine learning classifiers such as MLPs to predict labels (Zheng and Casari, 2018). These classifiers are comparatively simple, fast to train and only few network hyperparameters have to be defined. One may argue that this is one of the major benefits as compared to networks of higher complexity (e.g. convolutional deep learning nets) that are typically characterized by a more complex structure and hence a larger hyperparameter space. As a consequence, finding suitable parameters for MLPs (i.e. the number of hidden layers and neurons per layer) is comparatively easy(Footnote: The challenge of finding appropriate network parameters is well known in the deep learning community: "Designing and training a network using backprop requires making many seemingly arbitrary choices [...] These choices can be critical, yet there is no foolproof recipe for deciding them because they are largely problem and data dependent" (LeCun et al., 2012). The requirement to specify hyperparameters is a disadvantage of neural networks (including MLPs) as compared to parameter free methods (ScikitLearn https://scikit-learn.org/stable/modules/neuralnetworkssupervised.html). Due to the combinatorial explosion the above mentioned challenge of finding suitable is harder for more complex networks as more options must be considered.): An extensive search in hyperparameter space is practicable because of its small size and because training on sparse representations limits the number of neurons in the networks which results in fast training that allows for many experiments. As compared to convolutional networks for deep learning where suitable means to avoid the vanishing gradient problem are required (https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484 kernel sizes, strides, the number of kernels and other parameters must be defined there simply less one could do wrong when defining shallow MLPs. It deserves no further explanation that underfitting can potentially be due to an unsuitable network design or because predictions are impossible due to incongruencies or missing information in the sparse data set (LeCun et al., 2012). If the learning task is simple enough to be accomplished by MLPs (e.g. finding combinations of partial angles that correspond to the impression of curvature), one may hence speculate that underfitting can rather be explained by incongruencies or missing information in the labels then a result of issues in design and training hyperparameters of the network. This classical machine learning approach that relies on feature engineering was applied to predict features based on colour and partial angles of asparagus spears.

**Violet and rust prediction based on color histograms**
The initial approach of measuring the feature "violet" that is based on distribution of sufficiently intense colour hues in the violet range faces at least two drawbacks: First it requires to define two thresholds and second the impression of a violet asparagus piece could potentially be affected by the mix of colours (combinations of colours) that are potentially outside the violet range or are too pale to be considered (see XXX). The same holds for the feature "rust". Hence in a second approach histograms were computed for foreground pixels after transforming the images to palette images with 256 colour hues (see XXX). The resulting representation is arguably a sparse descriptor that allows to predict colour features using explicitly

defined rules or trainable machine learning models.

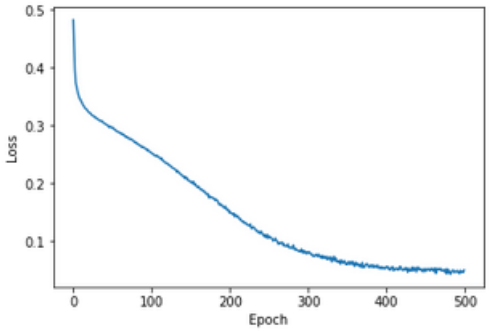



FIGURE 4.1: ??

| toprule | False positive | False negative | True positive | True negative | Sensitivity | Specificity |
|---|---|---|---|---|---|---|
| midrule violet | 0.04 | 0.03 | 0.05 | 0.88 | 0.62 | 0.96 |
| rost body | 0.19 | 0.14 | 0.33 | 0.34 | 0.71 | 0.65 |

A simple MLP with four hidden layers and 128 neurons in each of them was trained on the resulting normalized histograms of palette colours (ReLU activation / sigmoid activation in the final layer). Hyperparameters were optimized and the network was trained for a total of 500 epochs as the learning curve indicated convergence at this point.

**Curvature prediction based on partial angles**
Although the accuracies are far from perfect the results appear to be promising. The hit rate for curvature detection is high: 82% of all bended pieces were identified as such. In comparison, this holds for 71% of the pieces affected by rust and 62% of the violet pieces. In contrast, almost all pieces that are identified not to be violet are labelled accordingly (96% specificity) whereas the specificity for rust (65%) and curvature (67%) is lower. The receiver operating characteristic reveals that the prediction is of better quality for violet and curvature prediction as compared to rust prediction which is reflected in a smaller area under the curve. Possibly this reflects that rather small brown spots were considered rust by some coders that attributed labels.

| toprule | False positive | False negative | True positive | True negative | Sensitivity | Specificity |
|---|---|---|---|---|---|---|
| midrule bended | 0.19 | 0.07 | 0.34 | 0.4 | 0.82 | 0.67 |

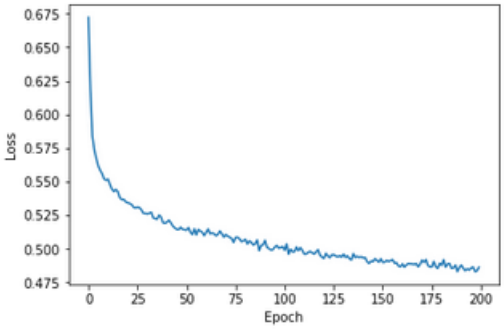



FIGURE 4.2: ??

Considering the low agreement in labeling, higher values for the specificity and sensitivity of the classifier were not expected. A likely explanation is that the model generalizes deviating understandings and incongruencies we had when attributing labels and affected the reliability of the data. Arguably only little information was discarded by computing the sparse representations that served as an input. Information about irregularities in the outline that a center line does not reflect but potentially contribute to the perception of curvature are not reflected in named sparse representation. The same holds for the spatial distribution of coloured pixels might contain additional information regarding rust and violet-detection. However, the major criteria are captured. As MLPs have little hyperparameters that are suitable for non-linear mappings and as the task of mapping high level features to human estimates appears to be rather simple, one could argue that there is little potential to improve the predictive quality using other techniques. By introducing a bias, the sensitivity (true positive rate) of the classifier can be adjusted at the cost of more false positives. Introducing a bias means that the threshold that is used to convert the floating point outputs of a neural network to booleans that indicate whether a feature is present or not can be set to values other than 0.5. The possibility of making the classifier more or less sensitive appears to be a good option to be implemented as a feature for customization by the user in asparagus sorting machines. The resulting behavior is reflected in the receiver operating characteristic that is calculated using different biases.

### 4.1.2 A dedicated network for head-related features

Some features relate to the asparagus heads only. Hence, it was assumed that classification is easiest when training a convolutional neural network on depictions of the respective region of the asparagus. Therefore, a dataset that consists only of images of the head area was used. Images of all three perspectives were appended horizontally such that each sample contains the information from all available viewpoints. This is especially important as rust affected spots are sometimes only visible from some angles. The depiction below shows one sample of rust affected asparagus heads.

A simple feedforward convolutional network was trained on the images. The features “flowering head” and “rust affected head” were chosen as target categories. The network comprises the input layer, three convolutional layers with kernel size two, a fully connected layer with 128 neurons as well as the output layer. For the final layer the sigmoid activation function was applied while the hidden layers have

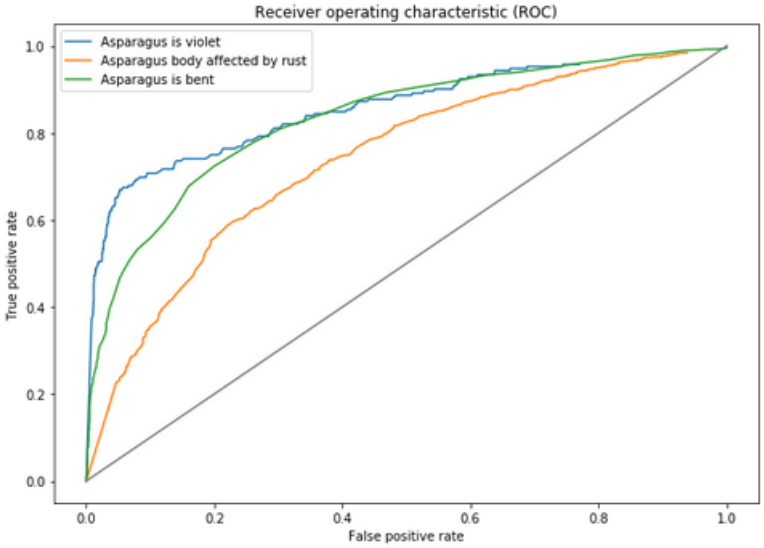



FIGURE 4.3: ??

FIGURE 4.4: ??

RELU activations. A dropout layer was added to avoid overfitting. The network was trained using mean squared error (MSE) as an error function. The development of loss in the learning curve indicated convergence after 40 epochs.

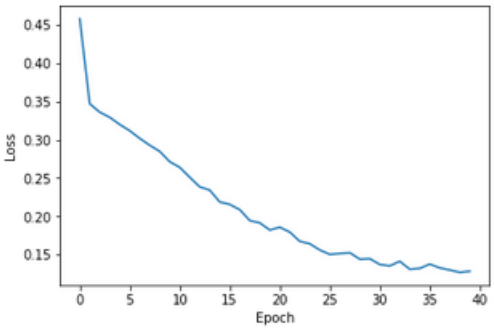



FIGURE 4.5: ??

The results for both features showed to be highly specific. In contrast the sensitivity is rather low. Only 55% of the asparagus pieces labelled as “flowering head” were identified as such whereas the true positive rate is only 19% for “rust head”. Given the low labeling agreement for these criteria these mediocre results are not surprising.

The ROC curve indicates how the classifiers respond to the introduction of a bias

and shows the overall prediction quality. The area under the curve is small for the feature "rost head". Beside incongruencies in the labels this is possibly due the choice of the head region. It might be the case that brown spots in regions other than the cropped head were considered as an indicator for a rusty head when attributing labels. Improvements by increasing the cropped head region appear to be possible.

| toprule | False positive | False negative | True positive | True negative | Sensitivity | Speci |
|---|---|---|---|---|---|---|
| midrule<br>blume | 0.04 | 0.06 | 0.08 | 0.82 | 0.55 | |
| rost<br>head | 0.02 | 0.13 | 0.03 | 0.83 | 0.19 | |

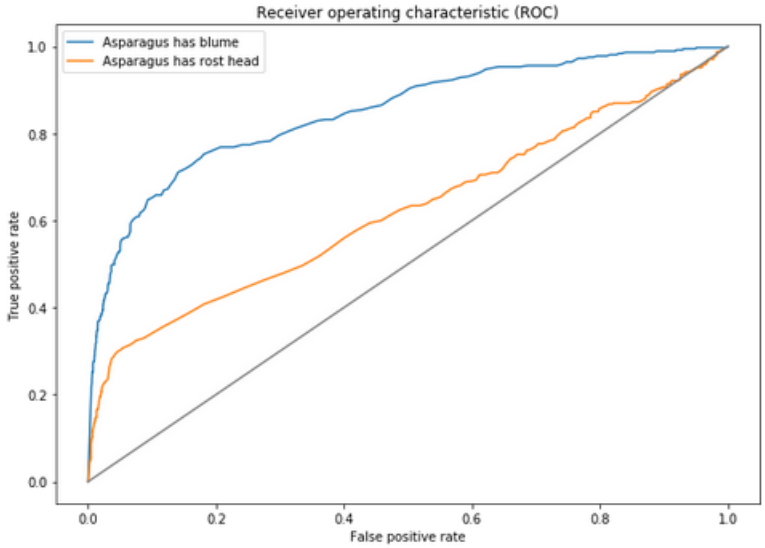



FIGURE 4.6: ??

### 4.1.3 Single-label classification

One supervised approach often used for training a model on image data is a convolutional neural network (CNN) (Géron, 2019) (LeCun and Bengio, 1995). The approach was tested for our data with the application of a CNN using single-label classification on features. CNN apply trainable filtering kernels to extract features from an input image and project these features onto feature maps (Géron, 2019) (Bishop, 2006) (LeCun and Bengio, 1995). Based on these maps, the label is predicted by a standard fully-connected layer.
Though using a CNN on the raw data images seemed promising, the results were only slightly better than chance level for the training accuracy.

When checking the asparagus images, an idea was to test whether the raw images with background and reduced to every 6th pixel (as created in chapter 3.5) can be

trained on a simple CNN using single-label classification. That is, the network tries to label the images for one of the 13 features that were manually extracted for the training image triplets. The data the model was trained on is stored in the grid folder XX and the labels were retrieved from the new_combined.csv at XX. The script to the model can be found on Github under XX.
As a starting point, a general model structure was needed as inspiration for a CNN. For example, the visual geometry group networks (VGGs) with varying depth seemed to be a good choice for image classification as their VGG16 had won the ImageNet challenge of 2014 and is often implemented for image classification tasks (https://neurohive.io/en/popular-networks/vgg16/) ( (Simonyan and Zisserman, 2014)). However, there are two major drawbacks on using them, namely their depth and the amount of fully-connected nodes which makes them slow to train and in need of a lot of memory space (https://neurohive.io/en/popular-networks/vgg16/) (Zhang et al., 2015). Part of these problems also arise with even deeper networks like ResNet (He et al., 2016b) (=https://neurohive.io/en/popular-networks/resnet/). Thus, AlexNet was chosen as a blueprint for the CNN because it is a small network in relation to other nets while still performing comparatively good (https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolu (Krizhevsky, Sutskever, and Hinton, 2012b) (Géron, 2019). As the variance in the data images is relatively small it was assumed that not as many layers are needed as employed in deeper networks like VGG (Géron, 2019).

In general, the model architecture is roughly based on AlexNet but it was strongly simplified to the level of variability and complexity of the underlying data. The network comprises four hidden layers: a convolutional layer, followed by a pooling layer, a second convolutional layer, and a dense layer. The input is an array of multiple horizontally stacked images. This input is trained on a set of binary labels containing information on whether the respective feature applies to the current image. The output of the network gives a prediction on each entered image gated by a sigmoid function on a range between 0 and 1. The rounded integer values of this output give a prediction of the apparent label. For the training phase of the model, the Adam optimizer was used because of its general acceptance as the state of the art optimizer for backpropagation (Kingma and Ba, 2014) (https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c).
As a loss function, binary cross entropy was used as it promised good results for binary single-label classification tasks (Géron, 2019) (https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a) (https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f6
When training artificial neural networks, it can be difficult to find clear guidelines on how to implement an architecture such that an optimal training performance is given (Heaton, 2015) (Géron, 2019) (=https://medium.com/tebs-lab/how-to-classify-mnist-digits-with-different-neural-network-architectures-39c75a0f03e3). Hence, the idea was to start with the simplest form of a CNN and then gradually increase the complexity of the network. While AlexNet provides a good baseline for an image classification network, its architecture was still assumed to be unnecessarily complex for the given task. First, the architecture was reduced to the minimum number of layers and parameters needed for a CNN. Over the period of training optimization, various processing steps and hyperparameters were implemented and compared according to their performance. During this process, the data was split between 12000 samples for training data and 1319 validation data in order to have a reasonable overview on the possible test performance and to prevent direct overfitting. The data used as test data was randomly chosen from the whole data

set.
The course of the hyperparameters is explained in the following. The batch size was initiated comparatively low with 64 samples per batch but soon increased to a value of 512 samples. The larger batch size was implemented in order to guarantee the convergence of the training data, since smaller batch sizes resulted in jumping gradients, which were not able to converge into any minimum (Bengio, 2012) (QUELLE).
As for the learning rate, various learning rates were tested during the optimization process. The initial learning rate of 0.003 (which is the standard learning rate for Adam optimizers in Tensorflow (Kingma and Ba, 2014) (Géron, 2019)) was soon found to be too large to guarantee convergence of the algorithm. Thus, the learning rate was gradually decreased and found to be most effective in the range of 0.000001 (1e-6) to 0.00000001 (1e-8). Learning rates as small as 0.00001 (1e-5) still seemed too large for training the net. In addition, a gradually decreasing learning rate was implemented in order to make the training more effective (Bengio, 2012), even though, in theory, the Adam optimizer already manages learning rate decay (Kingma and Ba, 2014). Starting with the smallest layer size possible, more layers were added. For example, for the feature 'fractured' one layer for the edge detection should have sufficed, or a layer for colour detection for the feature 'violet'. For other, higher-level features, such as bent, more convolutional layers were expected to be more helpful (Géron, 2019). During the training process, it was settled to a model with two convolutional layers, one max pooling layer and one hidden dense layer.
A small number of kernels was thought to be enough compared to the number of kernels of AlexNet since for single-label classification fewer kernels were expected to be needed (QUELLE). The kernel size was picked to be 32 53 $\times$ 4 5 kernels for the first convolutional layer and 64 33 $\times$ 4 3 kernels for the second convolutional layer. These sizes were assumed to be sufficient, especially since increasing the number of kernels did not lead to any better results.
Both convolutional layers were built with batch normalization nearly from the start of implementation of the network. To guarantee that exploding or vanishing gradients pose no further problem, the gradients were inspected visually and the results gave no reason for assuming problems with exploding or vanishing gradients (Pascanu, Mikolov, and Bengio, 2012). A next step was to weight the loss function because of the largely unbalanced data (He and Garcia, 2009) (Batista, Prati, and Monard, 2004). This ended with no better results, however, as the model still tended to make an unbalanced prediction by classifying all values as negative samples. Another idea was to reduce the dataset to make the number of images with the regarding feature present even to the number of images where it is absent. This translates to throwing away valuable data, which can otherwise provide information about negative cases to support the model in it's training (Batista, Prati, and Monard, 2004). Thus, it was decided to keep all data images and instead balance the data by multiplying the minority of samples to match the number of contrary samples. As there was no feature positively exceeding a presence of 50% in the data, solely positive labels were oversampled. The balancing was only performed on the training data, while the test data was not changed.
To improve training performance, some kinds of data augmentation were implemented. First, the images were flipped horizontally. This seemed to be a valid enhancement to the training data since it resulted in similar images to the original data (https://machinelearningmastery.com/how-to-configure-image-data-augmentation-
Additionally, small changes in image angles (up to 5°) were tested, however, this type of data augmentation was neither effective nor convenient to be computed.

In most cases, around 300 training steps were performed for the training, translating to XX epochs.

The results of the CNN showed for all features, that the training accuracy as well as the test accuracy converges. Further, the model does not overfit. The training loss and the test loss both decrease, with the training loss usually decreasing from XX to XX, and the test loss from around XX to XX. However, the training accuracy is not notably better than a 'best guess prediction' (always predicting one class). In no case the train accuracy considerably exceeded 50%. The test accuracy usually increased up to XX - XX. Sensitivity and specificity were, put together, often not considerably higher than 1.
Best results were achieved for the feature XX with a sensitivity of XX and a specificity of XX, while worst results were achieved for XX with a sensitivity of XX and a specificity of XX.
The results can be interpreted in that the simple CNN seems not able to notably learn better than a "best guess prediction"(= predicting only 0.0 or 1.0). For the data of this format, the model converges fast to a minimum. However, this minimum seems to be the "best guess" minimum and it is questionable whether it is possible to find the global minimum on the error surface by using a gradient descent algorithm. It might be that the network is not deep enough to find the right parameters to separate positive from negative samples (QUELLE). However, a further increase of hidden layers by adding a third convolutional layer and a second pooling layer did not lead to better results than for the original model. Added together, sensitivity and specificity were never considerably above 1 but usually below it. Either the network was good at predicting the feature to be present (1.0) or the feature not to be present (0.0) which essentially translates to the problem of not finding a minimum better than the "best guess prediction" (QUELLE).
The training accuracy converging to 50% gave rise to questions. Even while the batch size was at 512 samples per batch and the training data was balanced, with its actual parameters the model seemed to find no hint how to distinguish the samples.
A further influence on the training of the model could be the unbalanced data set. It was evened out, however, by this a lot of samples were represented multiple times in the training data. This was especially the case for features with low original representation, such as e.g. the feature 'fractured' or the feature 'violet'. Thus, it was expected that the model might overfit on the training data due to the oversampling of the positive samples in the training data which did not occur (He and Garcia, 2009).

On a more general level concerning the input, there might have been trouble for the model with the conformity of the data. As the samples were labelled by humans, there is a subjective bias in the data that might stop the model from finding a global difference between positive and negative samples (QUELLE).
Another factor poses the use of image triplets. It might have been easier for the network to learn to distinguish the data samples if single images would have been manually labelled instead of image triplets (QUELLE).
One idea for optimizing the process and to reach better results for features like, e.g. rust or violet which rely on a colour parameter, could be to simplify the input for the network (QUELLE). That is, the images are prepared in a way to guide the network into the direction of parameters that we want it to discern.

In conclusion, the simple CNN seems not able to learn from the data. The train

loss decreases and converges but the train accuracy cannot considerably exceed above chance level of 50%. Although the test accuracy is increasing this renders no further information about the classification ability of the net. Put together, sensitivity and specificity rarely reach above 1. Rather, one of both tends to reach close to 1 while the other is close to 0. Further preprocessing of the image data before training might have improved the results.

### 4.1.4 Multi-label classification

Building on the standard single-label classification we were further interested whether it would be possible to build a model that can predict several feature labels at the same time. A multi-label classification model hereby gets an image as the input and learns to predict the presence or absence of the feature labels.
For our model we decided to use a small convolutional neural network as described below and the features that we labelled by hand. Each of the six features (hollow, flower, rust head, rust body, bent and violet) is encoded by a binary output in the target vector, indicating whether the asparagus exhibits the feature in question or not.

BACKGROUND
Multi-label classification is a useful tool for classification problems in which several classes are assigned to a single input. In contrast to a multi-class classification, where the model is supposed to predict the most likely class for an input, the multi-label classification makes a prediction for each class separately, determining whether the class is present in the image or not. While the different classes are mutually-exclusive in the multi-class classification, they can be related in the multi-label classification. Further, there is no limit on how many classes can be depicted in one image. It is possible that all or none of the classes are present.
One could think of the multi-label classification task as consisting of different subtasks. Therefore, some multi-label classification problems can be transformed. There are two eminent ways to do so, either the problems are transformed into multiple binary classification tasks or into one multi-class classification task.
In the first approach, a new model for each feature is trained, which are then combined to give a single output. That means that all features are independent of one another, because they are learned separately. This can be seen as one of the major drawbacks as it is not always clear whether features are related or not and in many cases they are. Therefore, we decided to not only use single-class classification as described in chapter 4.1.1 but to explore the possibilities of multi-label classification.
In the second approach, each possible combination of features is interpreted as one class. For a classification problem with 6 features that means there are 64 classes to be learned. The problems with this approach are, on the one hand, the exponentially increasing number of classes, and on the other hand the sparsity of samples per class. In many cases, some of the classes will be highly underrepresented or even empty. For that reason we decided not to elaborate this approach further and implement a model for multi-label classification without transforming the task to a multi-class problem.

Inspiration for the model gave a blogpost (https://towardsdatascience.com/multi-label-cl which aims to classify images of the MNIST fashion dataset in the context of multi-label, rather than multi-class classification. The author altered the dataset in such a way that each input image contains four randomly selected items from the MNIST fashion dataset. The model then learns to predict which classes are present in the

image. There is no restriction on the random selection, therefore the items can be all from the same class, from four different classes or anything in between. The target vector has 10 values, one for each class, which are either 0 or 1 depending on whether that class can be found in the input image or not.

This model was chosen as inspiration for two main reasons. Firstly, it tackles a similar problem as ours. In both cases, the model is supposed to predict the presence or absence of different fashion items or labelled features, respectively, and the number of classes is similar, it is 10 in the fashion example and 6 for our model.
Secondly, the model uses a dataset of similar size, 9000 images were used for training and 1000 for validation, while we were training on 10800 images and validating on 1200. Despite the rather small dataset in comparison to many other machine learning problems, good results with an accuracy of 95-96% were reached (QUELLE https://towardsdatascience.com/multi-label-classification-and-class-activation-map-on-
This leads us to think, it might be a model with a good complexity for our problem too, as it is complex enough to model the underlying distribution, but not too complex for the medium-sized dataset.

MODEL STRUCTURE
A classical convolutional neural network was chosen for the multi-label classification task. It consists of five blocks of convolution layers with max pooling layers followed by a global average pooling layer and a dense layer.

In contrast to multi-class classification models, where usually a softmax activation function is used in the last layer together with a categorical cross entropy loss, the multi-label classification model uses a sigmoid activation function and a binary cross entropy loss.

As the input of the model a concatenation of the three perspectives of each asparagus piece were used in order to maximize the information the model gets about each asparagus. This yields input images that look like the three asparagus pieces are laying side by side. Further the images were downscaled by a factor of 6 to facilitate training.

The output, or target, of the model is a vector of length six in which each position encodes one of the six hand-labelled features (hollow, flower, rust head, rust body, bent and violet). Each feature can either be applicable to the input or not, which leads to a "1" or "0" in the target vector, respectively.

Several loss functions were tested to improve the models performance. For example, the hamming loss, which uses the fraction of the wrong labels to the total number of labels. Additionally to the in-built loss functions from keras, a custom loss function was implemented, that penalizes falsely classifying a feature as present more than falsely classifying one as absent. The motivation for this custom loss was the fact that the two labels "0" and "1" are highly unbalanced. As previously stated, there are noticeably more zeros than ones in many classes. To be more precise, the model can get an accuracy of 77% by labeling all features as zero. By penalizing this error more, we intended to counteract the unbalanced dataset. But at the end, the binary cross entropy loss remained the one with the best results.
Further, it was tested whether regularization would improve the performance of the model on the validation data by preventing overfitting. For this, the model was

trained by adding L1 or L2 regularization, respectively, to all five of the convolutional layers. Hereby, a kernel regularization was implemented with a value of 0.01.

L1 and L2 regularization can both be interpreted as constraints to the optimization that have to be considered when minimizing the loss term. The main difference between the two is that L1 regularization reduces the coefficient of irrelevant features to zero, which means they are removed completely. Hence, L1 regularization allows for sparse models and can be seen as a selection mechanism for features. As the inputs to our model are images, that consist of a large number of pixels, and additionally a large portion of those pixels are black, because the background was removed, it appears to be a good idea to reduce the number of features taken into account in the early layers. L2 regularization, on the contrary, does not set coefficients to zero, but punishes large coefficients more than smaller ones. This way the error is better distributed over the whole vector.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 224, 182, 32) | 896 |
| max_pooling2d_1 (MaxPooling2 | (None, 112, 91, 32) | 0 |
| conv2d_2 (Conv2D) | (None, 112, 91, 32) | 9248 |
| max_pooling2d_2 (MaxPooling2 | (None, 56, 45, 32) | 0 |
| conv2d_3 (Conv2D) | (None, 56, 45, 32) | 9248 |
| max_pooling2d_3 (MaxPooling2 | (None, 28, 22, 32) | 0 |
| conv2d_4 (Conv2D) | (None, 28, 22, 32) | 9248 |
| max_pooling2d_4 (MaxPooling2 | (None, 14, 11, 32) | 0 |
| conv2d_5 (Conv2D) | (None, 14, 11, 32) | 9248 |
| max_pooling2d_5 (MaxPooling2 | (None, 7, 5, 32) | 0 |
| global_average_pooling2d_1 ( | (None, 32) | 0 |
| dense_1 (Dense) | (None, 6) | 198 |

Total params: 38,086
Trainable params: 38,086
Non-trainable params: 0

FIGURE 4.7: This table shows the structure of the multilabel classification model. It describes which layers were implemented, how the output changes in each layer and how many parameters were trained in each layer and in total.

RESULTS
As one can see in the figures XX-YY, all the different approaches explained above show a similar behaviour in accuracy and loss values. The training and validation accuracy increase slowly but steadily with the training accuracy always being a little higher than the validation accuracy. The training loss decreases rapidly, while the validation loss only decreases very little and shows random fluctuations. This can be an indicator for overfitting. Usually, L1 and L2 regularization are used to

prevent overfitting, but in our case it did not improve the results, as one can see in figure XX.
When looking at the true positive rates, also called sensitivity, and the true negative rates or specificity, one can see that both increase during the training process , while the false negative and false positive rates decrease with the same slope. The false positive and false negative rates are mirror images to the true positive and true negative rates with the mirroring axis at the 50% mark. It can be observed that the rates change rapidly in the first two to four epochs, after which the change progresses slowly in the same direction with no greater disturbances. One exception is the model trained with the L2 loss, which does not show a large change in either of the rates.

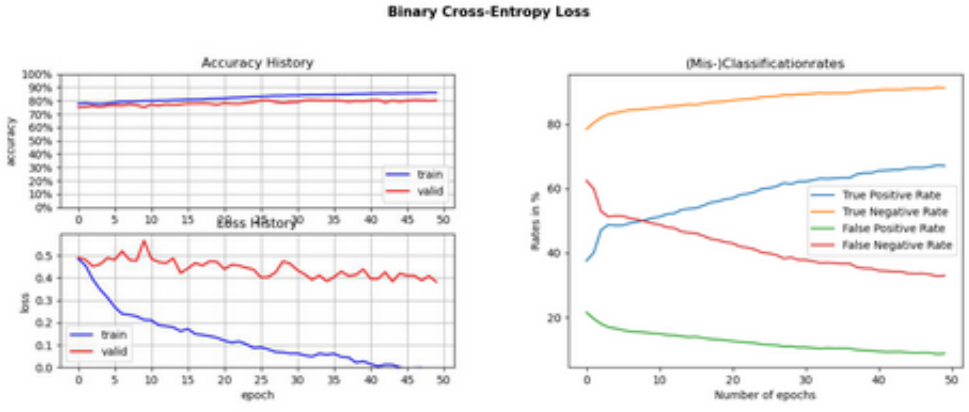



FIGURE 4.8: These graphs show the evaluation of the training with binary cross-entropy loss. The model was trained over 50 epochs and the accuracy and loss was measured. Further, the false/true positive/negative rates were determined.

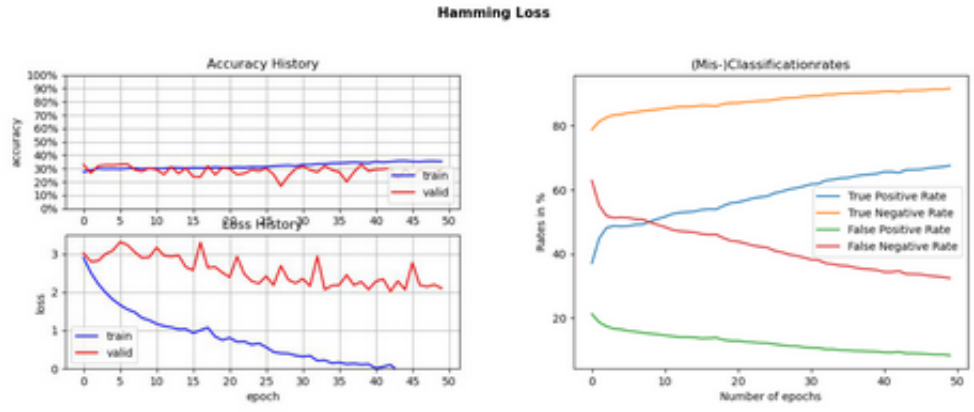



FIGURE 4.9: These graphs show the evaluation of the training with hamming loss. The model was trained over 50 epochs and the accuracy and loss was measured. Further, the false/true positive/negative rates were determined.

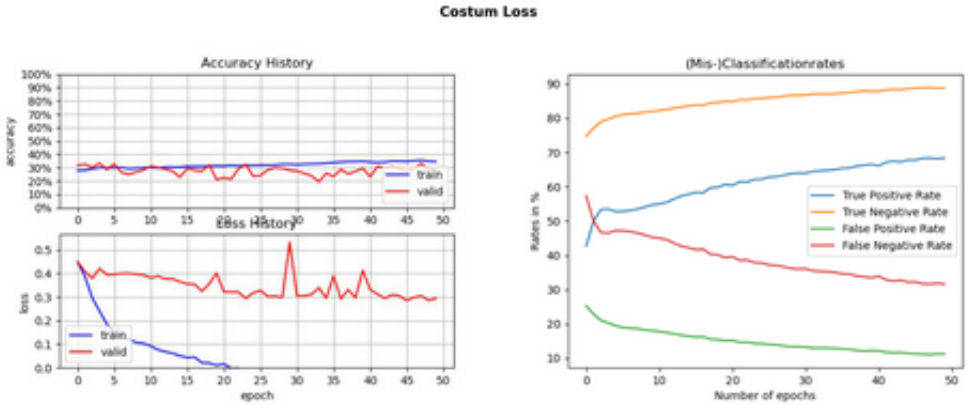



FIGURE 4.10: These graphs show the evaluation of the training with costum loss that punishes falsely classified ones more than falsely classified zeros. The model was trained over 50 epochs and the accuracy and loss was measured. Further, the false/true positive/negative rates were determined.

When comparing the three different loss functions, it is noticeable that the binary cross entropy loss has significantly larger accuracy values than the hamming loss and the costum loss. Its values start at 75%, while they start at around 30% for the other two loss functions. The behaviour of the curves and the (mis-)classification rates, however, are very similar in all three approaches. The specificities start off very high with values around 78,46% for the binary cross entropy loss, 78,73% for the hamming loss and 74,74% for the costum loss, and increase further during the training. The highest values are reached with the binary cross entropy loss (91,41%) , closely followed by the hamming loss (91,37%) and the costum loss (88,75%). The sensitivity values start off lower, at around 37% to 42%, and increase rapidly in the first few epochs, after which the rates proceed to increase but with a narrower slope. They reach values of up to 67,27% with the binary cross entropy loss, 68,17% with the hamming loss and 68,17% with the costum loss. As stated above, the false negative and false positive rates show the same slope but in the opposite direction.

The accuracy values of the models that were trained with L1 or L2 regularization, respectively, do not change over the epochs. The same holds for the validation loss. The training loss decreases in the first few epochs and remains stable after that. While the (mis)-classification rates of the model trained with L1 regularization behave similarly to the ones trained with no regularization, the rates of the model trained with L2 regularization show a smaller increase and lack the fast change in the first epochs.

DISCUSSION

The slopes of all curves indicate that the model is learning, because they are increasing in the case of the accuracy, sensitivity and specificity and decreasing in the case of the loss, false positive and false negative rates until the end of training. Hence, one might think that a longer training period will lead to better results. But the training loss decreases very rapidly while the validation loss does not. This suggests overfitting of the model, a problem which gets worse when increasing

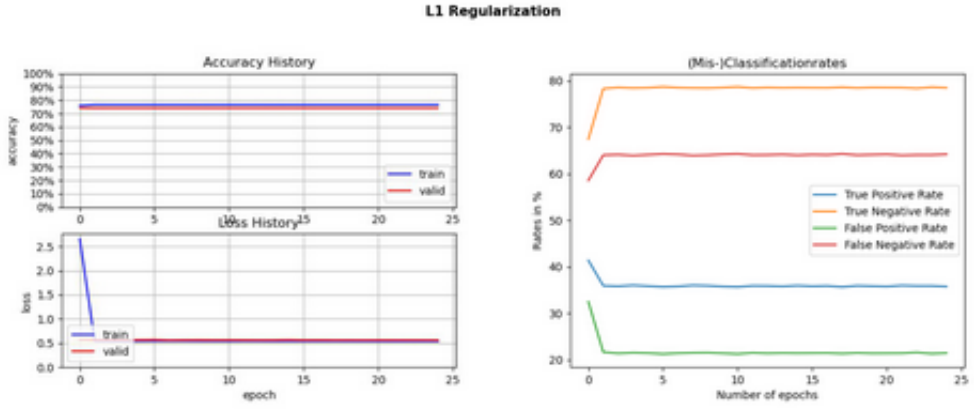



FIGURE 4.11: These graphs show the evaluation of the training with L1 regularization. As the loss function the binary cross-entropy loss was used. The model was trained over 25 epochs and the accuracy and loss was measured. Further, the false/true positive/negative rates were determined.

the training steps. Therefore, a longer training period most likely will not increase performance unless overfitting is prevented. As one can see in the result section neither L1 nor L2 regularization alone were able to prevent overfitting. Another common practice that could be tested is the drop-out, in which a certain amount of nodes are left out in different backpropagation steps. This way the model learns to not rely on a small number of nodes but distribute the information between all nodes available. Hence, the coefficients remain smaller. Another way to prevent overfitting is to reduce the model's complexity. A model with fewer parameters to train, is less prone to overfitting. A fitting degree of complexity should be found to model the data sufficiently good without losing the possibility of generalization.

Accuracy alone might not be a good indicator to evaluate a multi-label model (Gibaja and Ventura, 2015). As it highly depends on the loss function, it may have misleading results. This can be seen in the comparison between the three different loss functions. Although the sensitivity and specificity show similar values, the accuracy values suggest that the binary cross entropy loss outperforms the other two loss functions by far. The accuracy of the model trained with the binary cross entropy loss has an accuracy more than twice as high, but when looking at the slope of the curve it appears that the model with the binary cross entropy loss does not, in fact, perform better than the other two models, because all three have an increase of accuracy of roughly 10% and a similar sensitivity and specificity. This indicates that the slope of the accuracy function can be considered to evaluate the training process of the model, but the real values should be interpreted with caution.

One thing that comes to attention when looking at the (mis-)classification rates is that the sensitivities are a lot lower than the specificities. A reason for that might be that there are fewer positive values in the dataset and they are, thereby, more difficult to learn. The model might have learned that, if unsure, a zero is the more likely guess.

The L1 and L2 regularization both seem to prevent the model from learning all together instead of only preventing overfitting. A reason for that might be that the regularization factor was too high. More experiments should be conducted with

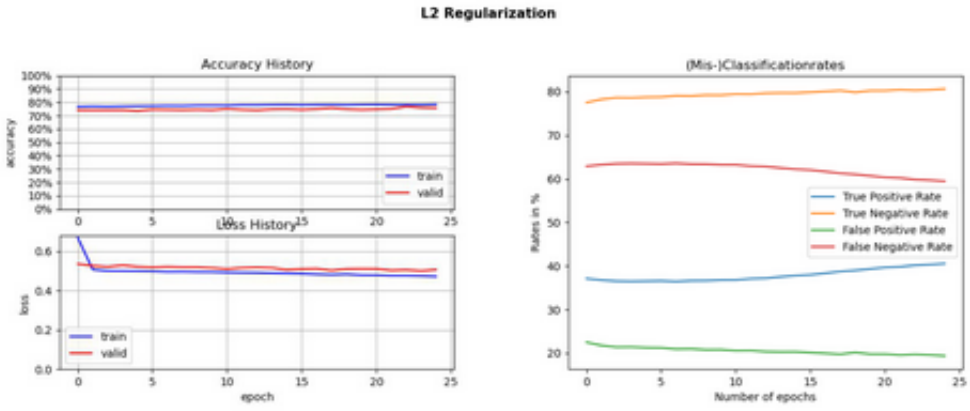



FIGURE 4.12: These graphs show the evaluation of the training with L2 regularization. As the loss function the binary cross-entropy loss was used. The model was trained over 25 epochs and the accuracy and loss was measured. Further, the false/true positive/negative rates were determined.

varying values to test this hypothesis.

In summary, the model improves its sensitivity and specificity, but it seems like it does so by overfitting the training data. Therefore, the next step should be to prevent the model from overfitting and after this problem is solved, it should be tested whether additional changes can improve the performance of the model further.

### 4.1.5 From feature to label

Approximately 200 asparagus pieces per label class were pre-sorted [cf. Chapter 1.4] and served as ground truth mappings between input images and output class labels. These images were manually annotated with features (see Chapter X.X and Table X.X table with the features). This allowed us to break up the classification process into two steps: In the first step we predict feature values from images and in a second step we predict class labels from feature values.

This chapter deals with the second step: using supervised learning to predict 13 ground truth class labels based on manually labelled features. We build a unified interface to load different models, train them and analyze their predictions. It provides compatibility for scikit-learn as well as for keras models. To explore the data and visualize the results, a streamlit app was built(Footnote: the source code can be found in the GitHub repository at code/pipeline). This has the advantage that the user can easily load a model which is automatically trained, select an asparagus piece, see the corresponding pictures and the predicted class label. The user is sees the distribution of the selected data (Figure XX) and can inspect the absolute and relative number of correctly and incorrectly classified asparagus pieces in the confusion matrix (Figure XX). n a confusion matrix (Figure X)", "overall distribution of true positives and negatives (Figure Y)" etc. We can clearly see that ... Furthermore, the user can see the images of an individual asparagus piece and check if his/her expectations for the manually labelled features are satisfied and if the class label of the piece is predicted correctly (Figure XX).

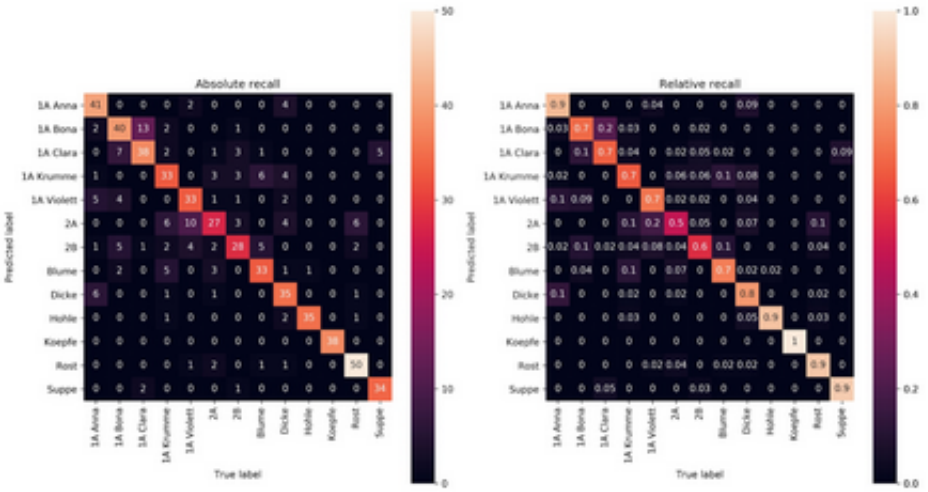



FIGURE 4.13: ??

Two exemplary models were implemented and tested to predict the classes from the features. The first one is a random forest (Breiman, 2001), with 100 trees.

Random forests are a popular machine learning approach, because they reach good results in both regression and classification tasks. Further, they are based on decision trees which offer an intuitive interpretation. Although decision trees themselves are powerful tools, they are prone to overfitting. Random forests aim to avoid this problem, while still using the advantages of decision trees, namely that they are flexible, easy to use and fast to train. They do so by training several decision trees on different random parts of the data, after which a majority voting decides on the final output or class. An additional trick is to use a random selection of features at each branch. This reduces the influence of highly correlated features and thereby makes the random forest more robust.

The second model to predict classes from features is a multilayer perceptron (six fully connected layers). It was only implemented to show how to integrate other models, but achieves a similar score as the random forest classifier when trained for XXX epochs (score of 0.76). However, it takes longer to train as the random forest classifier.

The score of 0.76 on the VALIDATION SET means that . . . . F1 / Precision / Recall Compared to the accuracy that would be achieved by random guessing (0.08 for 13 uniformly distributed classes), the random forest model can classify 76% of the asparagus pieces in the validation set correctly.
There are still many unanswered questions about how the selection of features or the agreement on the values of the labelled features influence the performance of the described classifiers. Further work is required to implement the decision tree described by the local farmer [cf. Chapter 1.3]. One could test if this decision tree which is based on expert knowledge can outperform the random forest trained on the training samples.

## 4.2 Unsupervised learning

Unsupervised learning are all kinds of machine learning algorithms which are not supervised. More specifically, they work without a known goal, a reward system or prior training, and find a structure within the data, or some form of clustering of data points. In supervised approaches, the model is given both the input and the labels. In unsupervised learning approaches, the model only receives the input data. Thus, unsupervised learning works without training samples, and without labelled data. The goal of unsupervised learning algorithms is to find hidden structures in the data.

Dimension reduction algorithms and clustering algorithms have been identified as the two main classes of unsupervised machine learning algorithms which are used in image categorization (Olaode, Naghdy, and Todd, 2014).

Multivariate datasets are generally high dimensional. However, it is common that some parts of that variable space are more filled with data points than others. A big amount of the high dimensional variable space is not used. In order to recognize a structure or pattern in the data, it is necessary to reduce the number of dimensions. For this, both linear- as well as non-linear approaches can be applied. Linear unsupervised learning methods for which also descriptive statistics can be acquired are e.g. Principal Component Analysis (PCA), Non-negative matrix factorization, and Independent component analysis (Olaode, Naghdy, and Todd, 2014). Some examples for non-linear approaches would be Kernel PCA, ( Scholkopf et al. – im oben genannten paper), Isometric Feature Mapping (ISOMAP), Local Linear Embedding, and Local Multi-Dimensional Scaling (Local MDS).
For the current work, the linear dimension reduction algorithm PCA was chosen.

### 4.2.1 Principal Component Analysis

PCA was chosen, because it is one of the standard unsupervised machine learning methods. Moreover, it is a linear, non-parametric method and widespread application to extract relevant information from high dimensional datasets. The hope is to reduce the complexity of the data, by only a minor loss of information (**shlens2009**). Besides being a dimension reduction algorithm, PCA can also be useful to visualize the data, filter noise, extract features, or compress the data.

Our initial aim was to reduce the dimension of our dataset for further models. As PCA was applied at the beginning of the data inspection, we also had the aim to visualize our data in a three-dimensional space, in order to get a better understanding of the data distribution. The images that comprise our original dataset have a high quality, and instead of only reducing the pixel size, we aimed for reducing the information contained in named depictions by analyzing the principal components in a first step and projecting all relevant images into the lower dimensional space. This information could serve as the input for later machine learning algorithms. As we decided to use an approach for semi supervised learning that is based on neural networks later we focussed on the question what principal components reveal about the data.

PCA relies on linear algebra. A general assumption especially with respect to images is that large variances are accompanied by important structure. The covariance matrix of the data reveals information about the overall structure and orientation of the data points in the multivariate space. The axis with the largest variance is

set as the first principal component. An additional assumption is that the principal components are orthogonal to each other. Therefore, the second principal component is the highest variability of all directions which are orthogonal to the first one (**Bohling2006**). The covariance between each pair of principal components is zero, as they are uncorrelated and generally, the higher the eigenvalues, the more useful it is for the analysis. As eigenvalues specify the variance of the data, more eigenvalues indicate more variation about more features.

The result of a PCA is a representation of the data in a new lower coordinate system, which depends on the axes of the largest variance. The dimensionality of this lower coordinate system should depend on the values of the eigenvalues. When plotting the data along the axes of the principal components, it is often easier to understand and interpret the data, than in the original variable space.

It is a consensus in the literature, that PCA can be a good method to apply dimension reduction on images (Turk and Pentland, 1991) (Lata et al., 2009). However, there are several different ways on how to do so. First of all, it has to be decided if the PCA should be performed on a black and white image, or if the structure of the data requires the use of colorized images. When working on black-and-white images, only one data point per pixel is given, therefore, performing a PCA is less computationally expensive, and also finding structure is easier. However, as we need to be able to recognize violet as well as rust, it was important to us, to be able to differentiate between color nuances, which cannot be represented in a black and white image.

There are three different opportunities regarding our project which can be considered as useful ways on on how to perform a PCA. First, it can be performed on images of different classes at the same time – similar as to capture several images of several people in one database, on which a PCA is performed. In our case this would mean that a PCA is applied to a dataset of several input images of all 13 classes of asparagus images. Second way would be to perform a PCA separately on each group. This way, an "Eigenasparagus" in each group would be calculated, and distances between the "Eigenasparagus" of different groups could be measured. Third PCA can be employed feature-wise. In this case, the dataset would consist of a collection of images with a certain feature present vs a collection of the same size with the feature absent.

We calculated the PCA for black-and-white images, as well as on images without background, and also tried to work on all groups at the same time, group-wise as well as feature-wise. We decided to perform our final PCA on sliced RGB images with background, that are labelled with features by us with the hand-label-app, as this seems to yield the best results. An amount of 400 pictures per feature was considered to later on perform a binary PCA for each feature (either the feature is absent or present).

The 200 pictures where the certain feature is present as well as the 200 pictures where the certain feature is absent are extracted through a function in code/pca_code/feature_ids.py. This function loops over the combinded_new.csv csv-file, where all hand-labelled information is stored as well as the path to the labeled pictures. For each feature a matrix is created, storing 200 pictures with the present feature and 200 pictures without the feature. E.g., m_hollow is the matrix created for the feature hollow (shape = 400, img_shape[0]$\times$ img_shape[1]$\times$ img_shape[2]). The first 200 entries

in the matrix are pictures of hollow asparagus, the last 200 pictures show asparagus, which is not hollow. These matrices were calculated for the features hollow, flower, rusty head, rusty body, bent, violet, length and width. The data points of those 400 images in 2D space can be seen in figure XX (die figure mit den scatterplots).

For all these features a PCA is calculated by first standardizing the matrix pixelwise (dimensions: 13403 $\times$ 4 3463 $\times$ 4 3), calculating the covariance matrix and then extracting the ordered eigenvalues. The principal components are calculated multiplying these eigenvectors with the standardized matrix. The highest 10 eigenvalues were plotted to visually decide where to set the threshold of how many principal components will be further included (see figure XX - die figure mit dem graph - die ersten 10 eigenvalues). The feature space, the principal components and the standardized matrices are saved to later perform a recognition function. The first ten "Eigenasparagus" for each feature can be seen in figure XX (die schön bunten spargelbilder).

The recognize function is a control function, which performs on unseen images and tries to predict if a feature is absent or present. This function is also performed feature-wise. It reads in a new picture of one asparagus, which is not part of the asparagus database, so not within the 400 images. Then, it searches for this picture the most similar asparagus in the feature matrices (which are 200 pictures with the feature, 200 pictures without).
In greater detail, the input picture is first centered by subtracting the mean asparagus and then the picture is projected into the corresponding feature-space. That means the picture is "translated" into the lower dimensional space, in order to compare it to the known 400 pictures. The comparison is made through calculating the distance between the single centered eigenasparagus and the 400 pictures in the feature space, by using the cdist function of the SciPy software. The smallest distance is considered as the most similar asparagus. If the index of the most similar asparagus is smaller than 200 we know that the feature is present, if it is above 200 the feature is absent. By comparing this to the information of the single asparagus piece, we know if the new asparagus has the same feature as its closest asparagus in the feature space, or not. By doing this for several images, we can already presume if the two features are likely to be easily separable or not. By evaluating this, we have a measure of how well our used principal components capture the distinguishing information of each feature.

The final results of the PCA are given feature-wise and were not translated back on the initial 13 groups. The features on which results are given are: hollow, flower, violet, rusty body, bent, width and length. All features are binary. For the first five features, the manually hand-labelled information was taken, for the last two features, the information, which was extracted by the algorithms used in the hand-label app were taken. The decision boundary for the first five therefore depends on our predefined criteria on labeling. The decision boundary for width was narrower or equal to 20 mm or wider than 20 mm, for length shorter or equal to 210mm or longer than 210 mm.

There are no results for the feature broken, even though it is one of the initial main features, as there were only five labelled pictures for this category. Therefore, it was excluded for further analysis. For the feature rust head, a calculation problem

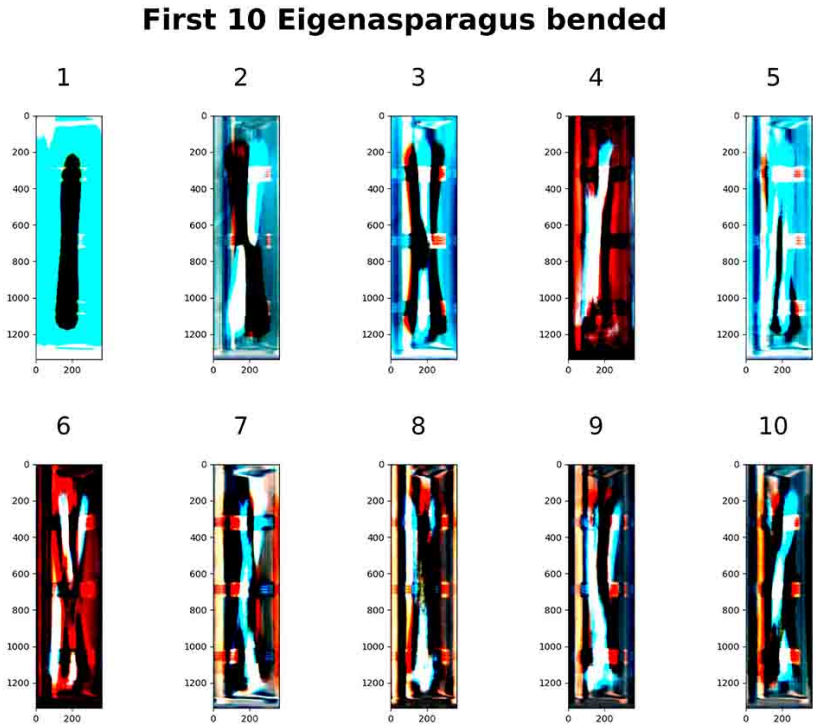



FIGURE 4.14: ??

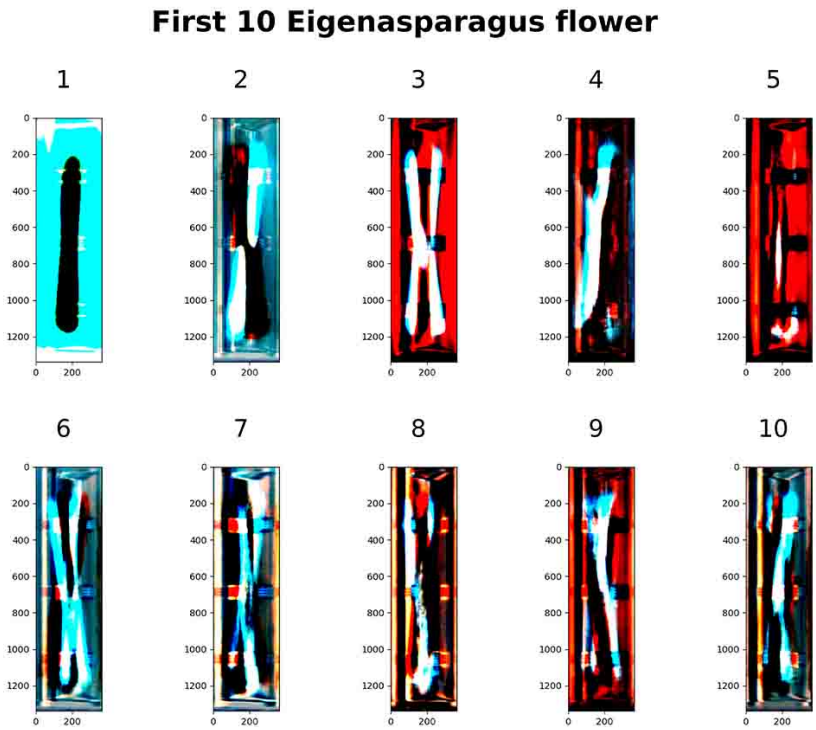



FIGURE 4.15: ??

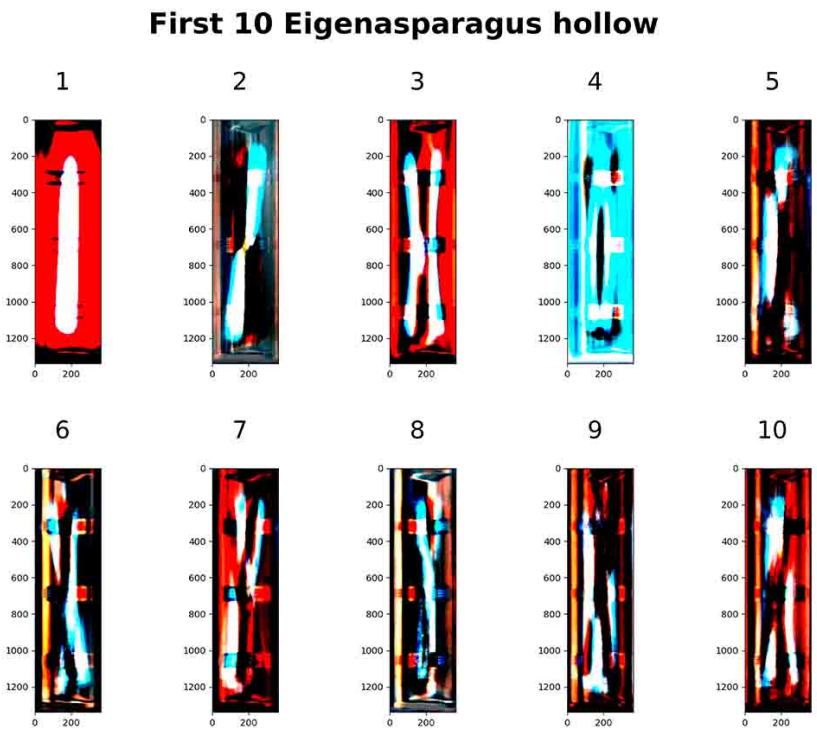



FIGURE 4.16: ??

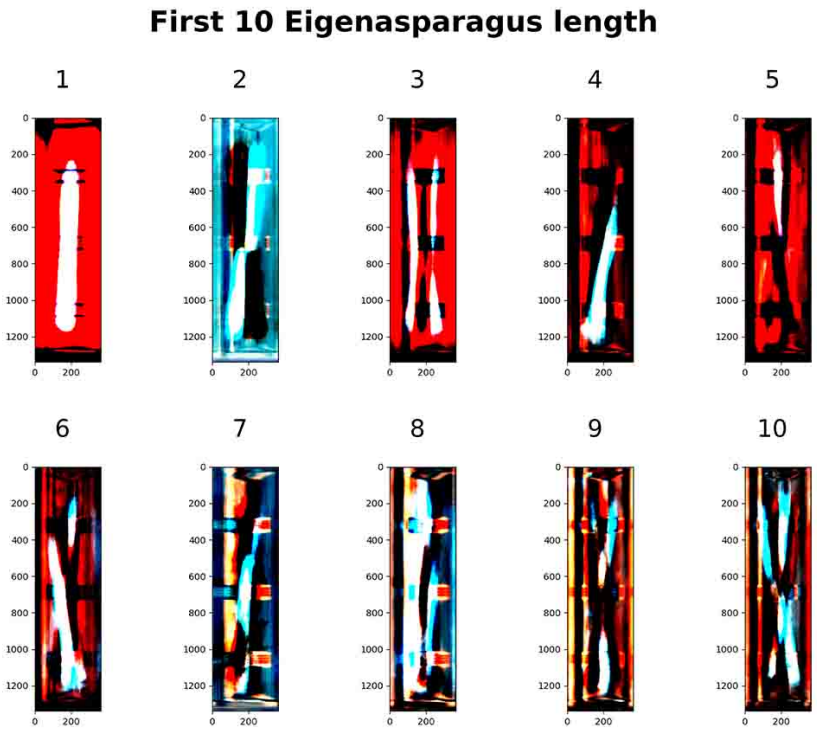



FIGURE 4.17: ??

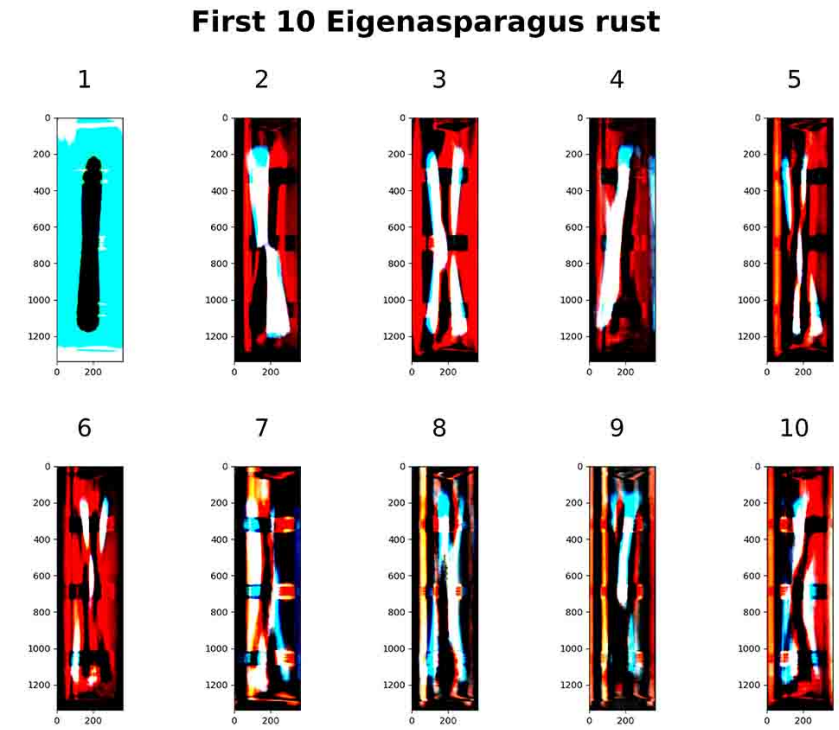



FIGURE 4.18: ??

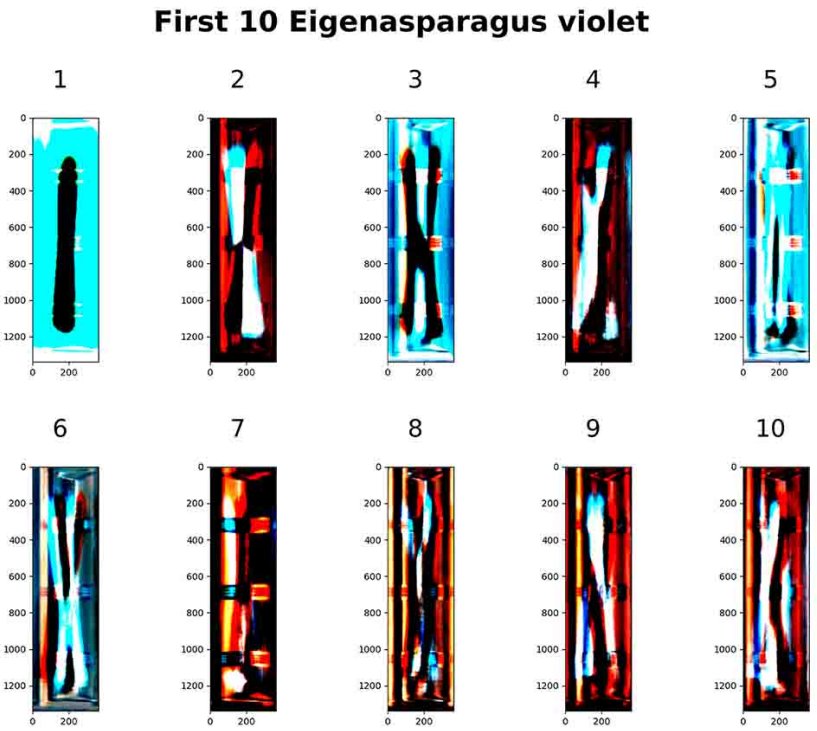



FIGURE 4.19: ??

emerged during the process. For an unexplainable reason, there were complex values in the calculation of the principal components. Due to time constraints, this problem was not solved, yet. Therefore, plotting of the feature rust head was not possible.

By applying the feature_pca.py file on each feature, the eigenvectors, eigenvalues, and principal components for each feature were computed and stored. In all cases, the first principal component is quite high (between 286,94 and 254,78), and drops down rapidly afterwards. The values of the principal components after the fourth principal component are very low (<= 9).

After inspection of the graph of the first 10 principal components, only the first four principal components were used for the analysis, as there was either a sharp drop in the slope after the first 4 or to maintain continuity between the different features. Following, the corresponding feature space is that space spanned by the first four principal components.

We plotted the four-dimensional data in the feature space as scatterplots in three-dimensional space and the fourth dimension as color, but as it was difficult to interpret and visually understand, we decided to show plots of only the first two dimensions, here.

The scatterplots in Figure 4.21 show the data of each feature lined up along the axes of the first two principal components of each feature.

For the recognition function, we used the same 10 images, to test each feature. The classification worked best for the features length and hollow (10/10 classified correctly) and then width (8/10 classified correctly). It performed around chance-level for flower (6/10 classified correctly), violet and rusty body (5/10 classified correctly) and extremely poor for bent (2/10 classified correctly).

DISCUSSION
The results we got for the final PCA approach, of calculating the principal components for each feature separately led to better results, then the first two approaches.

From the images of the first 10 principal components, we can visually assume that there is information about the length and shape stored in the first principal component, as a clear asparagus piece can be seen. The following images leave a lot of room for interpretation, about what information is contained there.

We performed the PCA on each feature separately, to extract the principal components of each feature. It is interesting to see that the pictures of the different features are all very similar. One reason for this might be that many of the 400 pictures for each feature are overlapping between features. Another reason might be, that even though the images vary between features, the general information of all asparagus images is very similar.

From the results of the recognition function, one can see that there are large differences between the features on how well our PCA performs (20% - 100%). One

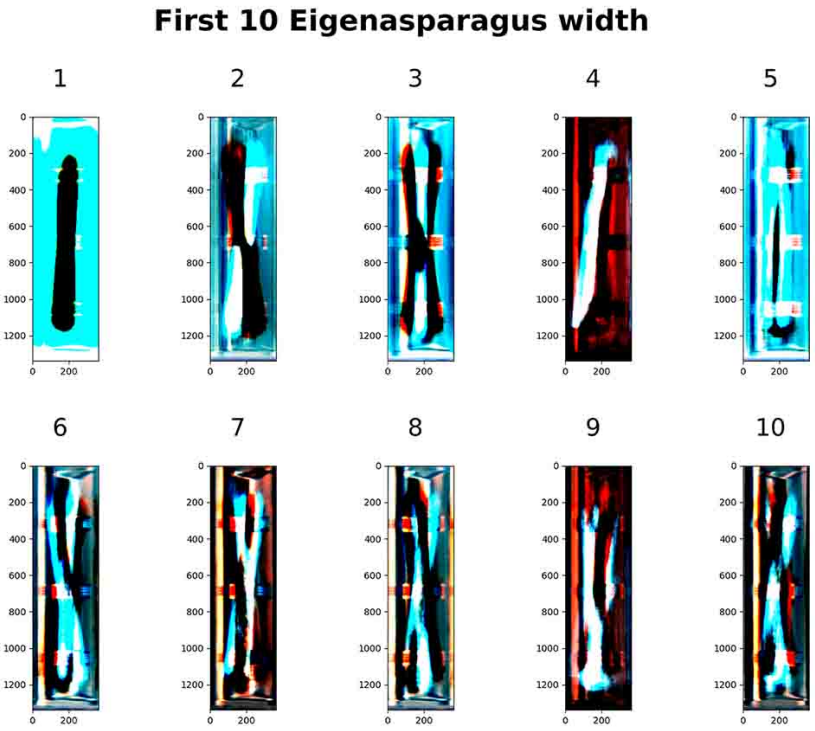



FIGURE 4.20: ??

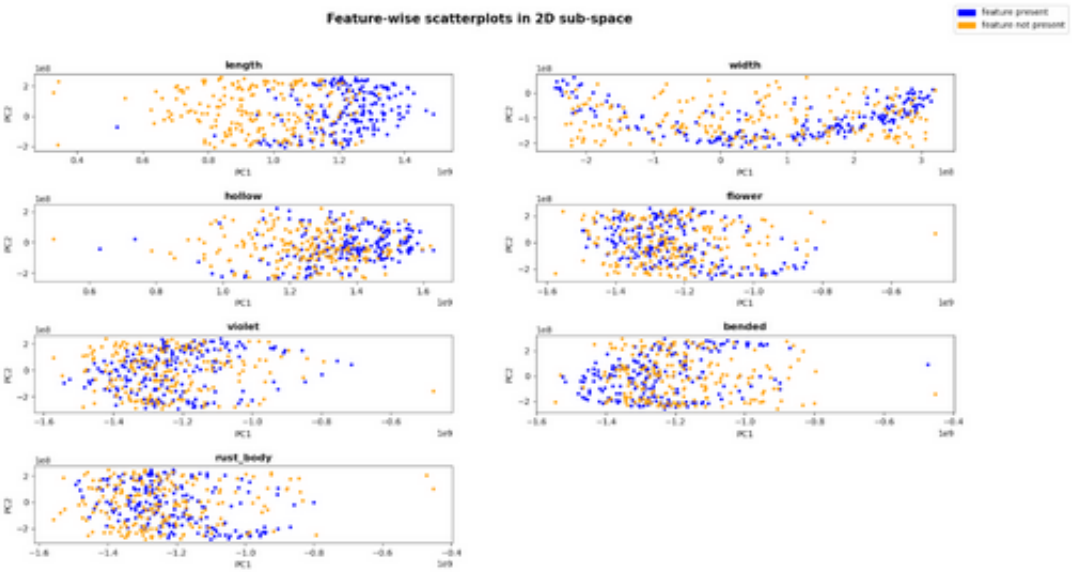



FIGURE 4.21: Plotting the data in the 2D PC space for each feature.

reason for this could be that certain features are simply more difficult to distinguish than others. Another reason for this large variation can be that certain features are also more difficult to label consistently (see also chapter 3.5.4 (agreement measures)), and that the results are due to inconsistencies within the data. One indicator that this is a considerable reason is that the performance of the width and length features, which is information which is not hand-labelled, is very high. Moreover, the poorest results can be observed for the features bent and rusty body. Those are the features, for which the agreement measures showed the largest discrepancies between annotators (see 3.5.4).

Another reason why the results are partly only moderate, is that RGB image data possesses complicated structures, and by representing it in a linear low dimensional feature space, it might be that simply too much information is lost. Even though there are papers reporting good results on PCA on image data (Turk and Pentland, 1991) (Lata et al., 2009), there are other papers claiming that nonlinear dimension reduction algorithms are needed for image data (Olaode, Naghdy, and Todd, 2014).

### 4.2.2 Autoencoder

Beside PCA, there are further techniques for dimensionality reduction. One alternative machine learning technique that can be employed to deduce sparse representations and automatically extract features by learning from examples are autoencoders. Simple autoencoders, where the decoder and encoder consist of multilayer perceptrons, were already proposed as an alternative to PCA in early days of artificial neural networks when computational resources were still comparatively limited (Kramer 1991). Today one can choose from a multitude of network architectures and designs that all have one property in common: A bottleneck layer. For image classification it is common practice to use convolutional autoencoders. There are numerous papers that employed convolutional autoencoders in various domains. Examples range from medical images to aerial radar images (Chen et al 2017). These include not only shallow networks but more recently the benefits of deep autoencoders were demonstrated (Geng et al. 2015). In addition, more complex architectures like ones that combine autoencoders with generative adversarial models were proposed recently (Bao et al. 2017). In many cases the purpose of autoencoders is dimensionality reduction and feature extraction or in other words the learned latent space of the bottleneck neurons. In other cases, autoencoders are used to map images from one domain to another, for example camera recordings to label-images such that after training a labelled image can be retrieved from the decoder's output layer (Iglovikov 2018). In short, there are many possible ways to apply autoencoders and many architectures to realize them.

This motivates the question of how autoencoders work. As mentioned all autoencoders have a bottleneck layer. If applied for dimensionality reduction autoencoders are usually used to predict the input, in this case the image, with the input itself. Autoencoders consist of an encoder that contains the initial layers as well as the bottleneck layer and the decoder that maps the respective latent space back to the image. The desired mapping function of the input to a sparse representation is generated as a side product of the optimization in end to end training, as weights of the decoder are trained such that meaningful features are extracted. The main difference to PCA is that nonlinear functions can be approximated. Feedforward ANNs such as the encoder are non-linear function approximators. Networks with

multiple layers are especially well known for establishing named nonlinear correlation. Hence, autoencoders allow for non-linear mappings to the latent space. This means that in the latter multiple features may be represented in a two dimensional space. It shows that compared to PCA where one dimension typically corresponds to one feature more information can be represented in fewer dimensions. Different properties of the input are mapped to different areas of the latent space. In this case, a convolutional variational autoencoder was used. In variational autoencoders a special loss function is used that ensures features in latent space are mapped to a compact cluster of values. This allows for interpolation between samples by moving on a straight line because regions between points in the latent space lie within the data and hence reconstructions of the decoder are more realistic. Other than that variational autoencoders share most properties with regular autoencoders. The location of a point in latent space refers to a compressed representation of the input.These can be interpreted as features of the input.

Different variational autoencoders were tested in the realm of the project. First, a simple variational autoencoder with a multilayer perceptron as decoder was implemented. The second approach was a comparatively shallow convolutional variational autoencoder. The third approach relates to an autoencoder with a deeper encoder that was later used to design the networks for semi supervised learning. As the third approach did not improve the mapping of the properties of asparagus pieces to a two latent asparagus space the results for the second of named networks are described here. It was derived from a standard example for convolutional variational autoencoders (Keras 2020a). The presented results are for a network that comprises two hidden convolutional layers in the encoder and two deconvolutional layers in the decoder.

Similar to the presented way of applying PCA, batches of images that contain only one perspective were used as input to the network. The downsampled dataset was used. Images had to be padded as the implementation does not work with inputs of arbitrary shape. The deconvolutional layers of the decoder can only increase dimensionality by an integer factor: The filters that were used for the deconvolution in the given net double the tensor dimensionality. For padded images, this means an increase for the vertical dimension from 34 to 68 and finally to the desired 136 pixels was achieved in the last three layers of the network (impossible for the original height of 134 pixels). The input shape which defines the shape of the output layer as well must be divisible by two without remainder twice.
The depiction below shows the results.

It demonstrates that the features short thick and thin are mapped to separable clusters. As a tendency curvature correlates with a region in the lower periphery as indicated especially by the deconstruction. The other features (violet, rust and blooming) are not mapped adequately and they are not visible in the reconstruction. This shows that only some features of interest were mapped to the latent space and used to decode images. Reconstructions of autoencoders are known to miss many details. Better results could potentially have been achieved using larger input images. The possibility to generate, for example, more or less curved asparagus pieces may help to define a clear cut decision boundary and classify images accordingly. As a potential feature for asparagus sorting machines this would allow the user to customize the definition of curvature to his or her own taste. For this approach to be viable for all features, however, the network performance appears to be insufficient. Some features are poorly separated by our network that

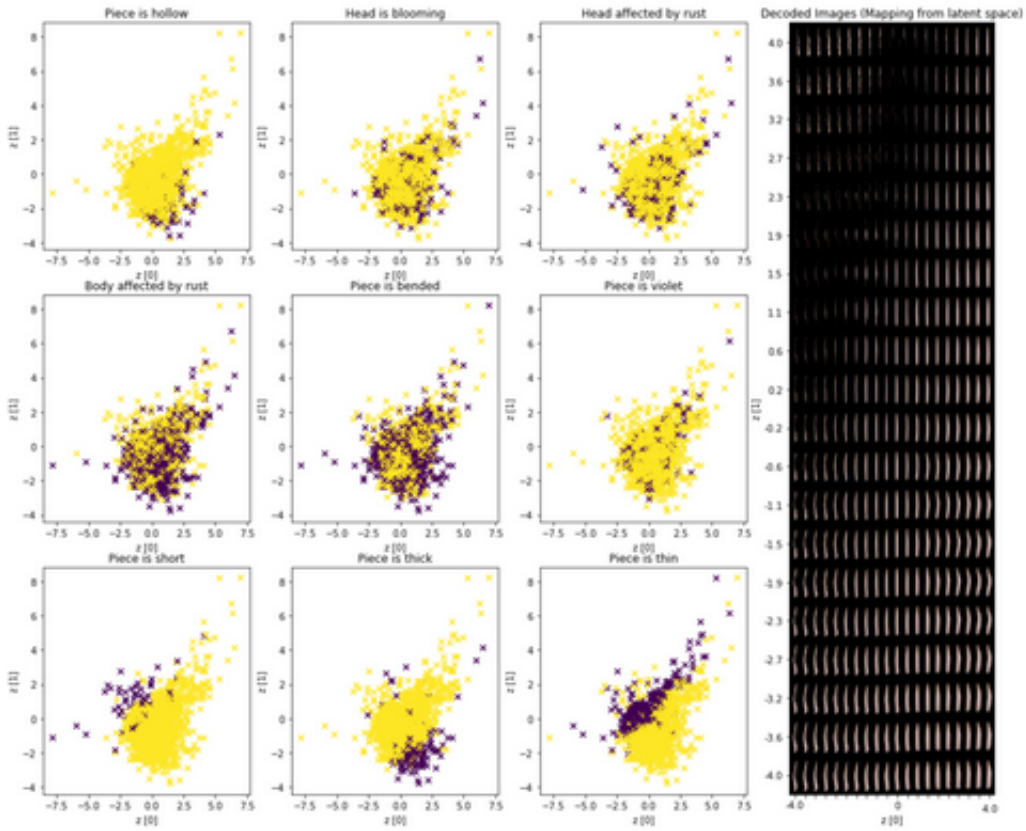



FIGURE 4.22: ??

was employed on downscaled images.

## 4.3 Semi-supervised learning

We collected more than 100.000 samples. Considering the uniform appearance of the images this represents a substantial amount. However, the target categories and information regarding the features present for each peace are unavailable. Labels had to be manually generated which was done for around 10.000 samples. As a consequence, there is only a small subset of data with attributed labels. Smaller amounts of labelled data mean that predictions can be successful only if the variance in the source values is limited. Hence, for high dimensional data such as images, sparse representations are desirable. Extracting features automatically instead of relying on manual feature engineering is a strategy that is especially appealing if large amounts of unlabelled data are available. In semi-supervised learning features are extracted from the main input (here: images) using unsupervised learning if target labels are unavailable (Keng 2017). Hence, semi-supervised learning promises better results by using not only labeled samples but also unlabeled data points of partially labeled datasets.

In the previous chapter, methods and results for unsupervised learning are presented. One example are convolutional autoencoders. In this section it is shown how convolutional autoencoders with additional soft constraints in the loss function can be used for semi supervised learning. Instead of computing another dataset of sparse representations that contains the results of unsupervised methods, in

semi-supervised learning sparse representations are retrieved and mapped to latent features at the same time (Keng 2017). Bottleneck layer activations represent automatically extracted features. For semi-supervised learning one tries to enforce that latent layer activations of autoencoders correlate with the target categories.

The network structure was derived from the convolutional autoencoder used for unsupervised learning. The feedforward CNN was replaced by a network that has proven to be suitable to detect at least some features with sufficient adequacy when trained on heads (see 4.1.2). It comprises three convolutional layers with a kernel size of two in the first and three in subsequent layers as well as 32 kernels each. A max pooling layer with stride two is added mainly to reduce the number of total neurons while maintaining a high number of kernels. Additionally, a dropout layer is added to avoid overfitting. In contrast to other implementations for semi-supervised learning (see Keng 2017) we used the same network for the prediction of labels (when they exist for the current batch) and endcoder for reconstructing images. For the decoder we chose the same one as in the variational autoencoder presented in the previous chapter. The effects of a bypass layer were tested that contains neurons not being subject to the label layer loss (see below). Two variations of the named network were tested: A convolutional variational autoencoder and a variational autoencoder for semi-supervised learning.

A challenge resulted from training with multiple inputs (images and labels). As deep learning frameworks (here we employed Keras) usually require a connected graph that links inputs to outputs (here: images), a trick had to be used. A dummy layer was introduced where all information that stems from the labels was multiplied by zero. The output vector was concatenated with the bottleneck of the encoder. As it contains no variance, training and more importantly validation accuracies remain unaffected even though information about the categories that are predicted is added on the input side. Nevertheless the labels are part of the network graph and could hence be used in the loss function.

A custom conditional loss function was used. If labels are present for the current batch the costum loss is equal to a combined loss that comprises the reconstruction loss and the label loss. Here, reconstruction loss refers to the pixelwise loss that was used for the main task of the network - namely mapping of input images back to the same images (fed into the netowork as target values to the output layer). The label loss is used with the goal of mapping label layer activations to the actual labels. It is low if activations in the sigmoid transformed label layer match the target values i.e. the sum of the error layer is low. The loss that is due to labels was multiplied by a custom factor (k). In addition it was defined such that it scales with the current pixel wise reconstruction loss and converges to a constant (c). These values were chosen aiming for an increase of the contribution of the label loss to the combined loss especially in late stages of training.

The results for the semi-supervised variational autoencoder are illustrated in table (X) and figure X. As one can immediately see, the feature “thin” was adequately mapped to a decisive region in latent space. For the other features, no such clear cut clustering is visible. Reconstructions indicate that the main purpose of the network of predicting the input images was accomplished successfully although the reconstructions look rather uniform. Values for accuracy and sensitivity indicate only poor performance. Only the features “rusty body”(se=.42), “thin”(se=.54) and “bent”(se=.1) sensitivities are above zero.

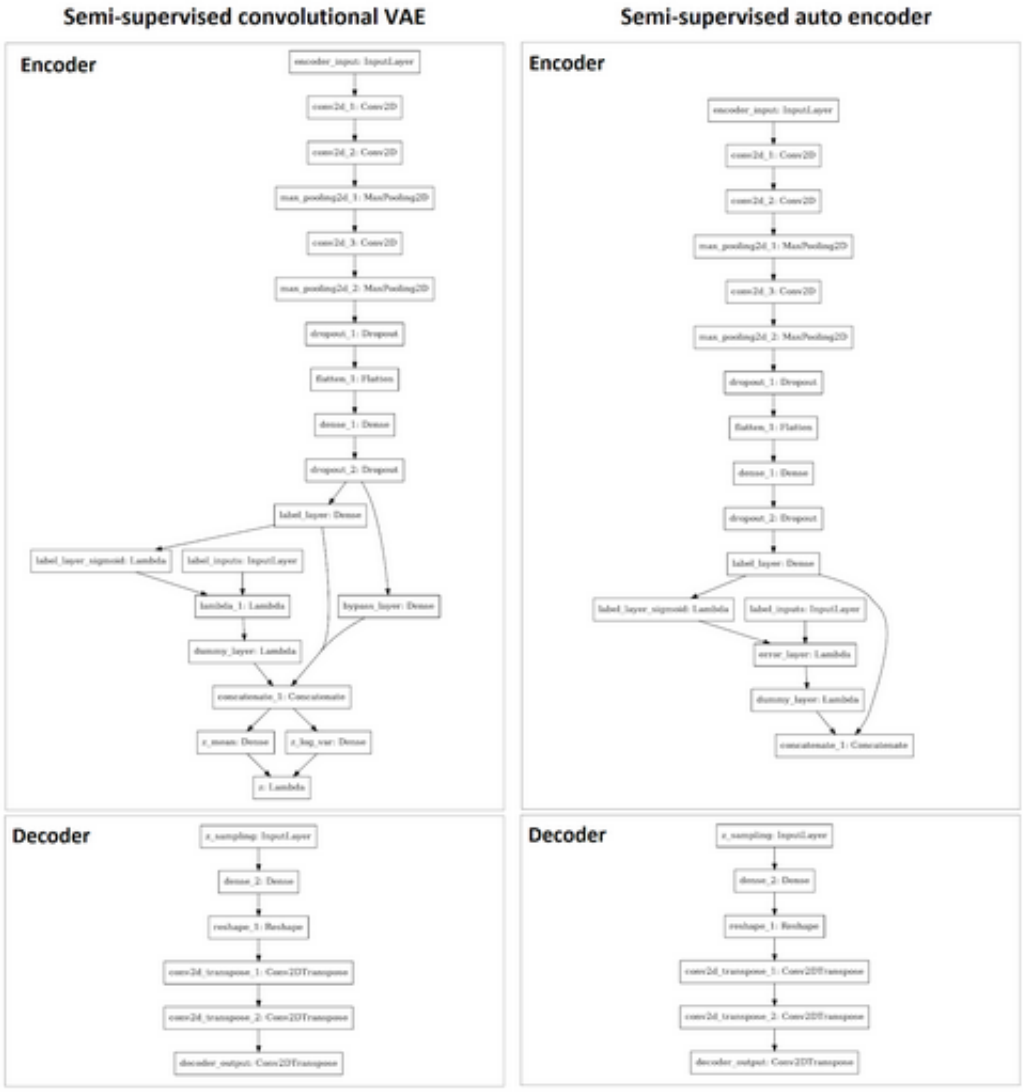



FIGURE 4.23: ??

| | False positive | False negative | True positive | True negative | Sensitivity |
|---|---|---|---|---|---|
| is_bended | 0.04 | 0.37 | 0.04 | 0.55 | 0.10 |
| is_violet | 0.00 | 0.08 | 0.00 | 0.92 | 0.00 |
| has_rost_body | 0.14 | 0.27 | 0.20 | 0.39 | 0.42 |
| short | 0.00 | 0.02 | 0.00 | 0.98 | 0.00 |
| thick | 0.00 | 0.07 | 0.00 | 0.93 | 0.00 |
| thin | 0.00 | 0.14 | 0.16 | 0.70 | 0.54 |

Compared to the variational convolutional autoencoder for semi-supervised learning, the convolutional autoencoder for supervised learning performs better, but performance could be improved further. In table (X) the results are summarized. Violet detection was not successful at all (se=.0). For the other features the network was trained on, mediocre results were achieved. Thickness detection showed little sensitivity (se=.04) however a high specificity of 1.0. Better results exist for curvature (se=.18) “rusty body” (se=.42) and “short” (se=.6). The specificity for the “rusty body” is lower (sp=.6) as compared to named other features (se=1 and

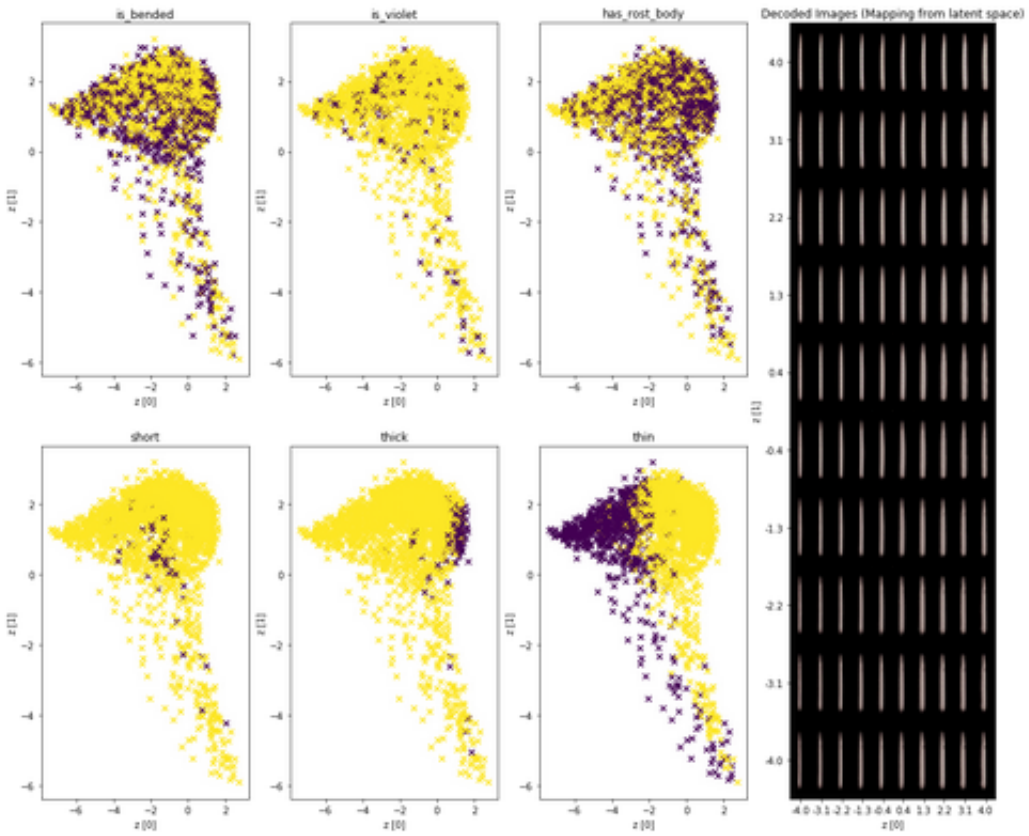



FIGURE 4.24: ??

se=.96). Thin pieces were detected in 76% of all cases and few false positives characterize the detection of named feature (sp=.97).

The approach for semi-supervised learning presented here faces two challenges. First, it should be mentioned that the networks for semi supervised learning were trained only on one perspective although labels were attributed per asparagus piece. Information that is visible on one or two out of three perspectives only can not be mapped to the desired target category. Imagewise labels would be desirable to improve the approach.

Second, reconstructions using convolutional autoencoders contain little detail. However, small details in the image, such as small brown spots, play an important role in the classification by human coders. These features are not sufficiently reconstructed by the variational autoencoder. Arguably they are not reflected in the sparse representation that corresponds to latent layer activations. One may speculate that a substantial increase of the network size would help to reconstruct more details and hence extract more features. As generative adversarial networks are known to generate more detailed images (Bao et al. 2017) they could possibly be adopted for semi supervised learning with greater success. However, this is a question that must be answered empirically.

In summary, one may conclude that autoencoders appear as an alternative to manual feature engineering if a large dataset is available and only a subset contains labelled samples. However, more research is necessary to find best suitable network structures.

| | False positive | False negative | True positive | True negative | Sensitivity |
|---|---|---|---|---|---|
| is_bended | 0.02 | 0.34 | 0.07 | 0.57 | 0.18 |
| is_violet | 0.00 | 0.08 | 0.00 | 0.92 | 0.00 |
| has_rost_body | 0.23 | 0.19 | 0.28 | 0.30 | 0.59 |
| short | 0.00 | 0.01 | 0.01 | 0.97 | 0.57 |
| thick | 0.00 | 0.06 | 0.00 | 0.93 | 0.04 |
| thin | 0.02 | 0.07 | 0.23 | 0.68 | 0.76 |

# Chapter 5

# Summary of classification results

The current work was conducted to investigate the state of the art on asparagus classification and to improve the current sorting algorithm implemented at the XX machine at the Spargelhof Gut Holsterfeld. Data was collected, preprocessed and then analyzed with seven different approaches. Out of our XX (all images) images, (XX) were re-sorted in the machine and 13319 images manually labeled by us. Those labelled data was considered for the supervised approaches. The semi-supervised approach was additionally based on (XX) unlabeled images. The results illustrate that classifying asparagus is not a trivial problem. This is partly due to the small differences between groups, and vague borders within features. However, the results also show that it is possible to extract relevant features and to improve current sorting approaches.

As supervised learning approaches, a MLP and a CNN are introduced for feature engineering, whereas the CNN only includes head related features. Following, single-label classification and multi-label classification are conducted. All these attempts aim to solve the same image classification problem, using supervised learning. The MLP, as well as the CNN approach for head feature extraction were based on automated extracted labels, generated through computer based feature extraction, whereas all other approaches used the hand-labelled data.

As the feature engineering MLP (mostly) and the single-label-classification CNN perform a binary classification (see 4.1.1, 4.1.3) and the CNN for head-related features as well as the multi-label approach perform a multi-label classification (chapter 4.1.2, 4.1.4), results are difficult to compare directly. With the random forest approach a third option, multiclass prediction is added. Multi-label classification is a more complex, holistic approach. Still, the results of the binary classification approaches yield a deeper insight on where difficulties lie and what features are more problematic than others. The multiclass approach had an accuracy of 75%. The accuracy for the CNN for head-related features was XX%. Especially flower reached 55% sensitivity, 95% specificity while rusty head attained 19% sensitivity and 98% specificity. The accuracy of the Multi-label approach reached up to 87%. For this model, accuracies werewhere not separated, to see accuracy feature wise.

The accuracies for binary classification can be stated more specific: The feature engineering approach showed best results on curvature (82% sensitivity, 67% specificity) and worst results for rusty body (71% sensitivity, 65% specificity). In the single-label classification CNN, best results were achieved for the feature XX with a sensitivity of XX and a specificity of XX, while worst results were achieved for XX with a sensitivity of XX and a specificity of XX. Train accuracy did not notably exceed 50%. The test accuracy usually increased up to XX - XX.

The unsupervised learning approaches, one is referred to as PCA, the other as

VAE, deal both with dimension reduction. Both were implemented, using the hand labelled data. Whereas the classification method of the PCA is based on binary feature prediction (absence or presence of a feature), the VAE does not predict labels. The accuracy of the PCA was promising for length and hollow (100%) but extremely poor for bent (20%). The accuracy of the VAE ...
The last approach, a semi-supervised learning method was based on a partially labeled dataset. It performed multi-label classification. Unfortunately the predicted power was rather poor, best for curvature (18% sensitivity, 96% specificity) and worst for violet as it did not detect any violet pieces (0% sensitivity).

Evaluation

# Chapter 6

# Discussion

In our study project we pursued three main objectives. The main goal was to develop and improve algorithms for fruit classification. The second objective is closely linked to this and relates to best practices in relation to applied data science and big data. This included storage of data on remote servers and computationally expensive procedures that are required for training in the computational grid of Osnabrück University: The methodological aspect of our study project. As our work also served as a sample project to learn more about possibilities to effectively organize collaborative work we also targeted at the organizational aspect that is closely linked to project management. In the following, the core results with respect to named objectives are shortly named and discussed.

## 6.1 Discussion of classification results

Asparagus pieces have several features that we aimed to extract. Some features such as the length and width of asparagus pieces were undoubtedly measurable using a pure computer vision approach that does rely on machine learning. For others direct filtering was not easily possible. Arguably, there are two reasons. First a clear cut definition of features such as “curvature” or “violet” that is precise enough to directly be implemented does not exist. Although relevant information can easily be extracted, the rules to infer the desired binary features are inaccessible: On one side, decision boundaries for binary classifiers have to be found while the perception of the features colour or curvature, on the other, has shown to be very subjective. Second, filtering features such as “flowering head” has proven difficult. Named attribute relates to details in a few pixels, comes in different forms and highly depends on the perspective. Because of both reasons machine learning must be employed to successfully classify asparagus.
We designed several neural networks and applied them in different ways to analyze and classify a large-scale asparagus image dataset. Some approaches worked better than others. The maximum accuracy we reached is X%, which was reached by model X.

Training MLP classifiers on histograms of palette images has proven a promising approach to predict colour features. Named histograms contain information about the fraction of foreground pixels that correspond to violet or rusty colours. As MLPs have few parameters the design is rather trivial and the training process quick. The results are considered as a good baseline performance for “rusty body” and “violet”. The application on palette images that show asparagus heads only might also be suitable to predict the feature “rusty head”. This is yet to be tested. MLPs have also shown to be suitable when applied to predict the binary curvature feature based on partial angles. The fact that predictions are far from perfect is arguably due to inconsistencies in the training data. One may assume, however, that

the models generalized well and represent rules that relate to average opinions or definitions of highly subjective features such as color or curvature.

Feedforward CNNs were applied to predict individual features, for multilabel prediction and predictions based on snippets that depict asparagus heads. In addition, effects of a custom loss function were tested. Promising in the multilabel prediction is that not only individual features, but also the relation between features can be considered in the learning process. Our multilabel CNN reaches an accuracy up to 87%, which seems high. However, when looking at the accuracy and loss values over time, one can see that the model does not learn much. While sensitivity and specificity improve, and therefore indicate learning, the validation loss remains high, indicating overfitting. This model seems to be especially sensitive to the imbalance between 0 and 1 in the label vectors. Concerning this, there is still room to play around with our parameters to further improve the architecture.

Applying PCA on individual features and projecting the image information into a lower dimensional subspace showed promising results, too. It really seemed that the first principal components managed to capture most of the information. However, differences between features seem to be too small, to reliably distinguish all features. In this approach, the features width, length and hollow seem to be classifiable with high performance, and the features bent and rusty body seem to be most difficult.

Similarly variational autoencoders were used to derive a low dimensional representation using unsupervised learning. While some features such as the width and length were mapped to clearly differentiable regions in latent asparagus space this was not the case for many others. Only as a tendency, spears labelled as “bent” are for example mapped to regions in the lower periphery. Autoencoders are known for blurry reconstructions. This is a possible explanation for the lack of clusters in latent space for features that relate to details that were not sufficiently reconstructed.

Convolutional autoencoders were used for semi-supervised learning. However the results could be described as mediocre only. One problem is arguably the mentioned insufficiency in reconstructing details. As details such as brown spots define target classes (e.g. “rusty head”) and they are not present in latent space. It is hard to establish a correlation of the respective latent layer activation and the target labels. Larger input image sizes or different network architectures that are suitable to reconstruct higher detail images could potentially help to improve performance of semi supervised learning.

Detecting the feature “rust head” has proven rather difficult even though a dedicated network was trained on snippets that show asparagus heads in rather high resolution. This is potentially the case because details (here: small brown spots) that are hardly visible even to the human eye have to be considered that occur in different locations. Although better results were achieved for flowering heads this arguably also holds for this category. In contrast better results were achieved for features that relate to the overall shape of asparagus pieces instead of fine details. Arguably this holds for the category “hallow” but e.g. also for curvature. Colour features are detected especially well based on histograms of palette images while convolutional networks have proven suitable to detect shape related features.

In summary we successfully measured the width and height of asparagus pieces and were able to develop detectors for the other features that performed surprisingly good, given the low inter-coder reliability that was arguably due to unclear definitions of binary features such as “bent” or “violet.”

## 6.2 Methodological discussion

Looking back, there are methodological issues, which we would do differently now. We started our study project at the beginning of April. This was at the same time, as the asparagus harvesting season started. Positive about this was that we knew that we could start collecting data straight away. Negative about it was that we had to start collecting data without a detailed plan in advance. This could have made the data acquisition more efficient, and structured. We did not have enough time to clarify some important things in advance, such as: How much data do we need, what format do we need it in? Is autonomous calibration possible? How exactly do we store the images effectively and efficiently? Is the setup as we need it - compare different ones and decide: How can we improve the setup? What kind of measurements or changes do we want to perform with the camera? Is the illumination as we want it? Could stereo cameras or other 3D view techniques such as depth cameras or latergitter be used? How should our pipeline look? How can we get labelled data right away?

Another discussion point concerns our data. The image quality in terms of pixel size of our images was really high. Also, the illumination was fairly consistent during the harvesting season. However, we did not investigate in trying to improve the illumination by more LEDs. This could have improved the amount of reflections, which can be seen on some images.

Even though we have three images of every asparagus spear, they are all taken from the same perspective – from above. In the ideal case, the asparagus spear rotates over the assembly line, and we can see each spear from a different viewpoint. The more the asparagus is rotated, the more reliable a later judgement of the spear in terms of class or features is possible. As the rotation often does not occur, because the spear is too bended, an additional viewpoint could improve the rating. This could be through an additional camera from the top, or an additional camera taking an image from the bottom of the spear (see also chapter XX).

As already previously mentioned, our labels of asparagus features were partly achieved by computer-vision algorithms, partly based on human perception. As previously outlined, human performance is commonly acknowledged as the baseline performance in classification tasks. While the performance of our automatic feature-extraction for length and width is really high, we decided that for the features violet, rusty head, rusty body, curvature, hollow and blooming a human perception would be more accurate. Even though this is commonly used as the “gold standard”, it can of course also bring more variation, and maybe even inconsistency between raters, than an algorithm (see also 3.3).

Also, we decided to keep the features binary, as this was easier to label, and easier to use for our supervised classification approaches. The down side of a binary label is however, that a clear boundary is set, where in real life there is a smooth transition. Even for our supervising farmer, it was sometimes not clear to set the

boundary – especially for the feature bent, this was difficult. While this makes certain analyses and classification much easier, it also brings certain restrictions.

Moreover, we also observed difficulties concerning the labels in the communication between us and the farmer. The communicated need is that the sorting algorithm works "better". But what does that technically mean? And what is technically possible? For him, the sorting would already be "better", if the sorting mistakes would be more systematic. This would not necessarily mean that the overall accuracy of correctly sorted asparagus into 1 out of 13 classes needs to improve, but that the overall impression of all spears sorted into one class is more homogeneous.

In connection to the previous difficulty, Mr. Schulze-Weddige mentioned a wish that there would always be the same amount of asparagus spears in the first class. In this case, it would be accompanied by the fact that we need a smooth transition for several features, so that we could sort e.g. the only little bent spears on some days into the first class, on other days into the second class. One idea we had for this was to implement a Bayesian classifier into our deep learning approaches, but due to time constraints we did not pursue this idea.

Different models ran with jpg or png files of our initial images. All color analysis was performed in the RGB color space. In the literature it is found that some authors (e.g. Bhargave 2018) use different color spaces for color-based feature extraction. Exploring other color spaces could potentially also improve our violet and rust detection.

## 6.3 Organizational results

- still in progress -

During the course of the study project, the team did not only pursue the objective of learning and implementing computer vision based approaches for a practical application in asparagus classification but also grew in respect to soft-skills like teamwork and project management.
In the following text, it is summarized and briefly discussed what each member has personally learned during the year on an organizational level.

On a more practical level, programming skills could be further developed together and a larger focus was set on clean and comprehensive code. Additional scripts and programs were introduced like Github or pages as read the docs. For better communication, it was indispensable to write code or text that can be understood by many people as well as formulate tasks and working goals in a clear and open way. Every member of the group had the responsibility to explain not only his or her views but also learn to listen to the ideas of others. Team discussions became a valuable basis for exchanging and learning from each other. It was understood that there is no point in closing yourself off from the group and minding your own business. Also, updating your team members is important and saves more than time, frustraction, and the double-distribution of tasks. Other practical skills comprise giving understandable presentations and holding productive meetings where goal-oriented questions are discussed. Working together towards a goal was also a good training ground for assessing how teamwork and task distribution are done in research groups at companies or in the scientific fields.

When evaluating the team dynamics through the year, especially the time management and the sharing of responsibilities were emphasized. It could be trained how a team meeting is held effectively and productive. Further, each member came at least once to their frustration limits. Mistakes had to be owned to be able to progress towards the shared goal. It had to be learned to deal with frustration and drawbacks as well as practicing open and constructive criticism. Again, communication was key when the team was struggling to build each other up and help each other out. It is also important to value the effort that your team members put into their task, take the time to listen to what they did and help with constructive feedback.

For the next project, it was agreed that the working environment had to be stricter and more focused. Everyone needs to know what he or she has to do and it has to be made sure by everyone. No one should be left behind. In turn, taking the initiative and also more responsibility were points we would have done differently if we could start anew. Delegating works and having an overview were further points that were considered worth of improvement for the next teamwork challenge. Democratic decision making does not necessarily have to exclude the role of a team leader or manager who has an overview of the tasks, can delegate work and, thus, enable faster decision-making. Formulation of concrete goals, overcoming fears of failing and perfectionism, more working together on tasks. ...
Analyse your team members. Evalue which strengths everyone can contribute to the team. Not everyone has to do the task he or she is best at but also tasks that are new, challenging and interesting to allow for growth. Do not only choose the tasks you are comfortable with but take the challenge. Further, include everyone into the team by sharing responsibilities and distributing roles. Think more about the bigger goal than of what lies directly ahead. Take more time into consideration when planning ahead. Usually, finishing work takes longer than previously planned.

In summary, we can say that we have not only learned new scientific skills and techniques of data acquisition, preparation, and analysis but also gained valuable new insights into the organization of a large project with many members. We understood how to organize ourselves more successfully and purposefully. First and foremost, we learned that this needs excellent communication. The whole team agrees that we would structure the next project in the same way as we did in the second half of our project. Communication is important, but it was further discovered that there is a right way to balance the amount and effectiveness of communication. Not everyone has to discuss or listen to all the details in every area, often it suffices when all team members have a broad overview. Additionally, it turned out to be helpful when one or two members exchange some of the task-related work in favour of more management-related work. This helps to gain a better team structure, time management and, in the end, make the team members work more effectively and efficiently towards the goal.
The experience of having two different working structures gave us the ability to compare and judge what is essential to successful teamwork. It also helped to understand how each member contributes to the team regarding personal skills and interests, and what each member can improve for future teamwork. As the main intention of the study project was understood as a learning unit, we wanted to seek out tasks that we were motivated to do and that brought us new experiences and skills, and not just tasks and practice what we already knew.

# Chapter 7

# Conclusion

## 7.1 Summary

It has been possible to develop various modern computer vision techniques with the aim of classifying asparagus pieces according to the quality of the asparagus pieces and to examine their practicability in commercial applications. The specifications of the project, which were given in the introduction, were taken into account in the planning and successfully implemented. By implementing different approaches it is possible to classify asparagus in dependence of the quality. After a time-consuming collection of data, laborious labelling and classical pre-processing steps, we applied different procedures. Different approaches were implemented: supervised learning, unsupervised learning and semi-supervised learning. As explained in chapter five - result discution - .

In conclusion, we can affirm that modern approaches from computer vision and machine learning can (not) improve sorting. We have shown that the algorithms can also be used (not) only for scientific purposes but also in industrial applications, such as the asparagus sorting problem.

Whether we have succeeded in improving the currently running sorting algorithm has not yet been irrefutably clarified, because we lack a comparison of suitable evaluation. In cooperation with the local asparagus farmer and the manufacturer of his asparagus sorting machine, a method can now be developed. However, we have proved that Computer Vision and Machine Learning are able to solve the problem, so the requirements of the project are fulfilled.

The intention of the study project (Universität Osnabrück, 2019b) (Universität Osnabrück, 2019a) has been followed up. Good self-organisation and team building became the formula for the progress of the project. We have learned to organize ourselves, to work as a team, to put theoretical knowledge into practice, to develop software, to analyze data, to practice the analysis and interpretation of statistical data under the conditions of a common research project.

## 7.2 Outlook of the project

After this project was discussed and summarized at previous chapters, this chapter gives an outlook on future steps. This will reach from setup, to the control of the system, the concept of a continuous data stream, a runtime optimization, other approaches with more complex requirements up to an evaluation procedure.

Due to the harvesting season at the beginning of the project period, the setup was not up for discussion. However, many improvements were discussed among the

manufacturer of the machine. A second camera to capture the head of the asparagus is particularly evident, also shown by our results. This should help to classify the properties "rust" and "flower". Accordingly, we would have to adapt our preprocessing and our pipeline.

In order to put the project into practice, a control system for the machine must be implemented. For this purpose, a cooperation with the manufacturing company must be arranged. After we have delivered results in the project, this can be the next step in the near future.

If the sorting machine follows our algorithms it will be possible to create an evaluation. We have already discussed the difficulties that exist. One method is to measure and compare the sorting of the harvesters. Other methods of evaluation can be considered, if necessary, the setup for automated procedures can be extended.

Furthermore, it is possible to continuously collect more data and use it for the training. An automated implementation, running either locally on the PC of the machine is conceivable. Thus one would be in the area of mobile application of neural networks. A merging of the different networks of different farms would be conceivable. On the other hand, the data could be collected from farms with a serial internet connection and the network could be updated manually and locally on a regular basis.

Another claim that we have not yet addressed is that the asparagus harvest also varies during the harvest season. For example, upon request, asparagus should be placed in a higher category than normal after an unfavourable weather situation. In order to meet this requirement, we need either manual adjusting screws, which is impossible with neural networks in their current form, or a different form. Since we introduce a temporal, sequential component, it may be worthwhile to consider LSTMs in combination with CNNs. Also the literature search shows that there are still many wide possibilities, for example ....

# Appendix A

# Appendix A

## A.1 Task list

TODO: Below the name of each project member, the tasks that were contributed by the member are listed.

**Katharina**

- ...

**Sophia**

- ...

**Richard**

- ...

**Maren**

- ...

**Josefine**

- ...

**Malin**

- ...

**Michael**

- ...

## A.2 Report list

TODO: Here, an overview of the report is given in chapters and subchapters. The name behind each chapter indicates who wrote the respective section.

1. Introduction - *Josefine*
   (a) The project - *Josefine*
   (b) Background on computer vision based classification tasks - *Sophia*
   (c) Background on sorting asparagus - *Josefine*
   (d) Expected outcome vs. actual outcome - *Malin*

2. Data acquisition and organization - *Josefine*
    (a) Roadmap and timetable of the project - *Josefine*
    (b) Organization of the study group - *Richard*
        i. Communication - *Richard*
        ii. Teamwork - *Richard*
    (c) Data collection - *Josefine, Richard*
    (d) Literature Research - *Josefine*
3. The Dataset - *Josefine*
    (a) Preprocessing - *Sophia*
    (b) Automatic feature extraction - *Sophia*
        i. Length - *Sophia*
        ii. Width - *Sophia*
        iii. Rust - *Sophia*
        iv. Violet - *Michael*
        v. Curvature - *Michael*
        vi. Flower - *Sophia*
    (c) The hand-label app: A GUI for labelling asparagus - *Michael*
    (d) Manual labelling - *Josefine*
        i. Sorting criteria - *Josefine*
        ii. Sorting outcome - *Josefine*
        iii. Agreement measures - *Malin*
        iv. Reliability - *Malin*
    (e) The asparagus dataset - *Richard*
        i. Different datasets - *Sophia, Richard*
        ii. Challenges - *Sophia, Richard*
4. Classification - *Malin*
    (a) Supervised learning - *Josefine*
        i. Prediction based on feature engineering - *Michael*
        ii. A dedicated network for head-related features - *Michael*
        iii. Single-label classification - *Josefine*
        iv. Multi-label classification - *Sophia*
        v. From feature to label - *Sophia*
    (b) Unsupervised learning - *Malin*
        i. Principal component analysis (PCA) - *Malin*
        ii. Variational autoencoder - *Michael*
    (c) Semi-supervised learning - *Michael*
5. Discussion
    (a) Comparison of classification approaches
    (b) Discussion of classification approaches
    (c) Methodological discussion - *Malin*
    (d) Organizational discussion

6. Conclusion
    (a) Summary
    (b) Next steps

# Appendix B

# Appendix B

TODO: Additional information will be given here.

# Bibliography


Al Ohali, Yousef (2011). "Computer vision based date fruit grading system: Design and implementation". In: *Journal of King Saud University-Computer and Information Sciences* 23.1, pp. 29–36.

Al-Rawi, Mohammed and Dimosthenis Karatzas (2018). "On the Labeling Correctness in Computer Vision Datasets." In: *IAL@ PKDD/ECML*, pp. 1–23.

Balaban, Stephen (2015). "Deep learning and face recognition: The state of the art". In: *Biometric and Surveillance Technology for Human and Activity Identification XII*. Vol. 9457. International Society for Optics and Photonics, 94570B.

Batista, Gustavo EAPA, Ronaldo C Prati, and Maria Carolina Monard (2004). "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD explorations newsletter* 6.1, pp. 20–29.

Bengio, Yoshua (2012). "Practical recommendations for gradient-based training of deep architectures". In: *Neural networks: Tricks of the trade*. Springer, pp. 437–478.

Bhargava, Anuja and Atul Bansal (2018). "Fruits and vegetables quality evaluation using computer vision: A review". In: *Journal of King Saud University-Computer and Information Sciences*.

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. springer.

Breiman, Leo (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Brosnan, Tadhg and Da-Wen Sun (2002). "Inspection and grading of agricultural and food products by computer vision systems—a review". In: *Computers and electronics in agriculture* 36.2-3, pp. 193–213.

Caruana, Rich and Alexandru Niculescu-Mizil (2006). "An Empirical Comparison of Supervised Learning Algorithms". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 161–168. ISBN: 1595933832. DOI: 10.1145/1143844.1143865. URL: https://doi.org/10.1145/1143844.1143865.

Cohen, Jacob (1960). "A coefficient of agreement for nominal scales". In: *Educational and psychological measurement* 20.1, pp. 37–46.

Daumé III, Hal (2012). "A course in machine learning". In: *Publisher, ciml. info* 5, p. 69.

Diaz, R et al. (2004). "Comparison of three algorithms in the classification of table olives by means of computer vision". In: *Journal of Food Engineering* 61.1, pp. 101–107.

Donis-González, Irwin R and Daniel E Guyer (2016). "Classification of processing asparagus sections using color images". In: *Computers and Electronics in Agriculture* 127, pp. 236–241.

Europäische Komission (1999). "Verordnung (EG) Nr. 2377/1999 der Kommission vom 9. November 1999 zur Festsetzung der Vermarktungsnorm für Spargel".

In: *Amtsblatt der Europäischen Gemeinschaften, OJ L 287, 10.11.1999, p. 6–11*. URL: https://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:31999R2377&from=DE.

Feinstein, Alvan R and Domenic V Cicchetti (1990). "High agreement but low kappa: I. The problems of two paradoxes". In: *Journal of clinical epidemiology* 43.6, pp. 543–549.

Figueroa, Rosa L et al. (2012). "Predicting sample size required for classification performance". In: *BMC medical informatics and decision making* 12.1, p. 8.

Géron, Aurélien (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

Gibaja, Eva and Sebastian Ventura (Apr. 2015). "A Tutorial on Multi-Label Learning". In: *ACM Computing Surveys* 47. DOI: 10.1145/2716262.

Gilpin, Leilani H et al. (2018). "Explaining explanations: An overview of interpretability of machine learning". In: *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, pp. 80–89.

Har-Peled, Sariel, Dan Roth, and Dav Zimak (2003). "Constraint classification for multiclass classification and ranking". In: *Advances in neural information processing systems*, pp. 809–816.

He, Haibo and Edwardo A Garcia (2009). "Learning from imbalanced data". In: *IEEE Transactions on knowledge and data engineering* 21.9, pp. 1263–1284.

He, Kaiming et al. (2016a). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

— (2016b). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Heaton, Jeff (2015). *AIFH, Volume 3: Deep Learning and Neural Networks*.

HMF Hermeler Maschinenbau GmbH (2015). "Bedienungsanleitung Automatische Spargelsortiermaschine Autoselect". In: URL: www.hmf-hermeler.de.

Huang, Gao et al. (2017). "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

Hubel, David H and Torsten N Wiesel (1962). "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of physiology* 160.1, pp. 106–154.

Kılıç, Kıvanç et al. (2007). "A classification system for beans using computer vision system and artificial neural networks". In: *Journal of Food Engineering* 78.3, pp. 897–904.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012a). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

— (2012b). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Lata, Prof et al. (Apr. 2009). "Facial recognition using eigenfaces by PCA". In: *SHORT PAPER International Journal of Recent Trends in Engineering* 1.

LeCun, Yann, Yoshua Bengio, et al. (1995). "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10, p. 1995.

LeCun, Yann A et al. (2012). "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, pp. 9–48.

Lin, Min, Qiang Chen, and Shuicheng Yan (2013). "Network in network". In: *arXiv preprint arXiv:1312.4400*.

Luo, M Ronnier et al. (2000). "A review of chromatic adaptation transforms". In: *Review of Progress in Coloration and Related Topics* 30, pp. 77–92.

McHugh, Mary L (2012). "Interrater reliability: the kappa statistic". In: *Biochemia medica: Biochemia medica* 22.3, pp. 276–282.

Mery, Domingo, Franco Pedreschi, and Alvaro Soto (2013). "Automated design of a computer vision system for visual food quality evaluation". In: *Food and Bioprocess Technology* 6.8, pp. 2093–2108.

Mirończuk, Marcin Michał and Jarosław Protasiewicz (2018). "A recent overview of the state-of-the-art elements of text classification". In: *Expert Systems with Applications* 106, pp. 36–54.

Olaode, Abass, Golshah Naghdy, and Catherine Todd (Sept. 2014). "Unsupervised Classification of Images: A Review". In: *International Journal of Image Processing* 8, pp. 2014–325.

Olivier, Chapelle, Scholkopf Bernhard, and Zien Alexander (2006). "Semi-supervised learning". In: *IEEE Transactions on Neural Networks*. Vol. 20. 3, pp. 542–542.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2012). "Understanding the exploding gradient problem". In: *CoRR, abs/1211.5063* 2, p. 417.

Pedreschi, Franco, Domingo Mery, and Thierry Marique (2016). "Grading of potatoes". In: *Computer vision technology for food quality evaluation*. Elsevier, pp. 369–382.

Powers, David MW (2012). "The problem with kappa". In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 345–355.

Prakash, J Suriya et al. (2012). "Multi class Support Vector Machines classifier for machine vision application". In: *2012 International Conference on Machine Vision and Image Processing (MVIP)*. IEEE, pp. 197–199.

Reeves, Adam (1981). "Metacontrast in hue substitution". In: *Vision Research* 21.6, pp. 907–912.

Russakovsky, Olga and Li Fei-Fei (2010). "Attribute learning in large-scale datasets". In: *European Conference on Computer Vision*. Springer, pp. 1–14.

Russakovsky, Olga et al. (2013). "Detecting avocados to zucchinis: what have we done, and where are we going?" In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2064–2071.

Russakovsky, Olga et al. (2015). "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3, pp. 211–252.

Sabour, Sara, Nicholas Frosst, and Geoffrey E Hinton (2017). "Dynamic routing between capsules". In: *Advances in neural information processing systems*, pp. 3856–3866.

Sim, Julius and Chris C Wright (2005). "The kappa statistic in reliability studies: use, interpretation, and sample size requirements". In: *Physical therapy* 85.3, pp. 257–268.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

Stallkamp, Johannes et al. (2011). "The German traffic sign recognition benchmark: a multi-class classification competition". In: *The 2011 international joint conference on neural networks*. IEEE, pp. 1453–1460.

Statistisches Bundesamt (Destatis) (2017). "Gemüseerhebung 2016 - Anbau und Ernte von Gemüse und Erdbeeren". In: URL: https://www.destatis.de/GPStatistik/receive/DEHeft_heft_00070882.

Szegedy, Christian et al. (2015). "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Tjoa, Erico and Cuntai Guan (2019). "A survey on explainable artificial intelligence (xai): Towards medical xai". In: *arXiv preprint arXiv:1907.07374*.

Turk, Matthew and Alex Pentland (1991). "Face recognition using eigenfaces". In: *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pp. 586–587.

United Nations Economic Commission for Europe (UNECE) (2017). "UNECE STANDARD FFV-04 concerning the marketing and commercial quality control of asparagus". In: URL: https://www.unece.org/trade/agr/standard/fresh/ffv-standardse.html.

Universität Osnabrück, Fachbereich Humanwissenschaften (2019a). "Modulbeschreibungen Cognitive Science". In: URL: https://www.uni-osnabrueck.de/studium/im_studium/zugangs_zulassungs_und_pruefungsordnungen/fach_master/cognitive_science_msc.html.

— (2019b). "Prüfungsordnung für den Master Studiengang Cognitive Science". In: URL: https://www.uni-osnabrueck.de/studium/im_studium/zugangs_zulassungs_und_pruefungsordnungen/fach_master/cognitive_science_msc.html.

Weber, Karola and Katrin Quinckhardt (2018). "Heimvorteil Spargel - Selbst angebaut, selbst zubereitet! Anbautipps und Rezepte für Spargel". In: URL: https://www.landwirtschaftskammer.de/verbraucher/rezepte/spargelrezepte.pdf.

Zhang, Xiangyu et al. (2015). "Accelerating very deep convolutional networks for classification and detection". In: *IEEE transactions on pattern analysis and machine intelligence* 38.10, pp. 1943–1955.

Zhang, Yudong and Lenan Wu (2012). "Classification of fruits using computer vision and a multiclass support vector machine". In: *sensors* 12.9, pp. 12489–12505.

Zheng, Alice and Amanda Casari (2018). *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc."

Zhu, Xiaojin (2005). *Semi-Supervised Learning Literature Survey*. Tech. rep. 1530. Computer Sciences, University of Wisconsin-Madison. URL: http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.