School of Engineering and Design
Electronic and Computer Engineering
M.Sc. Course in Distributed Computer Systems Engineering

Workshop 7

Embedded Systems Engineering
Real-World Smartphone Sensing

Team D:	BananaCo
Date:	23rd April, 2019
Lecturer:	Dionysios Satikidis, MSc

Team members

Scientific Researchers	1841780	Bimba Bhat
	1841787	Alexander Hermann
Data Analyst	1841797	Fabian Peltzer
Application Developer	1841795	Michael Watzko
Documentation	1841789	Julian Maier

Deadline:	23rd April, 2019

Content

Figure Index

Abstract

The report at hand was part of the workshop assignment of the lecture module "Embedded Systems Engineering" in the context of the MSc course "Distributed Computing Systems Engineering" at Brunel University London. It was worked on by a team of five students – their names and general project roles can be found on the title page.

The initial task of the workshop was to come up with a problem solvable with only a recent smartphone and its built-in sensor systems. Later on, a creative solution to that problem had to be developed in teams of four to five people.

The problem worked on in the report at hand was to identify a banana's ripeness with the camera hardware of a recent smartphone.

Initially, Computer Vision algorithms and image processing where considered. A mobile application was to be implemented, realising object detection for bananas first and colour information processing second. Brown spots or larger dark parts of the banana surface area were to be considered in the colour analysis. As bananas change their colour over the course of their ripening process, this approach made a promising impression.

Literature review on bananas themselves has been conducted to grasp the fundamentals of banana knowledge and identify visual criteria usable for identifying the ripeness stage of bananas. Out of all criteria to consider, the colour of certain parts of the banana turned out to be the most prevalent.

The initial technical problem solution approach using Computer Vision algorithms (object detection and colour analysis) was scrapped for a more interesting, up-to-date method. Using the software library TensorFlow for machine learning, a neural network was trained with labelled sets of images to function as a classifier for three different banana ripeness stages. The stages where set to be "not ripe yet", "ripe and edible" and "overripe".

After training the neural network with enough imagery, the classifier was able to clearly categorise bananas in one of the three ripeness stages chosen. It was then built into a mobile application that fed the classifier with images via the smartphone camera and displayed the classification results in a clear way.

In the end the mobile application we developed was able to detect the ripeness of banana imagery given by live camera accurately, with clear probability emphasis on one of the three ripening stages we trained the neural network for. The training of the network actually worked well enough to mostly mitigate the impact of the image background information on the classification result, which was a very surprising, but welcome outcome.

The workshop concluded in a successfully functioning prototype.

1   Introduction

1.1   Initial Problem

Our world is full of challenges to overcome. Most tasks are easy to tackle for most people, but can present hard- to non-solvable challenges to people with particular impairments.

For example, stairs are a serious hurdle for people unable to walk, acoustic announcements can miss people with hearing impairments, standard computer peripherals like monitors, keyboards and mice can be hardly usable for those with special needs on equipment, and special attention should be taken into consideration to form public environments which are easily and safely to navigate for the blind people in our communities.

Bananas are a good choice for a tasty and nutritious meal or ingredient, but one must choose the right bananas to buy for one's use case. As bananas change their colour over the course of their ripening process, its trivial for most people to detect the right ripening level of the bananas to buy.

For others, e.g. people with colour vision deficiencies, fruit colour and/or ripeness identification can present a serious challenge in general. Depending on the type of their colour vision alteration, bananas can appear to all be the same colour; at least green and yellow bananas. Others can have problems detecting patterns on the fruit. All that can make choosing the right bunch to buy quite difficult.

1.2   Problem Solution Approach

We wanted to solve a real world problem in a creative way. One of the project team members has a weakness in red-green colour vision. Bananas are counted among his favourite fruit – but he can't distinguish green from yellow bananas, as they appear to be the same colour for him.

This problem was to be solved with a mobile application, using the built-in smartphone camera for a live video feed of the banana(s) in question. The imagery was to be analysed in a way that would enable the ripeness / maturity detection of the banana(s) in the real world in "real time", so it could actually be used on-the-fly in the supermarket.

Direct and indirect target user groups of such a mobile application or further specialisation of the general solution would be:

Private end users:

Having the application installed and available on their smartphones, the application can help end users with colour vision deficiencies to aid them in choosing the right bananas to buy and eat.

Industry:

Adapted for and embedded in industrial automation technology, the banana ripeness stage classifier could be used in automated packaging processes, routing bananas of different ripening stages to distribution parcels intended for various marketplaces – sending ripe bananas to locations near the packaging centre and bananas with more ripening time left to places overseas.

Retailers:

The ripeness detection could be used to sort out fruit gone bad at unpacking after transportation.

2 Bananas

2.1 Introduction

The word *banana* originates from "BANAN", the Arabic word for "finger". Banana crop is a commercially and economically used crop. In the genus *Musa*, there are five main taxonomies. Out of five, two are considered as edible bananas.

In general, a distinction must be made between plantains and fruit-bananas. Plantain banana belongs to the Musa family, having the scientific name *Musa paradisiaca. Musa acuminate* is the most cultivated banana. The largest herbaceous plant is the banana plant. The two types can be differentiated by their texture and taste. Plantains are starchy and less sweet. They are not edible in raw form but rather must be cooked before consuming.

In the context of BananaCo project, only the latter is taken into consideration. Although the exact origin of Banana is unknown, some of the research stated that it originates from the Indo-Malaysian region. However, both types of banana originate from tropical regions, predominantly Africa and South America, also from the subtropics and are best suited for warm, costal climate. While fruit-bananas are edible instantaneously, plantains require to be cooked initially to be palatable.

Opposed to fruit-bananas, plantains are rather angular and thicker. In addition, plantains are coloured pale-yellow, grey or cream; once ripe they are characterised by a violet or black peel. Banana peel colour is to be considered as the first quality parameter evaluated by consumers. In fact, the external condition correlates well with its internal, physical and chemical changes during the ripening process.

Commercial bananas are plucked and packed when they have grown enough but are still green. These bananas are stored or transported in air tight room or container and release ethylene gas when they are ripening. This will speed up the production of enzymes that change texture, colour and flavour of the banana.

Some facts about the ripeness of banana:

- underripe bananas contain less sugar;
- overripe bananas are good for nutrition, since they are easier to digest;
- overripe brown pigmented bananas have antioxidants;
- overripe bananas are sweeter than just ripe bananas, because starch transformed into sugar;
- due to bananas being rich in fibre, it may lead to flatulence, stomach pain and cramps;
- in some cases, banana consumption may trigger migraine.

The BananaCo application shall support to find out the ripeness. Different category people get benefit according their need. For instance, the people who would like to lose weight could consume under ripe banana which is less sweet in taste. Whereas the people suffering from heart disease might like to have over-ripe banana. To detect the different ripeness level easily the application will support the people.

2.2 Background

Bananas may be consumed easily and have lots of health benefits with their richness in antioxidants, vitamin B6 and minerals such as potassium. The consumption of bananas is good for the overall health. It is especially healthy regarding the strengthening of bone structure, reduction of digestion problems, improving vision, and so on. Also, it is a natural source of potassium supplier to the human body.

The maturity stage of fresh banana is important in marketing, for both, industry and dealers on the one hand as well as end consumers on the other hand. In early ripening stages, banana fruits synthesize compounds such as alkaloids and tannins, making the fruit taste bitter and astringent. In progressing stages of growth, the fruit incorporates water, sugars, starches, acids and vitamins.

The banana peel is green due to the internal process of chlorophyll molecules breaking down enzymes. Bananas appear in a yellow, golden colour when the green pigment of chlorophyll is destroyed. An amylase enzyme is responsible for breaking down the starch into glucose, hence bananas become sweet. The pectinase enzyme breaks the cell walls in the fruit which makes bananas soft.

The softness of banana increases as it ripens more and more, leading to banana discolouring or blackening. Contusion is caused by the polyphenol oxidase enzyme which also increases the oxidation. In general, it is applicable to the peel, but it also affects the flesh of the fruit in case of deep bruising. At present days, seven ripening stages of bananas are recognized. All stages are purely based on the colour of the peel, which is considered as preliminary quality parameter by the buyers as well.

Variation in respiration of fruit during the ripening stage was a crucial factor since many years in classifying the fruit ripeness. Particularly, the colour development time changes for many fruits in relation to their climacteric peak. It is considered that ethylene concentration is one of the crucial factors for ripening, not only for bananas but also other fruits as well.

The taste also varies as colour changes. A banana fruit with a peel colour of golden yellow to light brown could be considered having the best flavour and texture. The ripening process itself can be speed up by increasing the ethylene gas concentration in the storage room. This can be achieved by covering loosely, unripe bananas.

In the meantime, the banana fruit turns from green to yellow, then from yellow into yellow with brown spots. Finally, starch and acid contents decrease, while sugar increases; alkaloids and tannins disappear, aromas develop. The calorie content however remains the same, independent of the degree of maturity.

> *"To ensure the productivity, competitively, quality standards, and reliability of banana fruit products, automatic image processing tools based upon intelligent techniques are paramount over visual features methods." [Mazen2019]*

According to *London and Walder*[1], the nutrition content of a ripe banana comprises:

- 105 calories,
- 27g carbohydrates,
- 1g protein,
- < 1g total fat,
- 0g saturated fat,
- 3g fibre,
- 14g sugar,
- 422mg potassium (12 % DV [daily vitamins]),
- 32mg magnesium (8 % DV),
- 10.3mg vitamin C (17 % DV),
- 0.433mg vitamin B6 (20 % DV).

---

[1] London, Jaclyn, Walder, Caroline, Why You Should Eat a Banana Every Single Day, Good Housekeeping 25 September 2018, see: https://www.goodhousekeeping.com/health/diet-nutrition/a47807/banana-nutrition/ (accessed 14/04/2019).

2.3   Maturity Assessment

2.3.1   General Criteria

To detect and classify bananas, certain criteria need to be examined, which will be provided subsequently. In theory, one can use several aspects to determine the maturity of fruits in general and of bananas in particular, encompassing:

- size / shape,
- peel texture features,
- degree of hardness (e.g. hard or soft),
- starch / sugar proportion,
- smell,
- flavour (e.g. blunt, sweetish, sweet) and, of course the
- peel colour (e.g. green vs. yellow vs. brown).

Hence the ripeness is categorized based on the colour of ripeness stages. It is difficult to distinguish between one stage ripeness and another. The colour does not vary much from the third to fourth, fourth to fifth and fifth to sixth stage. Therefore, BananaCo application uses only three stages of banana ripeness rather than seven stages.

Bananas can also be classified by their size. There are different types of banana. Peel thickness also varies with banana types. Fruit-banana breeds vary with the size as well. Some studies provide information about determining the banana size using automatic algorithm derived from computer vision. Banana images are processed in three stages.

The specific objectives of [Meng-Han2015]'s work was:

(1) To detect the pedicel location;

(2) to test the performance of the "Five Points Method", which is the key sub-algorithm of the automatic measurement algorithm;

(3) to determine the three size indicators of bananas using computer vision and to compare the performance of three different methods.

The report concluded that automatic algorithm for banana size determination was acceptable. But in the BananaCo application, an image processing technique is aimed to be implemented. The basic idea behind the application was to collect data from the Internet and process the image colours. Around three hundred images were collected to train and test the application.

The basic structure of any fruit or vegetable is its texture, colour and shape. Using these features play an important role in visual perception. By extracting the useful spectral property of the image and matching it best with the set of known predefined classification model, colour classification can be done. An application with combination of both texture and colour features is very demanding and helpful in the food industry.

Some of the challenges and limitations that may arise in fruit detection are recognizing the fruit based on the size. If the fruit shape is completely unique, it might not be a problem. But if the shape is alike others, then it might cause some problems. Using classification algorithms, it is possible to reduce the effect of identical shape problem.

Colour recognition itself might be a challenge or a limitation. In the application only fruit-bananas are used to train the system. In case the customer would like to differentiate between fruit-banana and plantains, then the application might not produce the correct, i.e. expected result.

2.3.2 Visual Criteria

In literature, a lot of methods that have been developed for ripeness classification involve *colour moments* and *colour histogram*. Also, the variance of RGB (Red Green Blue) or HSV (Hue, Saturation, Value) colour spaces of the banana fruit have been utilised for analysis. According to [Mazen2019], the classification of banana fruits as under-mature, mature and over-mature reached an accuracy of 99.1 %.

Visual inspection by humans may underlie subjection and is tedious as well as time-consuming and labour-intensive. Utilising instruments such as colorimeters provide the advantage of accurate and reproducible measurements but require quite unique surface colours. Additionally, several sample locations are required to produce representative results.

BananaCo on the contrary focuses onto automated visual, i.e. image recognition utilising cameras integrated into smartphones. Also, computer aided analysis techniques are used, offering objective measurement and mitigating deficiencies of manual visual and instrumental techniques. Suitable aspects for visual detection include:

- size / shape,
- peel colour,
- development / mottle of brown spots and
- analysis of peel texture features.

The BananaCo application can be used for different purposes. Some of the possible use cases are described in detail below:

- To detect defected fruits in the food industries. In food industry, it is difficult to identify damaged fruits from the lot. This process needs lots of human resources and is quite time consuming. To automate the defect detecting process, the application at hand can be used.
- To classify the fruits, the application can be used. Till recent year, food grading is also one of the most time consuming and manually carried out processes. Whether it may be farmers, food industries or supermarkets grading the food for different application, a lot of manual work is involved.
- By using the application, one could easily differentiate the food according to its grading. It can be installed on smartphones and it then used without the need of any additional components. To capture the image of the fruit, the embedded smartphone camera shall be used.
- The consumer could use it to check the fruit quality. Bananas colour ranges from green to yellow, pigmented yellow and brown. According to their needs, people could use the app to detect the ripeness stage of the banana fruit. Elderly people face difficulties because of the poor vision. The application could assist them in recognising the best fruits.
- Furthermore, the application could be used to assist and educate people suffering from disabilities such as Down syndrome, blindness or some other specially impaired and / or challenged people.
- For example, people suffering from the Down syndrome, often have trouble understanding certain patterns. The results produced by the app could be used to train them in pattern recognition.
- Finally, in the retail sector, the application could be used to label banana fruits (e.g. in supermarkets).

2.4 Classification

2.4.1 Feature Selection

Regarding literature, one encounters most approaches in classifying the maturity level of fruit-bananas to be based on at least five, more frequent seven[2] or even 15 stages. In the scope of BananaCo project, the smartphone camera is used to scan fruits and determine their maturity based on visuals.

To limit the complexity within the boundaries of the project, the granularity is initially limited to subsequent three ripening stages, which will serve for classification in the creation process of data sets. Each class exhibits distinct characteristics according to the three feature aspects stern, fruiting body and tip (table 1).

Class	Peel colour	Maturity stage	Feature Aspects		
			Stern	*Fruiting body*	*Tip*
1	green	unripe	green	green	green
2	yellow	ripe	yellow	yellow	brown
3	brown	overripe	brown	brown, min. 50 % of peel surface	brown

Table 1: Maturity categories

2.4.2 Methodology

The criteria listed before is used later to manually categorise banana images acquired from image research on the Internet. The classification will be carried out according to the three maturity stages *unripe*, *ripe*, *overripe* (cf. figure 2), thus creating data sets. These images will then be labelled and fed into the computer vision or rather neural network, serving as training data.

Figure 1: Banana ripeness stages as used for BananaCo classification

---

[2] According to [Mendoza2005], seven stages are recognised in the context of trading: stage 1: green; stage 2: green, traces of yellow; stage 3: more green than yellow; stage 4: more yellow than green; stage 5: green tip and yellow; stage 6: all yellow and stage 7: yellow, flecked with brown.

3   Neural Network

3.1   Introduction

Recognising the ripening state of a banana is a *classification problem*. Neural networks are made for this kind of problems. A classification problem has input data and outputs. The input data is classified during the classification process. All possible classes that can be the result of the output are predefined during the training process of the classifier (in this case the neural network). The output is always one of the classes that were defined in the training.

During the training process, labelled data is fed to the network. Every image of the training data has a label in the form of a class attached to it. The image is fed into the neural network and the result is compared to the attached label.

The neuronal network consists of many nodes, which are called *artificial neurons*. All neurons are structured in layers. At first comes the input layer, which is a pseudo layer and different from the other layers. The number of neurons has the same dimension as the data that is fed into the network. Behind the input layer follows the hidden and the output layers.
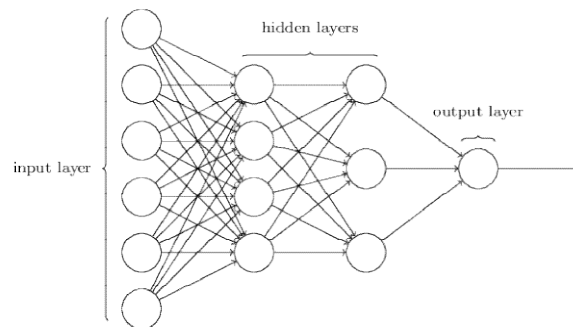

Figure 2: Layers in a Neural Network[3]

All neurons inside of them are working the same. Data is fed in the neuron and added to one value. This value is then inserted into an *activation function*. Activation functions output a value for every input they get. The outputs can be either binary or a real number. Inputs to the neurons are weighted. Every neuron (also the input layer neurons) output a value that is distributed to the next layer, usually to all nodes. Every connection that feeds output data into another node is weighted. These *weights* are the reason a network can learn. Initially, all weights are set to random values. The adjustment of the weights is called *training*.
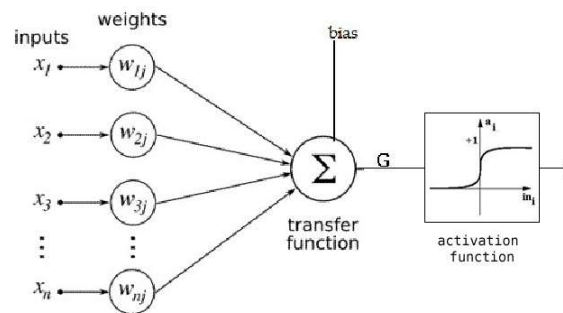

Figure 3: Artificial neuron[4]

---

[3] [Nielsen2015], Chapter 1: Using neural nets to recognize handwritten digits (accessed 14/04/2019).

After a dataset (in this case image) went through the neural network and the output is calculated, the output is compared to the label that is attached to the image. The difference is called *error*.

Backpropagation is the technology that trains the network after the error is calculated. Based on the learning rate and the epochs, the neural network tries to find the optimal values for all weights by walking backwards through the network.
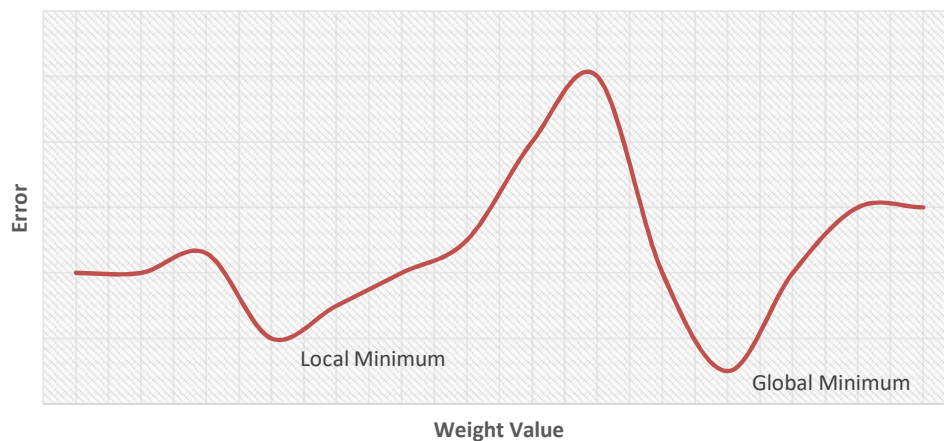
Error Function

Figure 4: Error function of a neuron

Optimal values for weights result in a minimal error. The backpropagation adapts the values in little steps to reach the minimal error. Only when the minimal error is reached, the neural network produces correct results. The size of the steps is determined by the function that is used in the backpropagation. The *learning rate* is added to the step size to prevent the training to get stuck in a *local minimum*. The error function can have multiple local minima. When the weight value would be of the local minima after the training, the neural network would not be precise. Only the *global minimum* leads to the desired results[5,6].

When using a neural network as classifier, several *problems* can occur. Neural networks are using reference data during the learning process. The more data, the better the result of the training. Not only the quantity but also the quality of data is important. When the data of a class is impure or does not distinguish itself from other classes, the neural net becomes very imprecise.

3.2 Methodology

With the machine learning library *TensorFlow*, a banana ripening state classifier is created. TensorFlow provides a Docker image that has all the required tools installed. The classifier is created with the help of a training script that is provided by TensorFlow. For the training, several images that are sorted into different subfolders, are needed. The script uses the names of the subfolders as labels for the images inside them. JPG and PNG images are accepted as inputs. As a standard value, 4,000 epochs are used and a learning rate of 0.01. 10 % of the data is used for validation and 10 % for testing. The network that is trained is a network from the TensorFlow hub called *Inception-v3*, which is specially designed for image classification. The model is a kind of neural network, but has several optimisations for image recognition.

---

[4] See https://mc.ai/the-activation-functions/ (accessed 14/04/2019).
[5] [Nielsen2015], Chapter 2: How the backpropagation algorithm works (accessed 14/04/2019).
[6] [Loy2018], How to build your own Neural Network from scratch in Python.

The training outputs a *labels file* containing the trained labels and a *graph file* that contains the classifier. When the graph outputs a value, the labels file can be used to map the labels to a human readable name. After the classifier is trained, a second script is used for testing. Images that are not part of the initial training set can then be used to test the quality of the classifier. As a result, the script outputs a decimal value for each of the possible result classes. The class highest number is the result of the input image. If this fulfils the expectations for a lot of images, the classifier works well. More information on that can be found in the GitHub repository https://github.com/peltzefa/nn_training_docker.

3.3 Training the Classifiers

The first step is to create a dataset for the training. This is done by getting images from the Google image search, followed by a manual classification based on the criteria described in the "Bananas" chapter. The result of the time-consuming manual classification process is three folders as shown in Figure 5.

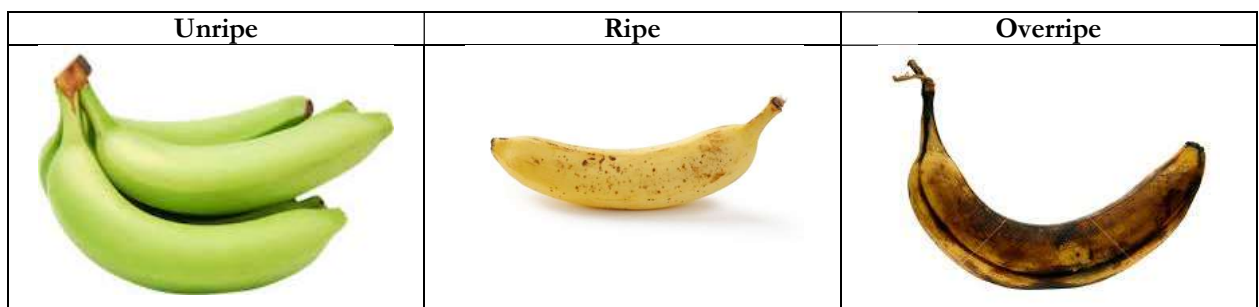| Unripe	Ripe	Overripe

Figure 5: Example data in classified image sets

Every image set contains around one hundred pictures. The training (4,000 epochs, training rate 0.01) takes between 10 and 15 minutes and creates an output graph and a labels file. Every training step creates an output.

```
I0413 10:28:15.486490 139858899928832 retrain.py:1103] 2019-04-13 10:28:15.486428: Step 3970:
Train accuracy = 100.0%
INFO:tensorflow:2019-04-13 10:28:15.486665: Step 3970: Cross entropy = 0.060070
I0413 10:28:15.486681 139858899928832 retrain.py:1105] 2019-04-13 10:28:15.486665: Step 3970:
Cross entropy = 0.060070
INFO:tensorflow:2019-04-13 10:28:15.548356: Step 3970: Validation accuracy = 64.0% (N=100)
```

Figure 6: Section of training log version 1

Although the validation accuracy is not very good and the testing accuracy is at 100 %, which means that the network is overfitted, the six test images are fed into the network for testing. Instead of using the network in the final Android application, the testing script is used.

Image 1: ripe	Image 2: overripe	Image 3: unripe

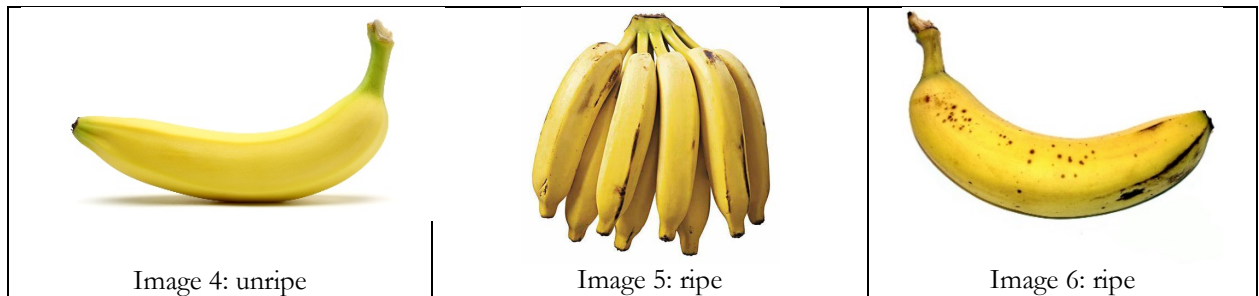| Image 4: unripe | Image 5: ripe | Image 6: ripe |

Figure 7: Test images of bananas

Results of classifier version 1:

Image 1: unripe 0.5017895, ripe 0.489748, overripe 0.008462544
Image 2: overripe 0.95115256, unripe 0.028853523, ripe 0.019994011
Image 3: unripe 0.9942332, ripe 0.005750307, overripe 1.6487336e-05
Image 4: ripe 0.8638092, unripe 0.13448384, overripe 0.0017069471
Image 5: ripe 0.6114174, unripe 0.3830901, overripe 0.005492488
Image 6: ripe 0.93043524, unripe 0.035402395, overripe 0.03416232

The results for the images 2, 3 and 6 are good enough for successful classification. Image 1 and 5 are not clear enough. The result for image 4 is false. The average validation accuracy was around 65 %.

The next training is based on the same images, but during the training, the script flips around half of the images by 90 degrees to increase the variation of the images. Also, the epochs are decreased to 1,000. This is done because of the computational intense image rotation which is probably affecting the training duration enormously. The other settings are the same as before.

```
INFO:tensorflow:2019-04-14 18:27:40.047457: Step 999: Train accuracy = 93.0%
I0414 18:27:40.047543 140663540672256 retrain.py:1103] 2019-04-14 18:27:40.047457: Step 999:
Train accuracy = 93.0%
INFO:tensorflow:2019-04-14 18:27:40.047786: Step 999: Cross entropy = 0.213686
I0414 18:27:40.047808 140663540672256 retrain.py:1105] 2019-04-14 18:27:40.047786: Step 999:
Cross entropy = 0.213686
INFO:tensorflow:2019-04-14 18:27:40.110069: Step 999: Validation accuracy = 76.0% (N=100)
I0414 18:27:40.110173 140663540672256 retrain.py:1124] 2019-04-14 18:27:40.110069: Step 999:
Validation accuracy = 76.0% (N=100)
```

Figure 8: Section of training log version 2

Results of classifier version 2:

Image 1: ripe 0.60269624, unripe 0.379709, overripe 0.017594725
Image 2: overripe 0.8488609, ripe 0.09400831, unripe 0.057130795
Image 3: unripe 0.97412986, ripe 0.025473239, overripe 0.0003968866
Image 4: ripe 0.7194154, unripe 0.26292732, overripe 0.017657291
Image 5: ripe 0.58646905, unripe 0.40532684, overripe 0.008204096
Image 6: ripe 0.8892048, unripe 0.059313156, overripe 0.051482074

The results for image 1 and 4 are false and image 5 became less clear than in the previous training. Overall, the precision is worse, even if the average validation accuracy increased. The biggest problem was the execution time of over three hours. The average validation accuracy was around 75 %.

10

The third training is done with the settings of training one and the images are changed before they are fed into the training process. The images are rotated left and right, mirrored vertically and horizontally and a border is added. From every image, five additional ones are created. This leads to 600 images per class instead of 100.

```
I0414 18:47:42.558793 139896281032448 retrain.py:1103] 2019-04-14 18:47:42.558704: Step 3999:
Train accuracy = 95.0%
INFO:tensorflow:2019-04-14 18:47:42.559072: Step 3999: Cross entropy = 0.168023
I0414 18:47:42.559093 139896281032448 retrain.py:1105] 2019-04-14 18:47:42.559072: Step 3999:
Cross entropy = 0.168023
INFO:tensorflow:2019-04-14 18:47:42.640013: Step 3999: Validation accuracy = 96.0% (N=100)
I0414 18:47:42.640121 139896281032448 retrain.py:1124] 2019-04-14 18:47:42.640013: Step 3999:
Validation accuracy = 96.0% (N=100)
```

Figure 9: Section of training log version 3

Results of classifier version 3:

Image 1: ripe 0.53403723, unripe 0.45865563, overripe 0.00730713
Image 2: overripe 0.9277073, ripe 0.05396977, unripe 0.018322913
Image 3: unripe 0.9983839, ripe 0.001605455, overripe 1.07051355e-05
Image 4: ripe 0.8593764, unripe 0.13914041, overripe 0.0014832125
Image 5: ripe 0.9407128, unripe 0.05830532, overripe 0.0009819551
Image 6: ripe 0.9592875, overripe 0.022224016, unripe 0.018488422

This time, images 1 and 4 are falsely recognised as ripe while they are unripe. The overall precision rises which resulted in higher values for the best results. The training duration was only four minutes and had an average validation accuracy of around 90 %.

At this point, it is time to look at the data that is used for the training. The only problem that existed continuously during all the trainings was the difference between ripe and unripe bananas. This could be caused by images that are too similar for the network to recognise them clearly.

From the unripe images, all bananas were removed, that had larger black spots on them. In the ripe image set, all bananas without any large black spots were removed. This is just possible because the dataset was blown up in the preparation step of the previous training. Without that artificial enlargement, the dataset would have become too small.

Results of classifier version 4:

Image 1: unripe 0.54014134, ripe 0.45006236, overripe 0.009796285
Image 2: overripe 0.95138705, ripe 0.036297567, unripe 0.012315433
Image 3: unripe 0.99930966, ripe 0.000680745, overripe 9.560022e-06
Image 4: ripe 0.7828451, unripe 0.21502283, overripe 0.0021320612
Image 5: ripe 0.9818424, unripe 0.017456755, overripe 0.0007008079
Image 6: ripe 0.97009134, overripe 0.020599488, unripe 0.009309138

While the validation rate did just become lightly bigger to just over 90 % and the training duration was about the same, it's just one false result left. Image 1 is now more correct than before, still not perfect, but better.

To create classifier number 5, the epochs were set to 10,000. At this point it would not make sense to change more data, this could lead to a training dataset, that is engineered to just classify the selected six test images correctly and this would be the wrong way.

Results of classifier version 5:

Image 1: unripe 0.65333074, ripe 0.34170565, overripe 0.004963606
Image 2: overripe 0.9818174, ripe 0.014765506, unripe 0.0034170772
Image 3: unripe 0.99996233, ripe 3.7266283e-05, overripe 3.4894268e-07
Image 4: ripe 0.814441, unripe 0.18513831, overripe 0.0004207134
Image 5: ripe 0.9952389, unripe 0.0045821695, overripe 0.00017894543
Image 6: ripe 0.9818419, overripe 0.014470144, unripe 0.0036880148

Image 4 is still falsely classified. The training took 10 minutes and had an average training accuracy of 95 %. The next training is performed with a bigger learning rate of 0.05 instead of 0.01.

Results of classifier version 6:

Image 1: unripe 0.74159855, ripe 0.25671983, overripe 0.001681662
Image 2: overripe 0.9979176, ripe 0.0019581674, unripe 0.00012424863
Image 3: unripe 0.9999999, ripe 1.542417e-07, overripe 4.785527e-10
Image 4: ripe 0.9095376, unripe 0.09043715, overripe 2.529819e-05
Image 5: ripe 0.99987423, unripe 0.00012118929, overripe 4.5269344e-06
Image 6: ripe 0.99302167, overripe 0.0066988794, unripe 0.00027954153

The only thing that changed is the accuracy of image 1 which went more precise. The last training increased the learning rate to 0.1.

Results of classifier version 7:

Image 1: unripe 0.80461556, ripe 0.19491084, overripe 0.00047365073
Image 2: overripe 0.9990741, ripe 0.0008820132, unripe 4.383398e-05
Image 3: unripe 1.0, ripe 4.3310915e-09, overripe 3.0523247e-12
Image 4: ripe 0.93253547, unripe 0.067462236, overripe 2.3774887e-06
Image 5: ripe 0.99999034, unripe 9.472842e-06, overripe 1.9263268e-07
Image 6: ripe 0.9984806, overripe 0.001467406, unripe 5.2032497e-05

The precision of image 1 still went up, also did the precision of image 4.

Only the trainings that had a noticeable effect are shown and discussed here. Looking back and seeing that the classification of image 4 was always ripe, could also be seen as a human error in classification. Whether this is caused by the selection of the training images or the human classification of the test image, the fourth image does not belong to the unripe class. It should be part of the ripe class. The high result value of the ripe class confirms that. No matter what parameters where changed, the biggest value was always outputted in the ripe class.

Image four behaves a bit different from image number one. The banana on the image has little black spots, which are a sign for a ripe banana. But the colour of the stalk is still green. Removing the colour information changed the results, but ripe had still the highest percentage. The next step was to remove the little black spots, which had almost no effect on the results. Probably because the images are downsized during the process and the small dots are lost.

Preview, black and white	Cropped preview, original	Cropped preview, no dots
ripe 0.677468 unripe 0.3033392 overripe 0.019192815	ripe 0.93253547 unripe 0.067462236 overripe 2.3774887e-06	ripe 0.9322824 unripe 0.06771397 overripe 3.655683e-06

Figure 10: Manipulated banana image 4, dots removed, colour removed

The change of the colour had the biggest impact on the results. To prove this, image 4 is manipulated further. With the help of Adobe Photoshop, the stalk and the end of the banana were darkened and the colour changed from light green to a dark yellow. This raised the precision of the classifier to 99 % ripe. To take this a step further, the banana from image 4 was dyed green. As a result, the classifier recognises the banana with 95.5 % percent as unripe. This proves, that the classifier not only uses texture to recognise the ripeness grade of the banana, it also uses the colour of the banana.

Colour can be hard to recognise under different lighting conditions. Therefore, the classifier could be false when the colour of the light is cool. This is a problem that has to be solved in the future.

Preview of a photoshopped, ripe banana	Preview of a photoshopped, unripe banana
ripe 0.9909157 unripe 0.00907342 overripe 1.0871911e-05	unripe 0.95513254 ripe 0.04486649 overripe 1.0119747e-06

Figure 11: Ripeness grade of banana 4 changed

4   Android App Development

The BananaCo application is implemented as app for Android. The workshop provided a reference Android App called "Smartphone Sensing Framework" which was already capable of monitoring various sensors, such as acceleration, orientation and magnetic fields as well as displaying a continuous video stream from the camera with a neuronal network as classifier. The continuous video stream with classifier was used as foundation for the BananaCo Android app. Although Java is used for writing Android apps, Android enforces additional requirements. For a basic understanding, this chapter will cover some of those.

4.1   Manifest

The file "AndroidManifest.xml" contains relevant information for the operating system. For example, required permissions, as well as required features are listed here. In addition, all invocable services and activities need to be listed. Otherwise those services and activities cannot be used at runtime.

In case of BananaCo, the permission to access the camera is listed as "uses-permission" and the main activity is listed as entry-point for the application.

4.2   Activity Lifecycle

In Android, the class inside the app responsible for handling the UI is called "Activity". Besides displaying the UI and handling user input, it must also adhere to the activity lifecycle (see Figure 12). To implement custom behaviour, one needs to write a class that extends the "Activity" class and overwrite methods where a custom behaviour is desired.
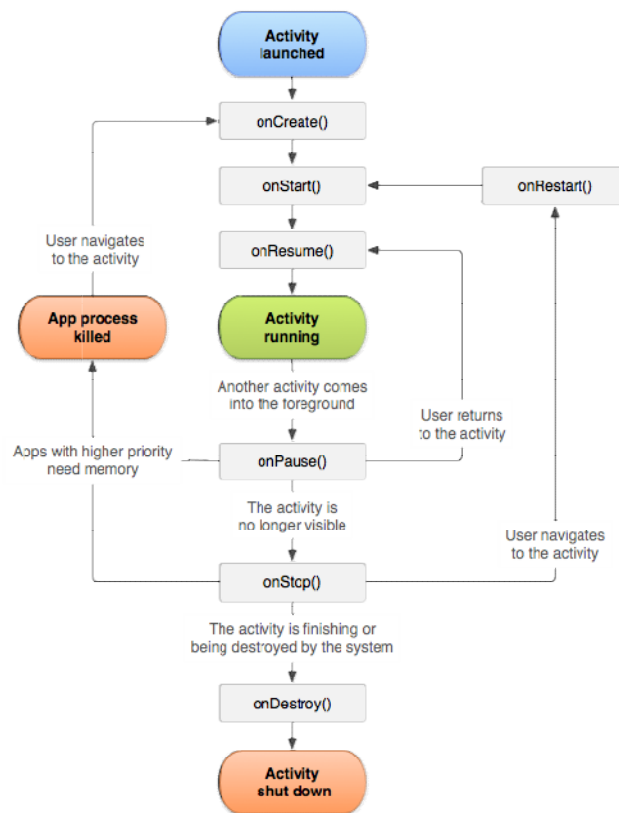

Figure 12: Android application activity lifecycle
(Source: https://developer.android.com/guide/components/activities/activity-lifecycle)

As the graph for the Activity Lifecycle shows, there are multiple steps before and after the running state. The steps before (onCreate, onStart and onResume) are used to prepare or re-activate resources required in the running state. For example, at the onCreate stage and in the ImageDetection activity, the connection to the camera is initialized but not activated before onResume was called.

The call-backs onPause, onStop and onDestroy might be invoked after the running stage to tell the application to free resources or to stop processing data. For example, the onPause call-back in the BananaCo application ensures that no more images from the camera are retrieved or classified to reduce battery usage, while the connection to the camera is not destroyed before onStop.

The method call-backs onCreate and onDestroy mark the lifetime boundaries of an activity and are trivial in regards of pre- and post-conditions. If the method onCreate is invoked, one can be sure that no other call-back was invoked yet. Once onDestroy is invoked, one can be sure that the activity will not receive any further notifications. All other invocations might occur more often and under other circumstances.

Once an app is no longer visible to the user, the method onStop is called, which allows the app to free all UI-relevant resources and to stop foreground tasks. It also allows the app to actively maintain remaining tasks as background tasks, such as continuing to download game assets.

A call of onRestart followed by onStart allows the application to re-allocate UI-relevant resources and to prepare for user interactions. The method onPause tells the application that it is currently no longer in the foreground (another activity or dialog is hiding it) and cannot be seen by the user. It is useful to pause foreground tasks to reduce resource usage until onResume is called, because those results are not visible to the user.

4.3   Android Layout Definitions

For Android apps it is common to have UI elements defined in a resource file, a so-called layout XML file. For each element, depending on the chosen layout, the position and size, default texts or values and IDs can be set. The activity will load this layout definition in the onCreate method and retrieve UI elements through their IDs.

This allows great flexibility, because the layout details can be adjusted without interfering with the UI logic. In addition, Android offers the possibility to silently load different XML files depending on the screen orientation if provided. This allows one to easily adjust or totally change the layout depending on the screen orientation without having to change anything in the activity, as long as the IDs of all UI elements remain the same.

5   Graphical User Interface

This chapter is split into two sections. On the first section, the initial mock-up idea is displayed and discussed. The second section will present the final result with the differences to the mock-up and the reasons for the changes.

5.1   Mock-up

The mock-up can be seen in Figure 13. The main part of the opened Android app is dedicated to the camera live feed. This allows the user to see what is being judged. Below the live feed, two indicators were planned. The first one was supposed to indicate whether the app recognised a banana in the image. The second indicator was supposed to indicate whether the banana is edible. As visible in Figure 14, the indicators used are two basic shapes: a green check mark and a red cross. With these shapes, the following total combinations can be displayed:
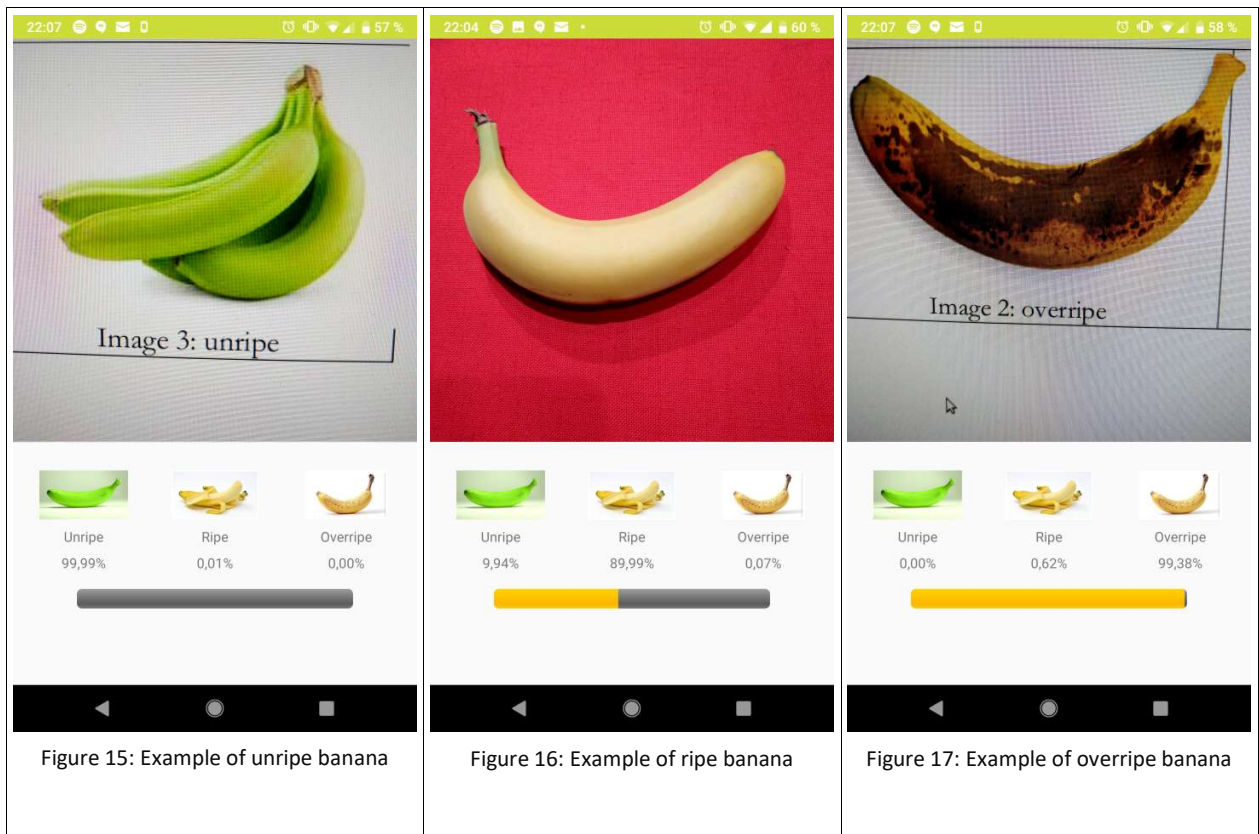

Figure 13: Initial, simple GUI design

Indicator 1	Indicator 2	Meaning
✗ Red cross	✓ ✗ Doesn't matter	No banana detected
✓ Green check mark	✗ Red cross	Banana detected, but do not eat
✓ Green check mark	✓ Green check mark	Banana detected, can be eaten

Figure 14: Indicator description

5.2   Final Result

The final implementation (see table below, figures 15-17) also shows the live feed of the camera, but no longer uses the two indicators from the mock-up to display the ripeness value. While investigating how to train the network (see chapter 3, "Neural Network") we discovered that the network can be trained to judge the ripeness of the banana more precisely than "edible" and "not edible". Therefore, the indicator displays the ripeness value now, using the results "unripe", "ripe" and "overripe" from the neuronal network. These values are displayed with an explanation and an example image in three separate columns. Finally, a merged result is displayed in the form of a progress bar.

| Figure 15: Example of unripe banana | Figure 16: Example of ripe banana | Figure 17: Example of overripe banana |

Starting on the left side, the first column shows the gauged "unripe" value, the second column shows the gauged "ripe" value and the third the "overripe" gauged value. These values are used to indicate the lifetime of the banana in the merged progress bar.

A low progress bar, ending near the first column "unripe" (see Figure 15) indicates an unripe banana, while an end near the second column indicates "ripe" (see Figure 16) and an ending near the third column indicates "overripe" (see Figure 17). This allows the user to grasp the current state of the banana at first sight.

6 Operating Principle

As mentioned in chapter 5.1 in the early stages, the app was supposed to have an object detection before the ripeness of the banana is judged. This change did not only surface in the UI differing from the UI mock-up but also the flowcharts.
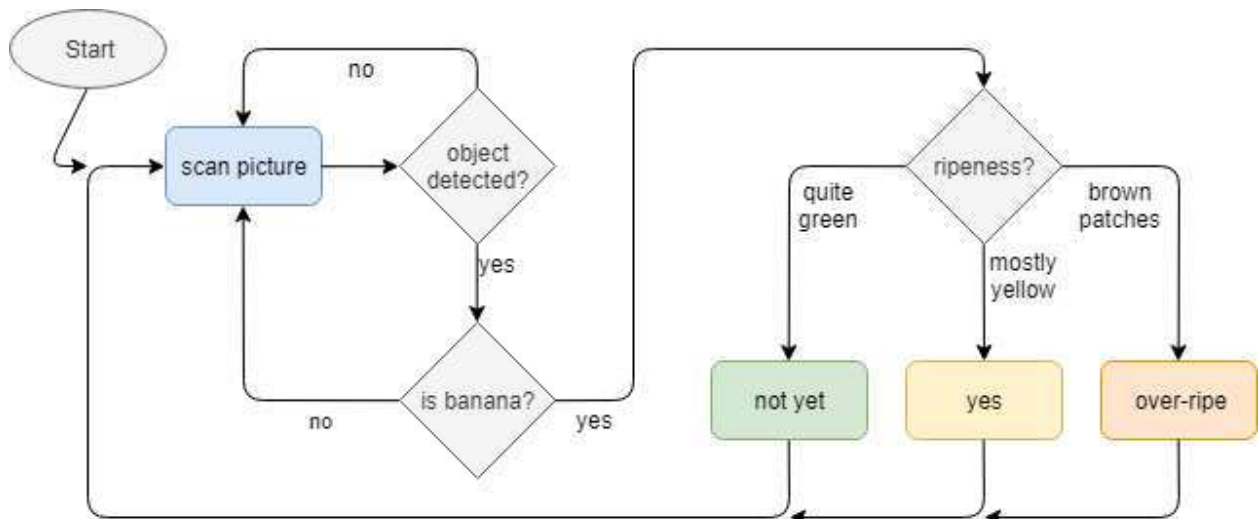

Figure 18: Operation principle, initial design

Figure 18 shows the early flowchart draft while **Error! Reference source not found.** shows the flow chart of he implemented app. In addition to the removal of the object detection, a new step has been added. The "cooldown" step is supposed to reduce the average load over time on the CPU of the smartphone.
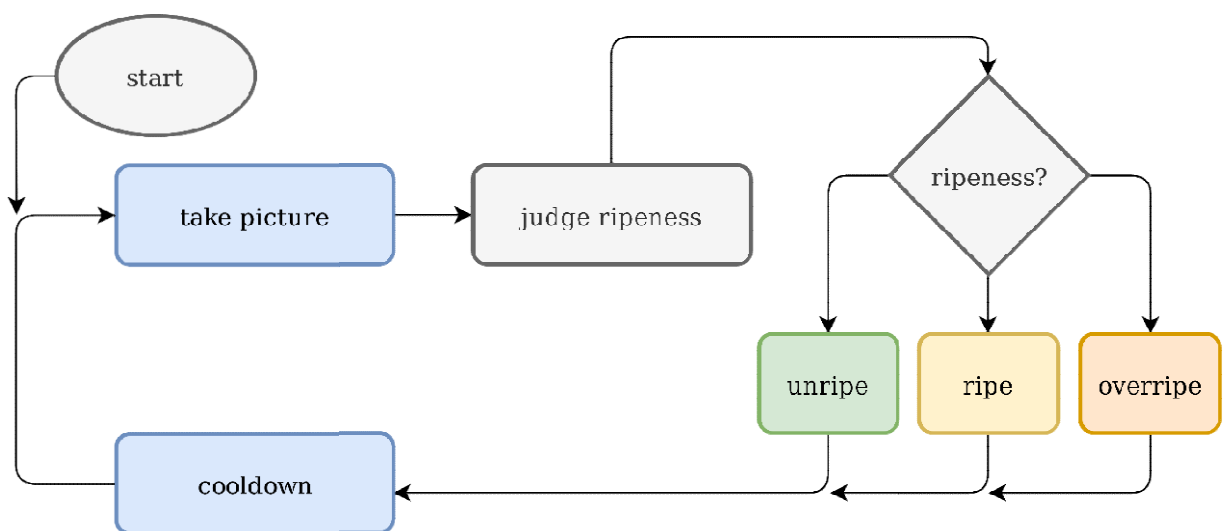

Figure 19: Final operation principle

This shall prevent overheating and CPU throttling as well as increasing battery lifetime. This change was a result of early tests, where the smartphone got uncomfortably warm and the battery was drained although the smartphone being connected through USB to the computer.

The following listing shall explain each step in more detail:

- "take picture": Takes the current picture from the camera live feed.
- "judge ripeness": Takes the selected picture and feeds it to the neuronal network to retrieve a judgment for the ripeness.
- "ripeness", "unripe", "ripe" and "overripe": Depending on the results, update the UI.
- "cooldown": Idle for a bit (500 ms) to allow the smartphone to cool down

7 Conclusion

7.1 Summary

The original expectations of the resulting classifier application were definitely met, and actually outperformed in a specific aspect.

Expectations ranged somewhere between "may work somewhat correctly" and "could actually work quite well", which it then actually did: classification accuracy (detecting the right ripening phase) with photographs of bananas as well as with real world recordings of bananas produced by the smartphone camera was nearly perfect. The probabilities given to the distinct classifications usually lay between 90 % and 100 %.

Most noticeably, the background of the image (everything around the actual banana to classify) did not noticeably affect the classification, which is very beneficial for the initial use-case: identifying banana ripeness(es) while grocery shopping, where the background may naturally differ greatly in every image.

The actual colour of the banana turned out to be the most prevalent factor for the classifier to analyse the given images. This prevalence and the issue that the ambient lighting of the image to analyse can, from time to time, significantly alter the recorded image colour information, lead to false classifications, which still is a problem to be solved in the future.

The chosen methodology, machine learning, proved to have some major advantages over the previously considered Computer Vision approach with object detection and colour filtering.

It was convenient to implement. Shape, size and colour detection could be integrated into a single process, as the neural network trained to distinguish ripeness stages basically calculated all those criteria simultaneously in one model.

Additionally, no in-depth knowledge on digital signal / image processing was necessary. In the end, the workshop concluded in a successfully functioning prototype.

7.2 Outlook

Several improvements and additional features may be considered for the application:

- provide the option to enter a preferred degree of ripeness to buy / eat;
- make a prediction on storage life of the banana analysed;
- give suggestions on usage of the banana, based on its current ripeness – e.g. whether to best cook, mix or directly eat it;
- add other fruit or vegetables to the classifier to broaden the field of application.

Appendix

The following example testing pictures can also be found on Team D's fork of MrDio's Smartphone-Sensing-Framework: BananaCo-Repository @ Github
in the subdirectory: **./BananaCo_test/**

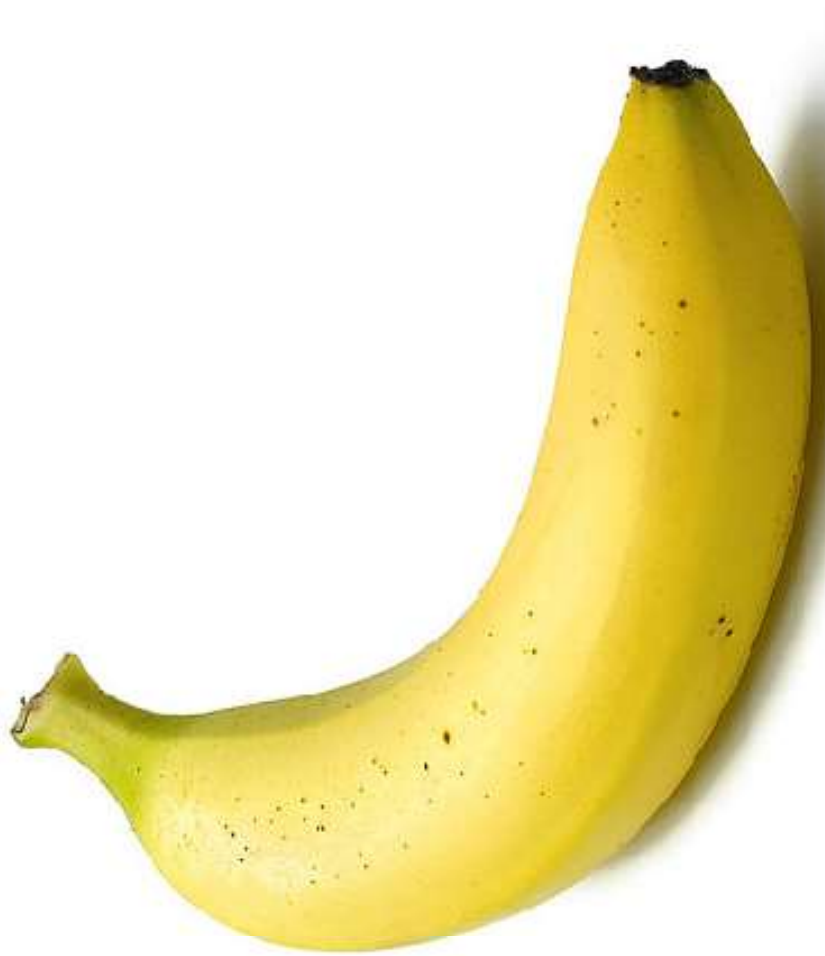Figure 20: Classifier testing image (1)

Figure 21: Classifier testing image (2)

Figure 22: Classifier testing image (3)

Figure 23: Classifier testing image (4)

List of abbreviations

Abbreviation	Explanation
BananaCo	"Banana Colour", the title of the project related to the undertaking of recognising the ripeness of fruit-bananas with the help of computer vision
DV	Daily vitamins
HSV	Hue Saturation Value colour model
RGB	Red Green Blue colour model

References

[Arivazhagan2010] *Selvaraj, Arivazhagan, Shebiah, Newlin, Nidhyananthan, Selva, Ganesan, Lakshmanan (2010)*, Fruit Recognition using Color and Texture Features. Journal of Emerging Trends in Computing and Information Sciences 1/2010, 90-94.

[Loy2018] *Loy, James (2018)*, How to build your own Neural Network from scratch in Python, https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6 (accessed 12/04/2019).

[Mazen2019] *Mazen, Fatma M. A., Nashat, Ahmed A. (2019)*, Ripeness Classification of Bananas Using an Artificial Neural Network. Arabian Journal for Science and Engineering, 1-10.

[Mendoza2005] *Mendoza, F., Aguilera, J. M., Dejmek, P. (2005)*, Predicting Ripening Stages of Bananas (Musa cavendish) by Computer Vision. Acta horticulturae 682, 1363-1370.

[Meng-Han2015] *Hu, Meng-Han, Dong, Qing-Li, Malakar, Pradeep K., Liu, Bao-Lin, Jaganathan, Ganesh K. (2015)*, Determining Banana Size Based on Computer Vision, International Journal of Food Properties 18(3) 2015, 508-520.

[Nelson2006] *Nelson, Scot C., Ploetz, Randy C., Kepler, Angela K. (2006)*, Musa species (banana and plantain). Species profiles for pacific island agro forestry 01/2006.

[Nielsen2015] *Nielsen, Michael A. (2015)*, Neural Networks and Deep Learning. Determination Press, 2015, http://neuralnetworksanddeeplearning.com/ (accessed 14/04/2019).

[Prabha2013] *Surya Prabha, D., Satheesh Kumar, J. (2013)*, Assessment of banana fruit maturity by image processing technique. Journal of food science and technology 52(3), 1316-27.

[Ware2017] *Ware, Megan (2017)*, Benefits and health risks of bananas. Medical News Today, 28 November 2017, see: https://www.medicalnewstoday.com/articles/271157.php (accessed 14/04/2019).