

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

Project leader <Student ID> …<First name><Surname>  
Developer(s) <Student ID> …<First name><Surname>  
 <Student ID> …<First name><Surname>  
Data-Analyst(s) <Student ID> …<First name><Surname>  
 <Student ID> …<First name><Surname>  
Documentation-Manager <Student ID> …<First name><Surname>

Deadline: 23rd April 2019

Content

[1 Introduction IV](#_Toc4260251)

[1.1 Initial Problem IV](#_Toc4260252)

[1.2 Proposed Solution IV](#_Toc4260253)

[2 Bananas V](#_Toc4260254)

[2.1 Introduction V](#_Toc4260255)

[2.2 Background V](#_Toc4260256)

[2.3 Maturity assessment V](#_Toc4260257)

[2.3.1 General criteria V](#_Toc4260258)

[2.3.2 Visual criteria V](#_Toc4260259)

[2.4 Classification and Feature Selection VI](#_Toc4260260)

[2.5 Methodology VI](#_Toc4260261)

[3 Neural Network VII](#_Toc4260262)

[3.1 Introduction VII](#_Toc4260263)

[3.2 Methodology VII](#_Toc4260264)

[3.3 Operating Principle VII](#_Toc4260265)

[4 Graphical User Interface VIII](#_Toc4260266)

[4.1 Mock-up VIII](#_Toc4260267)

[4.2 BananaCo App UI VIII](#_Toc4260268)

[5 Operating Principle IX](#_Toc4260269)

[5.1 Introduction IX](#_Toc4260270)

[5.2 Flowchart IX](#_Toc4260271)

[6 Conclusion X](#_Toc4260272)

Figure Index

[Figure 1: Sample Figure 1](#_Toc4181395)

[Figure 2: Banana ripe stages as used for BananaCo classification, from left to right: unripe, ripe, overripe 3](file:///D:\Projects\Smartphone-Sensing-Framework\BananaCo_report\BananaCo_report.docx#_Toc4181396)

Abstract

<TODO: 1 page>

# Introduction

**Feedback Dio & Dimi:**

TODO1: Begründung warum kein “einfaches” Color-Filter verwendet; warum keine vorgeschobene Bilderkennung via computer vision => Grund: Zusammenführung von Shape / Size und Farberkennung in einem Model / einer App

TODO2: Farbe-Reifegrad / Zuckergehalt Korrelation darstellen und z.B. Farbabstufungen auf Papier ausdrucken, damit der Leser dies mittels bereitgestellter App auch direct testen kann

TODO3: aus dem aktuellen Reifegrad Empfehlungen ableiten für:

1. die weitere geeignete Verwendung bzw. Verarbeitung (z.B. Verkochen, Drink / Mixer, direkter Verzehr) sowie
2. zur voraussichtlichen weiteren Haltbarkeit (i.S.v. „MHD“)

## Initial Problem

In real-world, detection of ripeness / maturity of banana-fruits in real-time

Target groups:

* industry
* retailer,
* end users,
* persons with disabilities (e.g. red-green deficiency) or sugar intolerances (e.g. diabetes)

## Proposed Solution

BananaCo – Banana colour: Detect objects or rather bananas via image recognition and output result, so that users get a definite statement on the ripeness.

# Bananas

## Introduction

In general, a distinction must be made between *plantains* and *fruit-bananas*. In the context of *BananaCo* project, only the latter is taken into consideration. Both originate from tropical regions, predominantly in Africa and South America. 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.

## Background

The maturity stage of fresh banana is important in marketing, for both, industry and dealers as well as end consumers. 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.

In the meantime, 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, competitivity, quality standards, and reliability of banana fruit products, automatic image processing tools based upon intelligent techniques are paramount over visual features methods.” [Mazen2019]

## Maturity assessment

### 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 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).

### Visual criteria

In literature, a lot of methods 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 product representative results.

BananaCo on the contrary focuses onto automated visual, i.e. image recognition using smartphone cameras. Also, computer aided analysis techniques are utilised, 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.

## Classification and 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[[1]](#footnote-1) 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 with according feature aspects (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, at least 50 % of peel surface | brown |

Table 1: Maturity categories

## Methodology

The criteria listed before is used later to manually categorise banana images acquired from the internet into 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 / neural network, serving as training data.



unripe ripe overripe

Figure 1: Banana ripe stages as used for BananaCo classification

# Neural Network

## Introduction

Recognising the ripening state of a banana is a classification problem. Neural networks are made for this kind of problem. A classification problem has input data and an output. 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 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 out 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 is 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 come the hidden and the output layers.



Figure 2: Layers neural network[[2]](#footnote-2)

All neurons inside of them are working the same. Data is fed in the neuron and is added to one value. This value is then inserted into an activation function. Activation functions outputs 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 of the 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[[3]](#footnote-3)

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 of the weights, by walking backwards through the network.

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 minimums. When the weight value would be of the local minimums after the training the neural network would not be precise. Only the global minimum leads to the desired results.

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. But not only the quantity, also is the quality of data important. When the data of a class is impure or does not distinguish itself from other classes the neural net becomes very imprecise.

## 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 us provided by TensorFlow. For the training a number of 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’s and png’s are accepted as inputs. As a standard value 4000 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 TensorFlow hub called Inception-v3, which is specially designed for image classification.

The training outputs a labels file with 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 for the input image. If this fulfils the expectations for a lot of images the classifier works well. More information to that can be found in the github repository <https://github.com/peltzefa/nn_training_docker>.

## Training the Classifiers

The first step is to create a dataset for the training. This is done by getting images form the google image search and a manual classification based on the criteria described in the Bananas chapter. The result of the time consuming manual classification process are three folders like shown in Figure 5.

|  |  |  |
| --- | --- | --- |
| **Unripen** | **Ripe** | **Overripe** |
| C:\Users\Fabian\AppData\Local\Microsoft\Windows\INetCache\Content.Word\001unripen.jpg | C:\Users\Fabian\AppData\Local\Microsoft\Windows\INetCache\Content.Word\016ripe.jpg | C:\Users\Fabian\AppData\Local\Microsoft\Windows\INetCache\Content.Word\022overripe.jpg |

Figure 5: Example data in classified image sets

Every image set contains around one hundred pictures. The training (4000 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 over fitted, the 6 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: unripen | ^C:\Users\Fabian\AppData\Local\Microsoft\Windows\INetCache\Content.Word\banana_2.png  Image 2: overripe | C:\Users\Fabian\AppData\Local\Microsoft\Windows\INetCache\Content.Word\banana_3.jpg  Image 3: unripen |
| C:\Users\Fabian\AppData\Local\Microsoft\Windows\INetCache\Content.Word\banana_4.jpg  Image 4: unripen | C:\Users\Fabian\AppData\Local\Microsoft\Windows\INetCache\Content.Word\banana_5.jpg  Image 5: ripe | C:\Users\Fabian\AppData\Local\Microsoft\Windows\INetCache\Content.Word\banana_6.jpg  Image 6: ripe |

Figure 7: Test images bananas

Results of classifier version 1:

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

The results for the images 2, 3 and 6 are good enough for a 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 half of the images around 90 degrees to increase the variation of the images. Also the epochs are decreased to 1000. 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, unripen 0.379709, overripe 0.017594725**Image 2: overripe 0.8488609, ripe 0.09400831, unripen 0.057130795  
Image 3: unripen 0.97412986, ripe 0.025473239, overripe 0.0003968866  
**Image 4: ripe 0.7194154, unripen 0.26292732, overripe 0.017657291**  
Image 5: ripe 0.58646905, unripen 0.40532684, overripe 0.008204096  
Image 6: ripe 0.8892048, unripen 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. Over all the precision is worse, even if the average validation accuracy got bigger. The biggest problem was the execution time of over 3h. The average validation accuracy was around 75%.

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

These time image 1 and 4 are falsely recognised as ripe while they are unripe. The overall precision is risen which resulted in higher values for the best results. The training duration was only 4 minutes and had an average validation accuracy 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 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: unripen 0.54014134, ripe 0.45006236, overripe 0.009796285  
Image 2: overripe 0.95138705, ripe 0.036297567, unripen 0.012315433  
Image 3: unripen 0.99930966, ripe 0.000680745, overripe 9.560022e-06  
**Image 4: ripe 0.7828451, unripen 0.21502283, overripe 0.0021320612**Image 5: ripe 0.9818424, unripen 0.017456755, overripe 0.0007008079  
Image 6: ripe 0.97009134, overripe 0.020599488, unripen 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 number one is now more correct than before, still not perfect, but better.

To create classifier number 5 the epochs were set to 10000. 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 6 test images right and this would be the wrong way.

Results of classifier version 5:

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

Image 4 is still falsely classified. The training took 10 minutes and hat 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 5:

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

Image 1: unripen 0.80461556, ripe 0.19491084, overripe 0.00047365073  
Image 2: overripe 0.9990741, ripe 0.0008820132, unripen 4.383398e-05  
Image 3: unripen 1.0, ripe 4.3310915e-09, overripe 3.0523247e-12  
**Image 4: ripe 0.93253547, unripen 0.067462236, overripe 2.3774887e-06**  
Image 5: ripe 0.99999034, unripen 9.472842e-06, overripe 1.9263268e-07  
Image 6: ripe 0.9984806, overripe 0.001467406, unripen 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 were 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 has been during the selection of the training images or the classification of the test image, the fourth image does not belong to the unripen 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.

# Graphical User Interface

## Mock-up

My text…

## BananaCo App UI

My text…

.

# Operating Principle

## Introduction

My text…

## Flowchart

My text…

# Conclusion

* Expected results vs. actual
* Prediction accuracy
* Chosen methodology
* Computer vision vs. manual / instrumental
* Possible extensions / improvements
* outlook

Appendix

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 … |
| HSV | Hue Saturation Value colour model |
| RGB | Red Green Blue colour model |
|  |  |

References

[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.

[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.

Template:

* template: \_ws5\_listing
* Listing
* Figure

Number

1. template: \_ws5\_number
2. number a
3. number b[[4]](#footnote-4)
4. number c[[5]](#footnote-5)

Template of figure: \_ws5\_figure

Figure 10: Sample Figure

1. 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. [↑](#footnote-ref-1)
2. http://neuralnetworksanddeeplearning.com/chap1.html [↑](#footnote-ref-2)
3. https://mc.ai/the-activation-functions/ [↑](#footnote-ref-3)
4. [footnote 1] [↑](#footnote-ref-4)
5. [footnote 2] [↑](#footnote-ref-5)