EE5571 Workshop
Embedded Systems Engineering
IoT Applications Prototyping

Distributed Computing Systems Engineering

Assignment

Group CropTomato

Kevin Linsmeier (1945959)
Project Leader, Data Analyst, App Developer, Documentation

Vanessa Richter (1950018)
Scientific Research, Documentation

Supervisor: Dionysios Satikidis, MSc

Date: 24 April 2020

Brunel University London

School of Engineering and Design
Electronic and Computer Engineering

Links

- **CropTomato on GitHub**
  https://github.com/CropTomato/Smartphone-Sensing-Framework/tree/CropTomato%40SSF2.2_ExperimentalTensorFlow/
- **Hackster.io blog entry**
  https://www.hackster.io/336763/croptomato-5d64a7
- **Prototype demo video**
  https://www.youtube.com/watch?v=Zdx08Odrx4s

Contents

List of Figures

List of Tables

Listings

Abstract

Fruit picking is a very time consuming and difficult task for both, private and commercial gardeners and farmers. The fruit state and especially its optimal ripening stage needs to be determined to ensure the exact timing of harvesting or letting the fruit remain on the plant for longer. There are many technical assistants such as, for example, harvesting robots with a ripeness recognition and localisation system available on the market [1] but such a solution is a big investment and might only benefit commercial agriculture but not private farmers. Although the IT industry has provided several technological solutions for the assistance and support in the agricultural sector, there remain many open problems to be solved through technology.

The disability of colour blindness as a widespread phenomenon in today's society [2] requires a suitable solution for technical assistance and support in identifying the actual colour of an object in nearly every area of affected people's daily lives.

This work faces the problem of the disability to distinguish between the green and red colour when being colour-blind and hence, being not able to determine the ripeness state of fruits. An easy to use tomato ripeness recognition solution is developed to be used in an effective and cheap way to deliver reliable results for colour-blind users.

Within the scope of this work, a supporting application for colour-blind private gardeners and farmers, with the option of being customised to other user groups as well, is designed under the consideration of the technical requirements usability, reliability and customisability.

The proposed solution uses a neural network and its classification method to determine the ripening stage of tomatoes. Through the integration of the neural network into an Android application, the smartphone camera can be used as single sensor to gather the data for the classifier in real-time which ensures a user-friendly and cheap option for daily technological assistance available for everyone.

The technological selection of using a neural network classification is compared with other possible approaches how to recognise the colour and ripeness of tomatoes and is justified as an excellent solution for an assistance application for harvesting decision making.

1 Introduction

"Colour blindness is more common than you might think! 1 in 12 men is colour blind. It is estimated that there are 300 million colour-blind people in the world." [2]

According to these fact, the disability of colour blindness is common in today's world and many people are affected by any kind of colour perception disorder. The diminished ability to perceive the colours red and green [2] results in difficulties for several parts of affected people's daily lives such as identifying signal signs. Another good example for this difficulty are fruits, as many of them change their colour from green to red when reaching a certain stage of ripeness. [3] Therefore, trivial tasks such as recognising the ripeness stage of a fruit become an extremely difficult and often insolvable problem for gardeners and farmers having a colour blindness disability. The technological solution to be developed is based on the initial problem that colour-blind people are not able to distinguish between the colours green and red and hence may not be able to recognise the ripeness state of fruits.

Therefore, the main use case of enabling colour-blind people to recognise the ripening stage of tomatoes and derive a harvesting decision through an intelligent technical solution was formulated for this project work.

The development of an Android application with the integration of an recognition or classification mechanism with the unique selling proposition (USP) of creating a cheap, easy to use application which delivers reliable results and can be customised to multiple user groups, especially private users, is subject of the following report.

Objective of this work

Objective of this work is to develop a customer-oriented, well-designed real-time application for ripeness recognition supporting people with a colour blindness disability for private cropping and harvesting of tomatoes. This application shall meet the requirements which will be gathered in a brief requirement analysis.

The application shall have the aim to assist people who are not able to make a harvesting decision based on the recognition of the colours green and red by themselves, and enable people's crop decision making through tomato ripeness recognition.

Central Question

In what manner can a real-time embedded system or application be developed and designed to best assist colour-blind people in making harvesting decisions based on the fruit state and what technological principles are the most suitable for this use case?

Outline

After this chapter examined the cause for this work through introducing the initial problem and the respective use case to develop a suitable technical solution for the identification of a tomato's ripening stage, chapter 2 introduces relevant theoretical foundations for this work. Hereafter, chapter 3 analyses briefly the core requirements for the solution development and chapter 4 displays an experimental research procedure and discussion about the use of a neural network and its supervised learning task with labelled input data as a suitable solution for the initial use case. Chapter 5 presents the design of the developed application considering different possible approaches to find and introduce the best fitting solution. Chapter 6 concludes the findings and briefly summarises the development process. Furthermore, it gives an outlook on possible extensions and improvements and additional use cases for the proposed solution.

2 Theoretical foundations

The following sections present the basic theoretical fundamentals of the relevant topics of this work.

2.1 Fundamentals of tomato cropping

The tomato plant (lat. Solanum lycopersicum) belongs to the nightshade (Solanaceae) family and is cropped worldwide in various sorts and different planting environments, mainly with the purpose of producing tomato fruits (berries) for human nutrition. [3] The economically significant plant for today's world has its origin in the Andes in South America and was formerly not planted for nutrition purposes as it was pronounced toxic when it first came to Europe in the 16th century. However, the tomato was introduced in some parts of Europe (such as Germany) only at the end of the 19th century. [3]

Today, the tomato plant is cultivated in grand style and has developed to one of the most important fruits of modern societies and can be grown in different ways in a warm and sunny climate. [3] Commercially, tomato plants are farmed in either a conventional or an organic tomato growing system which differ in the cropping and fertilisation methods of agricultural tomato producers. [4] However, the tomato plant can be also grown for private purposes which shall be the main focus of this work.

The berries of the tomato plant, the tomatoes, naturally have a red colour in their ripe state and can exist in several different sizes and shapes such as oval, big or small and round or in a heart shape, e.g. the oxheart tomato. [4] They build themselves during the fruit building process of the plant and remain green until they turn light red in the maturation process and finally become red when the fruits reach the full ripeness level. [1] Figure 2.1 shows the possible ripening stages of a tomato which are classified from left to right in *unripe* (green), *semi-ripe* (light red) and *ripe* (red).

Figure 2.1: Tomato ripening stages

Although the identification of the degree of ripeness seems to be easy through the recognition of the respective colour, there is a special case where it is not possible for a planter to precisely identify the ripening stage of a tomato with the naked eye. Such a case is the disability of colour blindness which will be introduced in the next section.

2.2 Fundamentals of colour blindness

The identification of a tomato's stage of ripeness by colour recognition is based on the ability of humans to have a functioning sense of vision. The human vision sense is achieved using cells called rods for luminosity and cones for colour perception. [5] People with a colour blindness disability have difficulties in perceiving colours which is often related to the inability of detecting and differing between the colours green and red. [2] Mostly, people discover their colour blindness by themselves when adjusting well-known objects such as traffic lights and realising not to be able to recognise the colours in the correct manner. There are different tests for colour blindness like the Ishihara plates test, which have numbers with colours that are confused with colour blindness. [5] The following figure shows an exemplary Ishihara test which helps to detect the existence and the degree of colour blindness through the ability to detect or not detect the green numbers (or only parts of it).

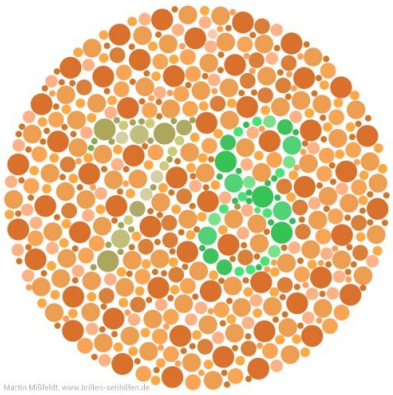

Figure 2.2: Ishihara test for colour blindness detection displaying the number 78

(https://brillen-sehhilfen.de/rot-gruen-farbschwaeche-simulator/rot-gruen-schwaeche-sehtest.php)

As colour-blind individuals can either recognise their disability through the adjustment of known objects with a certain colour order (traffic lights) or through a test (Ishihara) they have the difficulty to identify the paint of an object which is not clearly related to one colour state. [2] Due to that fact, the identification of the colour of tomatoes cannot be explicitly made successfully and therefore, colour-blind people are not able to recognise the ripening stage of their tomatoes by colour recognition.

Figure 2.3 shows two pictures. The picture on the left depicts the original image showing a ripe tomato on the left side and an unripe tomato on the right side. The picture on the right illustrates the perception with protanopia, a particular form of colour blindness.

Figure 2.3: Demonstration of the perception of colours with protanopia (right picture)

There are several digital assistant solutions available for the identification of colours for colour-blind individuals. These assistants developed and provided by computer scientists mostly rely on technological object and colour recognition methods. But an emerging technological solution for the improvement of such existing digital assistants is the usage of artificial intelligence (AI). [5] Recognising the ripeness of a tomato is a classification problem. As AI techniques exhibit great results by the classification of parameters, the use of neural networks which are based on such parameter classification seem to be a proper approach for a “cutting edge” ripeness detection solution.

Neural networks as an AI technique can learn to distinguish between ripe and unripe tomatoes and therefore can assist with a ripeness classification if colour-blind people are not able to.

Before entering the solution space for the development of a technical assistant for tomato ripeness classification, the following section gives an overview of basic fundamental principles of neural networks.

2.3 Fundamentals of neural networks

In this section, the general foundations of neural networks as an AI technique for learning tasks, such as recognising the colours that colour-blind people are not able to distinguish, are presented.

As this field of AI is complex and divers and cannot overall being presented on a few pages, the focus in this section shall lay on the overview of fundamentals for supervised learning with labelled training data, which implies that the neural network is generally used in a supervised way for a classification task. [6]

There are two main types of tasks within the field of machine learning: Supervised and unsupervised learning tasks. The main difference between these two types of tasks is that supervised learning relies on having knowledge of the target output values of samples (input data). This prior knowledge of what the output values should be is named “ground truth”. [7] By giving a sample of labelled data (input data) and desired outputs, a supervised learning task has the goal to learn a computational function which approximates the relationship between input and output observable in the data. [7] Supervised learning with training data is an important element of the further course of this work. On the other hand, unsupervised learning does not have labelled inputs or known outputs and unsupervised learning tasks follow the goal to infer the natural structure present within a set of data points. [7]

As mentioned above, supervised learning tasks in a neural network are using a labelled set of training data for the artificial learning process. The labeling of the data enables the possibility to measure the relationship between the input and the output data and hence must be known to have a comparable counterpart for the output produced by the learning task. As the goal is to learn a computational function which approximates the relationship between input and output observable in the data, the so called classifier must be trained. This classifier learns to classify the labelled input data through a learning algorithm. [6]
Although such a classifying learning method generally achieves fast convergence and good generalisation even with small labelled data sets as inputs, it can be further improved for a higher classification accuracy using larger sets of input data. [7]

Neural networks used in a supervised way for classification can be applied for tasks such as learning the colours that colour-blind people are not able to distinguish. The classification through neural networks can function as a colour determination method and can be made available and usable through deploying it into a piece of software.

The concrete approach of training the classifier within a neural network through a supervised learning task as well as the technical implementation within the scope of an Android application will be further presented in more detail in chapter 4 and 5.

3 Requirements analysis

This section surveys the customer demands on a support application for tomato ripeness recognition. These demands are captured from research of theoretical foundations [2] and the verbal interview of a befriended individual with a colour blindness disability.

The demands are formulated in the following core requirements (REQs).

- REQ1: Usability
  Target user groups shall get a well-designed, easy to use and intuitive application for tomato ripening stage recognition.
- REQ2: Reliability
  The application must be reliable in terms of a robust functionality in general and a high recognition accuracy of the tomato ripening stage.
- REQ3: Customisability
  Developers shall get the possibility to customise an initial application and extend functionality and features to other target user groups or different use cases at a later stage. This implies having a sophisticated technological concept and a best fitting software development method.

To create a suitable application which solves the initial problem and which meets the requirements REQ1 to REQ3, it is essential to carefully chose an appropriate technological concept for the development of the software. The following two chapters present the elaboration of a suitable solution based on the core requirements listed above.

4 Neural network

Initially during the presentation at the beginning of this workshop, there was the consideration whether to use object detection with colour recognition or a neural network in order to distinguish between ripe and unripe tomatoes. However, since recognising the ripening stage of a tomato is a classification problem (as mentioned in section 2.2), the decision was made to use a neural network.
Furthermore, neural networks can provide excellent results with a high reliability (REQ2), even in scenarios where object detection and colour filtering meet their limits. For instance, there are different shapes of tomatoes which can heavily complicate the object detection as well as branches and leaves that may overlay a part of the tomato. Moreover, object detection requires a high contrast to work reliably which might not always be given, e.g. if there is a green tomato with green leaves in the background that should be recognised.

In order to train the model, supervised learning with labelled input data was applied. This chapter covers the basic idea of training a neural network as well as the actual training steps and the iterative improvement of the classifier.

4.1 Collecting the training data

As preparatory task for the supervised learning, a set of training data was created by searching for tomato images on the web. Furthermore, the downloaded images were manually labelled and split into three classes *ripe*, *semi-ripe* and *unripe* according to the patterns and characteristics of the different ripening stages acquired during the scientific research in section 2.1. [1]

As creating the data set was expected to be very time-consuming, a web search was performed first to look for already existing tomato image sets. Unfortunately, no suitable data set was found. Hence an own one was created and gradually refined during the iterative training process described in section 4.2.2.

4.2 Training the neural network

The neural network was trained with TensorFlow, an open-source software library for machine learning. In order to setup and run TensorFlow, Anaconda was installed preliminary as Python command line application.

Figure 4.1: Three ripening stages: *unripe*, *semi-ripe* and *ripe*

4.2.1 Training script

Several Python scripts for TensorFlow are available on the internet [8] that allow to easily train and test a neural network by just running the script with custom input parameters. For this project, a script was chosen that is already optimised for image training. For this reason, the default configuration values were left unchanged. By default, the learning rate is set to 0.01 and the training runs 4,000 epochs. Furthermore, 10% of the input data set is used for validation and 10% is used for testing.

At the end of the training, the script creates two output files: a graph file containing the trained model and a labels text file containing the names of the classes in plain text. These files can be used as classifier in line with the Smartphone-Sensing-Framework that was used as basis for the developed Android application, see chapter 5 for further details.

4.2.2 Iterative training

The whole training was an iterative process of successively improving the classifier, i.e. multiple versions of the model were created and tested. The four main training iterations with major changes are itemised and described below. Several further trainings with minor adaptions were also conducted but are not worth mentioning since they only provided insignificantly different results. Furthermore, each iteration is compared with the previous one and training, validation and testing result values are given. The goal was to create a classifier that is capable of reliably (REQ2) recognising the three different ripening stages *unripe*, *semi-ripe* and *ripe*, as displayed in figure 4.1.

Iteration 1: The first version of the classifier was rather a proof of concept than a sincere attempt. The model was only trained with images of the two classes *ripe* and *unripe*. The training took about 20 minutes and resulted in a surprisingly well working model that was capable of classifying almost all of the test images correctly. This was not expected because only a very small set of training data (about 30 images per class) was used. The training and validation accuracy were both 100% while the testing accuracy was 92%. The great results are most likely caused by the fact that a ripe and an unripe tomato can be clearly differentiated because of the completely different colours. Listing 4.1 shows the console output (relevant lines) after the training.

```
INFO:tensorflow:2020-04-12 16:12:31.033071: Step 3999: Train accuracy = 100.0%
INFO:tensorflow:2020-04-12 16:12:31.033525: Step 3999: Cross entropy = 0.018120
INFO:tensorflow:2020-04-12 16:12:31.151234: Step 3999: Validation accuracy =
    100.0% (N=100)

INFO:tensorflow:Final test accuracy = 92.0% (N=10)
```

Listing 4.1: Console output for 1st training

Iteration 2: In the next step, the third class *semi-ripe* was introduced which unfortunately complicated the whole training significantly. Furthermore, a worse result was kind of expected since it's pretty easy to distinguish between a ripe and an unripe tomato but the transition between ripe and semi-ripe as well as between semi-ripe and ripe is fluent, i.e. the classification is not that distinct. However, this attempt with 3 classes was disastrous. The training accuracy after 4,000 epochs was only 72% and the validation accuracy was even worse with only 42% (which is a value that almost can be achieved when guessing the class randomly).

Iteration 3: Supervised learning has a high data dependency. The most critical success factor to get a good model is the quality and quantity of the training data. Hence the size of the data set was dramatically increased. For this purpose, another web search was performed and more tomato images were stored and labelled. Subsequently, another training was started. This time, the result was a lot better but still not good enough. The training and validation accuracy was 100% and 51% respectively and the testing accuracy was 76.9%. However, still not all images were recognised reliably, i.e. REQ2 was not yet fulfilled. In particular, a lot of results were inaccurate or not distinct enough, e.g. a test image showing an unripe tomato was classified as *unripe* with a confidence of around 40% and as *semi-ripe* with a confidence of approximately 60%.

Iteration 4: In order to further improve the accuracy, the training data was reviewed and some of the images were re-labelled. Mostly all images were sliced in order to reduce the amount of unnecessary information. Some images were removed because they were not distinct enough for a particular ripening stage (it was even hard to classify them manually). To further increase the training data set, 3 additional versions of each image were created with a rotation of 90, 180 and 270 degree. After these adaptions, another training was started and it turned out that these measures had a great effect on the result. Furthermore, the training and validation accuracy increased to 100% and 97% respectively while the testing accuracy was 87.8% which is pretty good. The shortened console output of the training script is shown in listing 4.2.

```
INFO:tensorflow:2020-04-13 20:05:18.041195: Step 3999: Train accuracy = 100.0%
INFO:tensorflow:2020-04-13 20:05:18.042185: Step 3999: Cross entropy = 0.042996
INFO:tensorflow:2020-04-13 20:05:18.173626: Step 3999: Validation accuracy = 97.0%
      (N=100)

INFO:tensorflow:Final test accuracy = 87.8% (N=41)
```

Listing 4.2: Console output for 4th training

After each training iteration, the created classifier was tested, amongst others with the test images shown in figure 4.1. Whereas the earlier versions of the model could not provide satisfying testing results, the final model classified all three images correctly with a very high confidence. Concluding, the testing results of the final model are shown in the following listings.

```
ripe 0.99176675
halfripe 0.0065282513
unripe 0.0017049202
```

Listing 4.3: Testing result for image A.1 (*ripe* tomato)

```
halfripe 0.99167824
unripe 0.0071457503
ripe 0.0011759808
```

Listing 4.4: Testing result for image A.2 (*semi-ripe* tomato)

```
unripe 0.9640221
halfripe 0.03187752
ripe 0.004100373
```

Listing 4.5: Testing result for image A.3 (*unripe* tomato)

The classifier was subsequently integrated into the developed Android application and tested with real sensor data from the camera. It turned out to solve the initial problem of recognising the ripening stage of tomatoes pretty good. Furthermore, the background of an image seems to have only little impact on the classification accuracy since even tomatoes with a non-white background (see images A.4 and A.5) are classified correctly with a high confidence which is important to be suitable for the original use case. However, as the colour turned out to be the most significant classification factor, problems can arise with falsified colour information of the captured camera image, e.g. caused by the automatic white balance of the camera, which can negatively affect the reliability (REQ2). This leaves room for further improvements which are discussed in the future prospects in chapter 6, although this is no problem with the model itself but rather with the captured sensor data.

4.3 Alternative approach

As an alternative solution to train the neural network, Teachable Machine was investigated. Teachable Machine is a web application developed by Google which allows training a neural network directly in the browser. Furthermore, it uses transfer learning which can significantly improve the accuracy of the trained model, especially for very small sets of input data. In deep learning, a large amount of training data is required to teach the classifier to understand particular patterns of the input data. With transfer learning an already existing model that was trained on a similar task is re-trained with the new input data. In this way, the knowledge of the original model (in respect of understanding the patterns) can be transferred to a new data set. As a result, the re-trained model offers a very high classification accuracy, even for a training data set that is in fact insufficient. [9][10]

For this reason, Teachable Machine is an interesting alternative to the classical way of training a neural network because it can significantly decrease the effort required to collect training data. Therefore, a quick experiment was conducted. Although only 20 images of each ripening class were put into the neural network, the accuracy of the resulted classifier was pretty good. Furthermore, training the model only took about 2 minutes. Unfortunately, no way was found to use the exported model together with the Smartphone-Sensing-Framework. After several unsuccessful attempts and hours of internet research, this solution was deemed to be incompatible and thus was discarded. Eventually, a model was trained locally instead as described in the previous section.

5 Android application

An Android application called CropTomato was developed and implemented in line with this workshop. It is based on the Smartphone-Sensing-Framework, developed by Dionysios Satikidis and Dimitrios Lagamtzis, which was provided in line with the lecture documents. The framework provides example code of how to capture and process the data of the smartphone's sensors, e.g. gyroscope, accelerometer, microphone, camera etc., which simplifies the implementation. The application was developed in Java with Android Studio. Although it was implemented as prototype, attention was payed to develop a clean architecture providing the possibility to easily reuse and customise the application (REQ3). Furthermore, the trained TensorFlow model was integrated.

In the developed application, only one sensor is needed which is the smartphone's camera. It is used in order to capture images of tomatoes which are then passed to the neural network to classify the ripening stage. The camera fulfils the requirements of the application albeit bad light conditions can negatively affect the quality of the sensor data which can have an impact on the reliability (REQ2).

In the following sections, the application is described in detail. The final implementation is discussed and compared to the initial concept introduced in the presentation at the beginning of the workshop.

5.1 Graphical user interface

A detailed description of the application's graphical user interface (GUI) is given in this section. In subsection 5.1.1 the initial design of the GUI is depicted. However, the first draft had to give way for an improved version which is described in subsection 5.1.2.

5.1.1 Initial idea

The initial concept of the GUI was a quick draft which was designed for the project presentation. Figure 5.1 shows a mockup of the initial design.

The interface should originally consist of a view showing the real-time video stream captured by the camera. Moreover, the initial idea was to take a photo of a tomato first (by tipping on the display) which should subsequently be analysed. On the bottom of the layout, the degree of ripeness of the tomato should be displayed next to a crop recommendation (or a prediction when the tomato will be ripe enough).

However, the original idea was discarded for the following reasons: Having to take a photo each time does not provide a good user experience and usability (REQ1). Furthermore, since the decision was made to use a neural network to classify the ripening stage, a much more detailed ripeness information could be given to the user. Moreover, whether a tomato should be cropped or not is rather a subjective decision that depends on several factors, e.g. if the user plans to eat the tomato directly or in a few days, or whether he wants to use it for cooking or eat it raw. Likewise, a prediction of when a tomato will be ripe is tied to too many factors, e.g. weather conditions, temperature etc. With this in mind, the user should rather decide himself whether he wants to crop the tomato by considering the displayed ripeness information.

Figure 5.1: Mockup of the initial GUI design

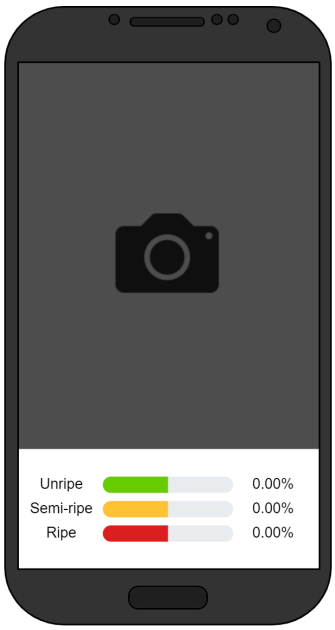

Figure 5.2: Mockup of the final GUI design

5.1.2 Final version

The final and improved UI design presents the user more detailed information about the degree of ripeness. A mockup of the final GUI is shown in figure 5.2.

Since the classifier distinguishes between three different ripening stages (*ripe*, *semi-ripe* and *unripe*), the GUI visualises the confidence levels of all of these three classes. Likewise, each progress bar has a particular colour which correlates with the ripening stage. The crop recommendation was removed as mentioned before. In contrast to the initial concept, there is no need to take a photo.

Instead the classification is done automatically in real-time, i.e. the progress bars and percentage values at the bottom of the GUI are continuously updated. In conclusion, the final design is a big improvement in respect of the user experience and usability (REQ1) of the application compared to the initial draft.

Figure 5.3 shows three screenshots of the GUI, one for each ripening stage. The results are close to perfect due to the white background, not only because there is almost no interference in the test images but also because it prevents the camera from performing an automatic white balance that could falsify the colour information in the image.

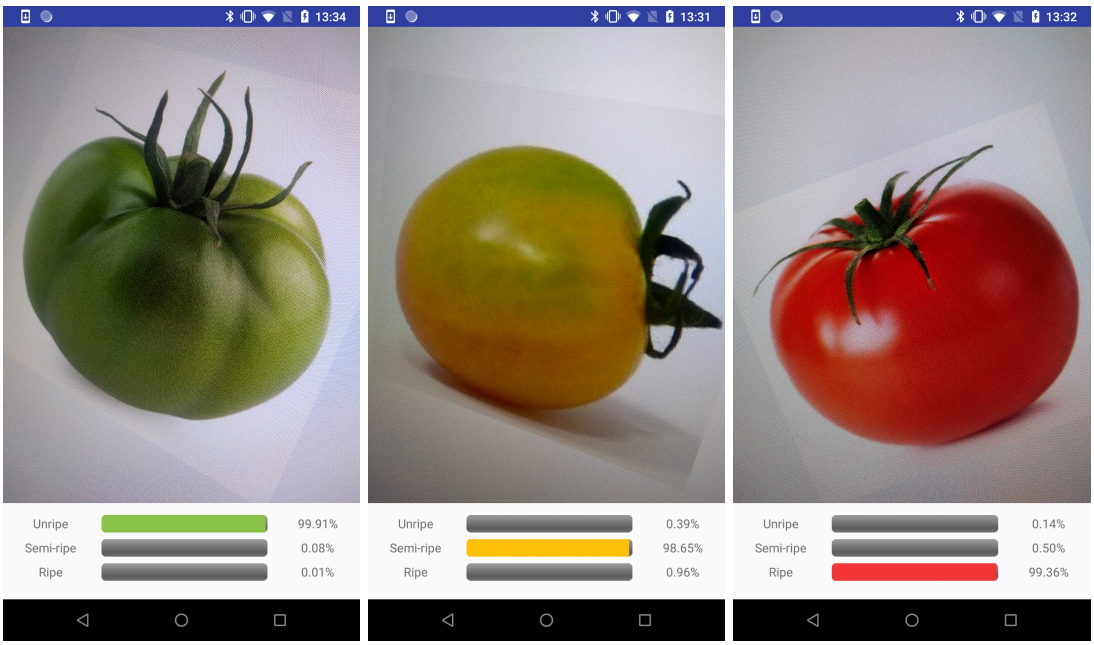

Figure 5.3: Screenshots of the GUI for all three ripening stages

5.2 Operating mode

In this section, a detailed description is given of how the implemented application works. The initial idea from the presentation was adapted during the development process. Initially, it was in consideration to use object detection in combination with colour recognition to determine the degree of ripeness. But eventually, the decision was made to use a neural network to classify the ripeness for several reasons (as already described before). Moreover, adaptions in the GUI layout were made as already mentioned. These adaptions, of course, resulted in changes in the program flow. In the following, the initial concept and the final implementation are compared to each other.

5.2.1 Initial idea

In the initial concept, the user had to take a photo of a tomato first which should then be passed to the object recognition. If a tomato was detected, the colour of the tomato should be analysed and afterwards the result should be returned by updating the GUI. The user could then take the next photo to be analysed. Figure 5.4 shows the initial flow chart. The process of updating the GUI is missing in that chart.

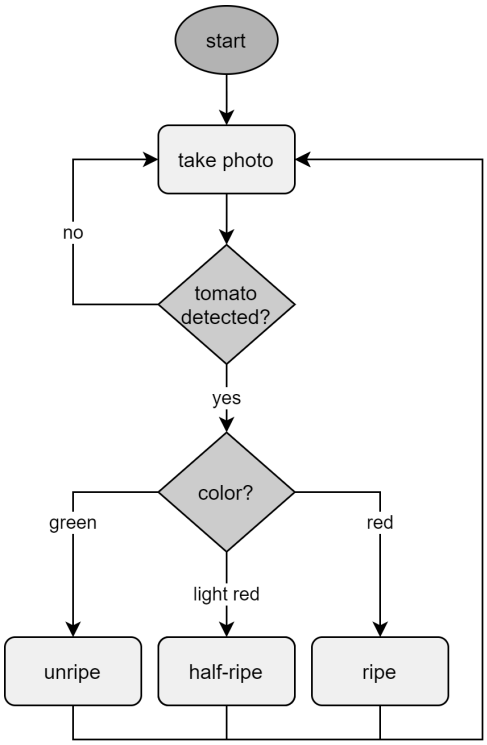

Figure 5.4: Initial program flow

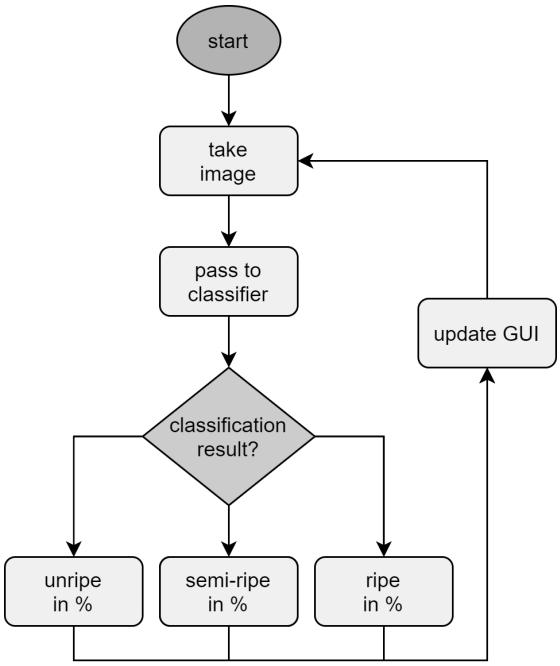

Figure 5.5: Final program flow

5.2.2 Final version

The initial draft was discarded for user experience and usability (REQ1) reasons. The program flow of the final solution is shown in figure 5.5. Instead of having to take a photo, the camera video stream is processed in real-time. Continuously, a single frame from the stream is captured and passed to an asynchronously executed classifier task which uses the trained TensorFlow model. When the classification is finished, the GUI is updated with the classification results (confidence values of each class) and the classifier task ends. Afterwards the next image is taken and passed to the classifier automatically. The whole procedure runs in an infinite loop. In conclusion, this solution is much easier and more straightforward than the initial approach. Furthermore, it does not require any user interaction which improves the usability (REQ1).

6 Conclusions and future prospects

Conclusions drawn from the workshop are depicted in this chapter. Moreover, a short statement about the performance of the developed application is issued. Furthermore, several possibilities of improvement and usage of the application in the future are discussed.

During this workshop, an Android application was developed using technologies which were considered most suitable to solve the initial problem. A neural network was trained to enable the application to recognise the ripening stage of tomatoes. The initial GUI design and program flow were discarded for a more straightforward solution. Finally, the project resulted in a functional prototype that is capable of recognising the ripening stage of tomatoes.

Using a neural network has several advantages over using object and colour recognition. It not even provides a very high reliability (REQ2) but also decreases the implementation effort significantly since only the trained model has to be integrated into the application. The model contains the whole classification logic. Furthermore, due to the TensorFlow training and testing scripts [8], no deep knowledge of neural networks is necessary to achieve a satisfying result. In conclusion, it's a more straightforward solution. However, collecting the training data turned out to be very time-consuming.

The application meets the expectations and requirements (see chapter 3) albeit there is room for improvements in all aspects. A good usability (REQ1) was ensured by discarding the initial GUI and operating concepts in favour of a more straightforward approach. By using a neural network, a good reliability (REQ2) was achieved. The customisability (REQ3) of the application was ensured by building a clean and extensible architecture. However, there are several possibilities of how the application could be further improved in the future.

First of all, the user experience and usability (REQ1) could be improved by adding a bounding box in order to show which of the tomatoes in the image is classified. Furthermore, it would be nice to have the possibility to choose the tomato to classify by tipping on the corresponding image section.

The reliability (REQ2) of the application mainly depends on the accuracy of the classifier. In order to improve it, the TensorFlow model could be retrained with an improved

and increased set of input data or by using alternative approaches as transfer learning, which was already discussed in section 4.3. Likewise, it is also conceivable to extend the model and customise (REQ3) the application in order to recognise the ripening stages of other types of fruit and vegetables.

During the testing of the application, it was observed that the quality of the sensor data (i.e. the captured camera image) could negatively affect the result under certain circumstances. Depending on the light conditions the colour information of the image can be falsified which may lead to an inaccurate classification that negatively affects the reliability (REQ2). Further investigation could be made of how this issue can be solved.

Last but not least, a better user experience and usability (REQ1) could be achieved by improving the speed and latency of the application. For instance, OpenCV was already replaced during the development process which significantly increased the FPS (frames per second) of the camera stream.

Although the primary use case (objective) for this work was the provision of a real-time application for recognising the ripeness of a tomato, supporting people with a colour blindness disability when cropping and harvesting tomatoes in private, there are several conceivable other business cases as well.

As an outlook it can be proposed that the initial use case can be extended on other use cases such as the support of commercial harvest workers or even for the use of harvesting robots in the future as mentioned in the introduction of this work. These use cases would create a benefit for the commercial agriculture sector by providing a reliable (REQ2) solution for tomato ripeness recognition. As already mentioned, the classifier could be extended to recognise other fruits and vegetables such as strawberries, peppers, amongst others, and therefore the application could be widely used in private and commercial environments. Hence, the use case of supporting disabled private people would be extended to also enable commercial efficiency through the usage of AI techniques.

Concluding, it can be said that a neural network based classifying application which meets the requirements REQ1 (usability), REQ2 (reliability) and REQ3 (customisability) can be widely used. Therefore, such an application can be seen as an effective and useful technological tool to enable personal and economical improvements.

References

[1] H. Yin, Y. Chai, S. X. Yang, and G. S. Mittal. Ripe Tomato Recognition and Localization for a Tomato Harvesting Robotic System. pages 557–562, 2009.

[2] R. N. Shepard and L. A Cooper. Representation of Colors in the Blind, Color-Blind, and Normally Sighted. *Psychological Science*, 3(2):97–104, 1992.

[3] Gabriella Natho and W. Franke. *Nutzpflanzenkunde. Nutzbare Gewächse der gemäßigten Breiten, Subtropen und Tropen*. Thieme-Verlag Stuttgart, New York, 5th edition, 1997. ISBN 3-13-530406-X.

[4] Wagner Bettiol, Raquel GHINI, José Abrahão Haddad Galvao, and Romildo Cássio SILOTO. Organic and conventional tomato cropping systems. *Sci. agric. (Piracicaba, Braz.)*, 61:253–259, 2004.

[5] Rivera, Martín Montes, Alejandro Padilla, Juana Canul-Reich, and Julio Ponce. Real-Time Recoloring Ishihara Plates Using Artificial Neural Networks for Helping Colorblind People. User-Centered Software Development for the Blind and Visually Impaired: Emerging Research and Opportunities. *IGI Global*, pages 138–156, 2020.

[6] Dimitrios Moshou, Els Vrindts, Bart De Ketelaere, Josse De Baerdemaeker, and Herman Ramon. A neural network based plant classifier. *Computers and Electronics in Agriculture, ISSN 0168-1699*, 31:5–16, 2001.

[7] Devin Soni. Supervised vs. Unsupervised Learning. https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d, Accessed: 15.04.2020.

[8] GitHub - tensorflow/hub: A library for transfer learning by reusing parts of TensorFlow models. https://github.com/tensorflow/hub, Accessed: 11.04.2020.

[9] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. *A Survey on Deep Transfer Learning*. Tsinghua University, 2018.

[10] Kartik Balasubramaniam. Have you taught your machine yet? https://towardsdatascience.com/have-you-taught-your-machine-yet-45540b7e646b, Accessed: 20.04.2020.

A Appendix

The appendix contains several example testing images that can be used to test the trained model directly or with the developed Android application.

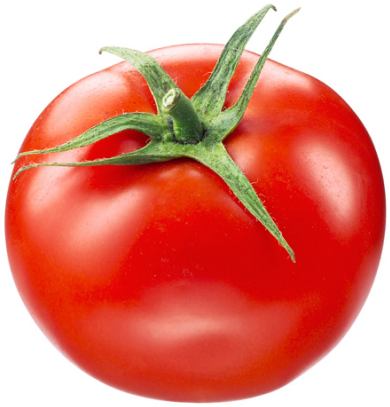

Figure A.1: Test image with *ripe* tomato

Figure A.2: Test image with *semi-ripe* tomato

Figure A.3: Test image with *unripe* tomato

Figure A.4: Test image with *ripe* and *unripe* tomato

Figure A.5: Test image with *ripe*, *semi-ripe* and *unripe* tomato