Fruit and Vegetable Identification Using Machine Learning

Adam Olsson, Frida Femling

Degree Project in Computer Engineering, 15 credits

Halmstad 2018-06-03
Abstract

This report describes an approach of creating a system identifying fruit and vegetables in the retail market using images captured with a video camera attached to the system. The system helps the customers to label desired fruits and vegetables with a price according to its weight. The purpose of the system is to minimize the number of human computer interactions, speed up the identification process and improve the usability of the graphical user interface compared to existing systems. To accomplish creating a system improving these properties, an idea of implementing machine learning to identify the products aroused. Instead of assigning the responsibility to the user, who usually identify the products manually, the responsibility is given to a computer.

To classify an object, different convolutional neural networks have been tested and retrained. The networks have been retrained on data sets collected from ImageNet. To improve the accuracy, the networks have also been retrained on images where the background environment is similar to the environment the networks are supposed to perform in. The networks tested in this report are MobileNet and Inception. The networks have different propagation time and varies in accuracy. MobileNet performs the classification about seven times faster than Inception, but Inception gives more accurate results.

To improve the systems further, usability testing has been performed on the graphical user interface of existing system and resulted system. To test the usability, a heuristic evaluation has been performed in combination of a second test produced by the authors. The tests concluded that the resulted system was more user friendly compared to existing systems.

The hardware of the system constitutes of a Raspberry Pi, camera, display, load cell and a case. The software includes Python-code to label an image, a graphical user interface to interact with the user and a server created with Node.js. The graphical user interface has been programmed with JavaScript supplemented with the React library.

To conclude, implementing convolutional neural networks to classify images and developing a new user interface resulted in a faster identification process together with fewer usability flaws.
Acknowledgements

We would like to acknowledge our supervisor Fernando Alonso-Fernandez for great inputs and guidance on the way throughout this project.

Furthermore, we would like to acknowledge the users who made it possible to evaluate the usability in the systems.

Lastly, we would like to acknowledge Halmstad University for providing us with the hardware and Fhäré-fund for sponsoring this project.

Frida Femling & Adam Olsson
Students of Halmstad University
Halmstad, May 2017
# Contents

1 Introduction 1
  1.1 Purpose ................................. 1
  1.2 Goals and Specification .................. 2
  1.3 Questions at Issue ....................... 3
  1.4 Delimitations ........................... 3

2 Background and Theory 5
  2.1 Related Work ........................... 5
  2.2 Usability ................................. 7
  2.3 ImageNet ................................. 8
  2.4 Convolutional Neural Networks ............ 8
  2.5 Transfer Learning ........................ 10
  2.6 Architectures ............................ 12

3 Methodology 17
  3.1 Tools .................................. 17
  3.2 Implementation ........................... 20
  3.3 Network Evaluation ........................ 22
  3.4 System Testing ........................... 23

4 Results 25
  4.1 Network Architectures .................... 25
  4.2 System Design ............................. 31
  4.3 Usability ................................ 34

5 Discussion 39
  5.1 Results .................................. 39
  5.1.1 Usability Evaluation .................... 39
  5.2 Further Development ...................... 41

6 Conclusion 46

References 48

Appendices 51
Glossary

BGD Batch Gradient Descent. 11, 14, 19
BN Batch Normalization. 11–13
CMC Cumulative Matrix Curve. 27–29
CNN Convolutional Neural Networks. 2, 5, 6, 17

computer vision A research field of where computers are made for understanding visual data. 1

CPU Central Processing Unit. 17
CSI Camera Serial Interface. A port that defines an interface between a camera and a host processor. 17
CSS Cascading Style Sheets. 22
DOM Document Object Model. 8
DSI Display Serial Interface. A port that defines an interface between a display and a host processor. 17

GD Gradient Descent. 11
GUI Graphical User Interface. 2, 7

HCI Human-Computer Interaction. 1
HTML Hypertext Markup Language. 8
HTTP Hypertext Transfer Protocol. 18
JSX JavaScript XML. 8, 32, 33

LAN Local Area Network. 17
RAM Random Access Memory. 17
SGD Stochastic Gradient Descent. 11, 19
SMART Specific, Measurable, Acceptable, Realistic, Timed. 2

XML eXtensible Markup Language. 8
Chapter 1

Introduction

For decades, humankind has tried to recompose her technological equivalent when it comes to intelligence, feelings and behaviour. She has accomplished creating computers capable of performing various tasks better than many individuals. While a robot computes prime factorization in seconds, it takes weeks, or even years, for a human to accomplish the same task. However, humans are superior when it comes to abilities like understanding social codes and identifying objects, thanks to our natural intelligence. Software implemented with artificial intelligence, like the intelligent assistant Siri\footnote{https://www.apple.com/ios/siri/}, struggles to notice verbal sarcasm, jokes and feelings, although they can provide any user with historical facts fetched over the internet within seconds.

A single glance of an image is sufficient for a human to extract essential information about what is in the photograph. In seconds, the human brain has discovered several shapes and patterns out of an image. In comparison to a computer, it is in the nature of humans to analyze and identify visual data. To get a computer to identify objects in an image with sufficient accuracy, it has to be fed and trained with thousands or even millions of samples. To get an image classifier to be able to recognize objects is therefore a huge challenge.

The idea of implementing image classification in the self-service systems in the retail business were born by experiencing bad usability in existing systems. A thought of decreasing the Human Computer Interaction (HCI) time and give the identification task to a computer gave the idea of implementing image recognition in self-service systems.

1.1 Purpose

The purpose of this project is to improve the identification process of fruit and vegetables performed by the self-service systems in the retail market. More specifically, the improvement should consist of a faster process and a more user friendly system. The purpose of implementing computer vision to the system is to narrow the selection of possible objects and thus reduce the strain on the user. Additionally, the use of computer vision in self-service systems can simplify the process of identifying objects by moving the process from a human to a computer. Theoretically, this could hasten the process to identify products and minimize the amount of errors by removing the human factor.

Complex and time consuming self-service systems may result in customers choosing another grocery store. Since customers are the reason companies survive, their satisfaction is the businesses’ key to success. The necessity of systems which decreases the process time exists because of consumers’ expectations of their constant endeavor to save time.
Furthermore, in the perspective of the business, an important reason to improve the self-serving systems is to maintain the idea of work reduction. Since the systems exist to reduce work which leads to cost reduction, companies benefit from having user friendly systems. If organizations use time consuming systems, their customers may not use them at all and the idea of time efficiency and work replacement disappears.

1.2 Goals and Specification

Goals

The goals set for this project follow the SMART-model. This model is used to set up goals and to get an easy evaluation of the goals when the deadline is reached. SMART stands for Specific, Measurable, Attainable, Realistic, Timely.

1. Shorten the time it takes to identify a product

2. Minimize the number of usability problems which occur in existing systems

1. This is the main goal of the project and fills the criteria for the SMART-model. The goal could be measured by comparing the time it takes to identify a product with existing systems and the system implemented with image classification. It is acceptable since it is a motivating task, built on own experiences. To reach the goal, general knowledge about programming is required which means the goal is realistic. It is timed since the project includes a deadline.

2. Since this goal is difficult to measure, a heuristic evaluation could be performed on the new system and compared to the existing product. By analyzing users during the human-computer interaction and comparing number of problems according to the heuristic evaluation-method, it could be easier to measure if the goal is reached or not. The goal is specific, acceptable and will follow the specified time plan. To reach this goal, some knowledge about usability engineering is required.

Specification

Product and Requirements

The final product should consist of a classifier which identifies fruits and vegetables. The classifier will perform its estimations on a processor. The processor will be connected to a camera which is used to provide the software with images. The processor is also connected to a display. The display will show a graphical user interface (GUI) for the user to interact with and present the classifiers output. For classification purposes, we will investigate convolutional neural networks (CNN) architectures, given the huge success shown in recent years by CNN’s in several object recognition and classification tasks. Figure 1.1 displays the ideal result.

Functional Requirements

Functional requirements define what a system is supposed to perform.

- The system should, with the help of an image classifier and a camera, identify at least five different number of products
- The system should have a GUI for the user to interact with
- The system should be able to fetch products from a database
- The system should include an activator to activate the software
Non-functional Requirements

Non-functional requirements define how a system is supposed to be and includes usability, capacity, and availability.

- The usability of the system should be improved compared to existing systems.
- The time to identify a product should not exceed 5 seconds.
- The process to identify a product should require less than five interactions from the user.

1.3 Questions at Issue

To accomplish the goals and fulfill the specification, following questions will be investigated:

- Is the Raspberry Pi suitable for managing neural networks?
- Which CNN is suitable for the system?
- Which problems occur with chosen CNN?
- How do we handle a situation where the classifier provide incorrect results?
- What flaws of the usability was found in existing systems and reduced in resulted system?

1.4 Delimitations

The project will be limited to the fundamentals of the identification system. In terms of hardware, the fundamentals are a camera, a display, an activation mechanism representing the scale in the ideal system, and a processor to run the system. A user interface has to be developed to handle user interaction via the display. An image classifier has to be trained to classify images from the camera. All types of a fruit or vegetable reside under the same class. This means that the system will not distinguish between different colors of the same fruit or vegetable. Finally, a database has to be integrated to hold the stock of the retail business. It is important to note that the image classifier will not be written by the project group. Instead, existing classifiers will be compared to each other and the most suitable will be integrated with the system. Furthermore, the system will exclude a label printer but simulate the process in the graphical user interface.
Chapter 2

Background and Theory

Since the project is based on the idea of improving the usability of the user interface in existing systems and decrease the number of human-computer interactions, knowledge about usability engineering in combination of programming insight is preferable. Some theory about mathematical operations, which is the basis for training the CNN, is required to understand a part of the image classification process. In this chapter, essential theory and background information to accomplish this project is presented.

2.1 Related Work

Identification of fruits and vegetables are implemented in different areas. The most common areas are identification in the retail business and in areas where the purpose is to ease the harvest in the perspective of agriculture. Mostly, the identification is done manually by a cashier or via the self-service systems in a store. In this section, different methods of identifying fruits and vegetables will be presented.

Amazon Go

A company which has made great progress in its technical evolution when it comes to artificial intelligence, image recognition and automating physical work is Amazon\textsuperscript{1}. Amazon developed a product, called Amazon Go\textsuperscript{2}, which enabled a shopping experience without cashiers or self-service checkouts. The company built the store where the customers check in with a smart phone using the application Amazon Pay\textsuperscript{3}. The store is set up with a large amount of cameras and sensors. Thanks to computer vision and deep learning algorithms, Amazon managed to create a store where technology identifies the products the customers choose. No checkout is required, the chosen products are debited from Amazon Pay account that the customer checked in with.

Digi

StrongPoint\textsuperscript{4} is a company, with its headquarters in Norway, offering technical solutions to the retail business. StrongPoint recently released an identification system called Digi\textsuperscript{5}. Digi consists of a user interface displayed on a touchscreen, a scale, a camera and a label printer. The software is implemented with image recognition in the identification process and can be compared to the existing

\textsuperscript{1}https://www.amazon.com/
\textsuperscript{2}https://www.amazon.com/b?node=16008589011
\textsuperscript{3}https://pay.amazon.com/uk
\textsuperscript{4}https://www.strongpoint.se/
\textsuperscript{5}https://www.strongpoint.se/produkter/sm-5600bs-cam/
counterpart of this project. Digi is new to the market, hence it is not used in many stores. It is an economic issue of the retail business whether the business will change the existing systems or not.

![The Digi-product](image1)

**Figure 2.1: The Digi-product**

**Fruit Classification in Other Areas**

Related work including image recognition has been done in the purpose of controlling the vegetation and harvest of fruitage and other growths at fields of farmers [1][18][23][29]. The technology has been used to automate the yield with the help of robotic harvesting. Several CNN’s has been used to localize the fruits in the purpose of either collecting or counting. However, the issue of creating a fast and reliable fruit detection system persists[10]. This is due to large variation in the appearance of the fruits in field, including colour, shape, size and texture properties.

![Fruits in field](image2)

**Figure 2.2: Image courtesy of Suchet Bargoti and James Underwood from The University of Sydney**
2.2 Usability

Heuristic Evaluation

Nowadays, the identification of products in the retail business is done manually, either by the cashier or the consumer with the help of self-service systems. When the identification is performed manually, the human factor may affect the outcome. There is a chance that the user press the wrong button or could misinterpret the application. An evaluation of the existing system could help to create a GUI better than the existing one. Out of four experiments [17], created by Nielsen and Molich, they concluded that heuristic evaluation is difficult. However, the meaning of an evaluation to test the usability is of importance. Nielsen [16] states: ”My main advice for the teaching of usability engineering would be to base the course firmly in the laboratory.... A required part of any usability engineering course should be to have the students conduct a user test with a small number of real users. Not only is this a good way to teach proper evaluation methodology, but more important, it is the only way to achieve the required revolutionary change in student attitudes (p. xii).”

Heuristic evaluation is an informal method of usability analyses for user interfaces[15][17]. It is simply done by looking at an interface to gather an opinion of what is positive and negative about the interface. Formal collections of guidelines exist when developing interfaces but may come across as intimidating since they are in an order of thousands[14][26]. These guidelines has been reduced to nine guidelines that capture the most crucial errors[14] and can be found in Table 2.1.

Table 2.1: The Nine Guidelines

- Simple and Natural Dialogue
- Speak the User’s Language
- Minimize the User’s Memory Load
- Be consistent
- Provide Feedback
- Provide Clearly Marked Exits
- Provide Shortcuts
- Provide Good Error Messages
- Error Prevention

Experiments found that a single participant following these nine guidelines are rarely able to find more than 50% of usability problems[17]. However, by aggregating multiple problems identified by the participants, the heuristic evaluation method performs quite well. Only as few as three to five participants are in most cases able to find more than 70% of the usability problems[17].

JavaScript and React

The requirements for the graphical user interface are to interpret the conveyed result from the image classification and interact with the user. Furthermore, it has to meet the criteria of being user friendly. Even though creating a GUI in Python is possible and would simplify the communication between the GUI and the image classification, JavaScript with the React library extension is a better alternative due to the slow nature of Python. The system could increase in performance by using a faster language intended to create user interfaces. However, the React library is built to easily create user interfaces and is useful since the project is time limited.

React[^1] is a JavaScript library for creating user interfaces. Some key termi-
technology and concepts of React is the Virtual Document Object Model (DOM) and JSX.

**Virtual DOM**

Since changing objects and its visible data is a slow performance, Virtual DOM takes a lightweight version of the object and changes the data. This results in a faster data change process. Simply, Virtual DOM is used to calculate the minimum set of changes needed to update the application’s actual DOM [7].

**JSX**

JSX is a syntax extension to JavaScript. It is the transform layer which transforms the XML syntax for writing components into the syntax that React uses to render the elements in JavaScript. JSX is easy to read and let us use XML/HTML-like language to put XML/HTML into JavaScript-code.

### 2.3 ImageNet

ImageNet [7] is a large scale image database. In 2009 ImageNet had over 3.2 million of images of over 5000 categories[6] and has only expanded since. ImageNet is a popular database for collections of data sets to train neural networks and has since 2010 held the annual Large Scale Visual Recognition Challenge [8] which has participants from more than 50 institutions[22]. The variety of images makes the database a great source to train a neural network. ImageNet is a common data set to have as a foundation when applying training techniques on a CNN because of variety of categories the database provides[1][23].

### 2.4 Convolutional Neural Networks

Convolutional neural networks (CNN) have over the recent years become great at large-scale image recognition tasks[11]. Large-scale image recognition has been become possible because of large public image databases such as ImageNet. CNN are networks made up of neurons similar to the human brain[12][31]. Figure 2.3 shows an example of a CNN. These neurons consists of weights and biases that form layers and fire in a particular order to end up with a final output. The networks can be trained in order to recognize particular patterns by feeding them large amounts of data. Figure 2.4 displays an example of a CNN searching for features to recognize a motorbike. This is very useful in the field of computer vision since it means that a computer can be trained to recognize different objects. CNN generally consists of three types of layers, not counting the input and output layer. These types of layers are the convolutional layer, the pooling layer and the fully connected layer. Each layer performs a different kind of operation. These operations eventually outputs a value at the end of the network. In the following description of the layers, an image acts like an input and the output is a prediction of what the image is, although this is not always the case, it is related to the project.

**Convolutional Layer**

Taking an image as an input, each convolutional layer performs a different mathematical operation over the pixels. The goal is to extract different kinds of features in the image. These features can be corners, edges or end-points. However, not all pixels in an image are part of the operation. Instead a square,
referred to as kernel or filter, divides the pixels into a subset and performs the mathematical operation on this subset. The kernel is then reapplied all over the image performing the same operation. The notation of how much the kernel is offset from the previous mathematical operation is called stride. Furthermore, multiple mathematical operations can be performed in a single layer in purpose to extract more features. As the network progress through the layers the features extracted are then combined by the higher layers to form feature maps. Feature maps are a new image where every pixel is the result of the mathematical operation performed by the kernel. Figure 2.5 displays an example of a convolutional layer.

**Pooling Layer**

The pooling layer\([5][13]\) combines the output of multiple neurons to one single neuron. The function deciding what the value of the new combined neuron is called pooling function. Popular pooling functions are called max, average and stochastic. The purpose of this layer is to scale down the resolution of the feature maps and thus reducing the sensitivity of the output to shifts and distortions. Figure 2.6 displays an example of a max pooling layer.

**Fully Connected Layer**

The fully connected layer is located at the end of the network. It takes the output of the previous layer and produces an N-dimensional vector where N is
the number of classes the network tries to identify. The vector contains values of how high the probability of each class is. This is again done by performing a mathematical operation on all connected neurons. A visualization of a fully connected layer can be found in Figure 2.7.

Figure 2.5: Example of a convolutional layer performing on an input image with a kernel size of $2 \times 2$ and a stride of 1.

Figure 2.6: Example of a max pooling layer with a kernel size of $2 \times 2$ and a stride of 2.

2.5 Transfer Learning

Transfer learning is the method of storing knowledge gained from a particular problem and re-applying it to a new problem[19]. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. In many real-world applications, it is expensive or impossible to recollect the needed training data and rebuild the models[19]. Transfer learning is thus a way to create new models with very little data compared to the initial training.
Loss Function

Loss function is a mathematical function used when training networks. It measures how far the estimated value is from the true value. This also allows the network to change its weights and biases to more accurately create an estimate next iteration by minimizing the loss function. The loss function is minimized by computing the gradient descent (GD).

\[ \omega_{t+1} = \omega_t - \gamma \nabla Q(\omega_t) \] (2.1)

The gradient \( \nabla Q(\omega_t) \) is a vector pointing in the direction of steepest ascent. Gradient descent learning attempts to find a point \( \omega \) in some parameter space that minimizes a loss function \( Q(\omega) \)[28]. It is computed by taking the partial derivatives for each dimension in the loss function. The following is an example of computing the gradient from a function \( f \) with three dimensions:

\[ \nabla f = \text{grad} f = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} + \frac{\partial f}{\partial z} \] (2.2)

The chosen gain \( \gamma \), referring to equation 2.1, is a measurement of how large steps the network should take in the direction of the gradient. By subtracting the \( \gamma \nabla Q(\omega_t) \) from the current iteration \( t \) from the parameters \( \omega \), a descending step in the direction of the gradient is made. This process is repeated for each iteration during the training procedure. However, a more practical real world example is to compute the gradient from a batch of samples. This is called batch gradient descent (BGD).

\[ \omega_{t+1} = \omega_t - \frac{\gamma}{n} \sum_{i=1}^{n} \nabla Q(\omega_t), \] (2.3)

Although it is discussed to be less efficient than regular gradient descent[28], it reduces the overall number of computations needed.

Stochastic gradient descent (SGD) is an optimization technique applied to the learning process. Instead of computing the gradient exactly, the gradient is estimated each iteration from a single sample \( z_t \)[3].

\[ \omega_{t+1} = \omega_t - \gamma \nabla Q(z_t, \omega_t) \] (2.4)

The single sample is randomly chosen in hope that the behaviour will resemble equation 2.1. The convergence of stochastic gradient descent has been studied extensively in the stochastic approximation literature[2] and is with correct means almost always sure to convergence under mild conditions[4].

Batch Normalization

Imagine two inputs of different scale. The first input is a kilometer count of how long a specific car has driven. For the sake of the argument, lets limit the values to a range from 0 to 100000 km. The second input node is an age of a person, typically in the range of 0 to 100. Since the distribution of these two inputs differ in such large amounts, the network will have problems adapting its trainable parameters[9] during the training process. The network experiences something known as covariate shift[24]. Furthermore, the large variations will cascade through the network leading to imbalance in the gradient. This is known as the exploding gradient problem[20]. To minimize the covariate shift and solving the exploding gradient problem, batch normalization (BN) is introduced. The purpose of BN is to minimize the covariate shift and thus removing

---

9Also known as step size or learning rate
the need of the network to adapt to a new distribution\cite{9}. BN takes the batch of samples and calculating the mean as shown in equation \ref{2.5}.

\[ \mu_\beta = \frac{1}{m} \sum_{i=1}^{n} x_i \]  

(2.5)

Where \( \mu_\beta \) is the batch mean and \( x_i \) is an sample from the batch \( \beta \) of size \( m \). From the batch mean, the batch variance \( \sigma^2_\beta \) is then determined as in equation 2.6.

\[ \sigma^2_\beta = \frac{1}{m} \sum_{i=1}^{n} (x_i - \mu_\beta)^2 \]  

(2.6)

The normalization of the input is then done as in 2.7. Here, parameter \( \epsilon \) is introduced which is a trainable value that the network will adjust over the training period.

\[ \hat{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma^2_\beta + \epsilon}} \]  

(2.7)

\section{2.6 Architectures}

The number of available architectures to train goes beyond count. Comparing all architectures to each other is a difficult task. Instead, the following three architectures has properties making them worth while evaluating for this project.

\subsection{Inception v3}

Inception v3 is an open source architecture created by Google\footnote{https://www.google.com} and trained on 1.2 million images from thousands of different categories. It is a module of GoogleLeNet designed to function under strict constraints on memory and on a computational budget \cite{27}. In the ImageNet Challenge 2014, GoogleLeNet with the Inception v3 module, had the least error rate comparing to other architectures \cite{22}. With an average error rate of 6.66\%, the network defeated all the other competitors. The Inception v3 module is 42 layers deep and uses BN in both the convolution layers and in the fully connected layers. At the start of Inception v3, is a sequence of three convolutional layers which takes an input image of \( 299 \times 299 \times 3 \). The most unique part of the Inception network constitutes of the Inception modules.

Table 2.2: Outline of the Inception v3 architecture. \cite{12}

<table>
<thead>
<tr>
<th>type</th>
<th>patch size/stride</th>
<th>input size</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv</td>
<td>3x3/2</td>
<td>299x299x3</td>
</tr>
<tr>
<td>conv</td>
<td>3x3/1</td>
<td>149x149x32</td>
</tr>
<tr>
<td>conv padded</td>
<td>3x3/1</td>
<td>147x147x32</td>
</tr>
<tr>
<td>pool</td>
<td>3x3/2</td>
<td>147x147x64</td>
</tr>
<tr>
<td>conv</td>
<td>3x3/1</td>
<td>73x73x64</td>
</tr>
<tr>
<td>conv</td>
<td>3x3/2</td>
<td>71x71x80</td>
</tr>
<tr>
<td>conv</td>
<td>3x3/1</td>
<td>35x35x192</td>
</tr>
<tr>
<td>3xInception</td>
<td>As in Traditional</td>
<td>35x35x288</td>
</tr>
<tr>
<td>5xInception</td>
<td>As in Factorized</td>
<td>17x17x768</td>
</tr>
<tr>
<td>2xInception</td>
<td>As in Asymmetric</td>
<td>8x8x1280</td>
</tr>
<tr>
<td>pool</td>
<td>8x8</td>
<td>8x8x2048</td>
</tr>
<tr>
<td>linear</td>
<td>logits</td>
<td>1x1x2048</td>
</tr>
<tr>
<td>softmax</td>
<td>classifier</td>
<td>1x1x1000</td>
</tr>
</tbody>
</table>

\footnote{https://www.google.com}
Three types of Inception modules are used. These three modules can be seen in Figure 2.8. First, three traditional Inception modules with an input of $35 \times 35 \times 288$ reduces the image to $17 \times 17 \times 768$. Second, five factorized Inception modules reduces the output from the traditional modules to the size of to $8 \times 8 \times 1280$. Third, two Inception modules reduces the output from factorized modules to $8 \times 8 \times 2048$. The three types of Inception modules are followed by a pooling layer which reduces the image to $1 \times 1 \times 2048$. Finally, Inception v3 network has a fully connected layer with a depth of 1000, one for each class in the ImageNet data set.

**MobileNet**

In many real time mobile applications implemented with recognition tasks to identify certain objects or surroundings, light weight architectures are preferable to match the resource restrictions on the platforms. MobileNet is an architecture developed to function on mobile and embedded vision applications [8]. MobileNet is used in Inception models and is built on depthwise separated convolutions to reduce the computation and model size. The depthwise separated convolutions splits the standard convolution method of combining and filtering in one sequence, into different layers. One layer for combining and one layer for filtering. This method reduces the computation size drastically. The architectures input layer takes an image of size $224 \times 224 \times 3$. Following the input layer is a stack of convolutional layers, one average-pooling layer and a fully connected layer. The kernel sizes of the convolutional layers vary between $3 \times 3$ and $1 \times 1$. The depthwise separated convolutions structure is clearly shown in the network. Almost 75% of the total parameters in the network are located in convolutional layers using a kernel of $1 \times 1$. This is what reduces the computation size. Table 2.3 shows the MobileNet’s architecture body. All layers all followed by a BN and ReLU with the exception to the final fully connected layer. Depthwise, MobileNet has 28 layers.

**VGG**

Very deep convolutional neural networks was first mentioned by Karen Simonyan and Andrew Zisserman[25]. The paper evaluates different models of a network architecture. Andrew Zisserman and Karen Simonyan found that model D, shown in 2.4 performed with the lowest error rate. The finding were later used to secure the first and second place in ImageNet Challenge 2014 in the localization and classification tracks respectively. The model was trained.
Table 2.3: Outline of the MobileNet body architecture. (from Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017)

<table>
<thead>
<tr>
<th>Type / Stride</th>
<th>Filter Shape</th>
<th>Input Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv / s2</td>
<td>$3 \times 3 \times 3$</td>
<td>$224 \times 224 \times 3$</td>
</tr>
<tr>
<td>Conv dw / s1</td>
<td>$3 \times 3 \times 32$</td>
<td>$112 \times 112 \times 32$</td>
</tr>
<tr>
<td>Conv / s1</td>
<td>$1 \times 1 \times 32 \times 64$</td>
<td>$112 \times 112 \times 64$</td>
</tr>
<tr>
<td>Conv dw / s2</td>
<td>$3 \times 3 \times 64$</td>
<td>$112 \times 112 \times 64$</td>
</tr>
<tr>
<td>Conv / s1</td>
<td>$1 \times 1 \times 64 \times 128$</td>
<td>$56 \times 56 \times 128$</td>
</tr>
<tr>
<td>Conv dw / s1</td>
<td>$3 \times 3 \times 128 \times 64$</td>
<td>$56 \times 56 \times 128$</td>
</tr>
<tr>
<td>Conv / s1</td>
<td>$1 \times 1 \times 128 \times 128$</td>
<td>$56 \times 56 \times 128$</td>
</tr>
<tr>
<td>Conv dw / s2</td>
<td>$3 \times 3 \times 128 \times 64$</td>
<td>$56 \times 56 \times 128$</td>
</tr>
<tr>
<td>Conv / s1</td>
<td>$1 \times 1 \times 128 \times 256$</td>
<td>$28 \times 28 \times 256$</td>
</tr>
<tr>
<td>Conv dw / s1</td>
<td>$3 \times 3 \times 256 \times 64$</td>
<td>$28 \times 28 \times 256$</td>
</tr>
<tr>
<td>Conv / s1</td>
<td>$1 \times 1 \times 256 \times 256$</td>
<td>$28 \times 28 \times 256$</td>
</tr>
<tr>
<td>Conv dw / s2</td>
<td>$3 \times 3 \times 256 \times 64$</td>
<td>$28 \times 28 \times 256$</td>
</tr>
<tr>
<td>Conv / s1</td>
<td>$1 \times 1 \times 256 \times 512$</td>
<td>$14 \times 14 \times 512$</td>
</tr>
<tr>
<td>Conv dw / s1</td>
<td>$3 \times 3 \times 512 \times 64$</td>
<td>$14 \times 14 \times 512$</td>
</tr>
<tr>
<td>Conv dw / s2</td>
<td>$3 \times 3 \times 512 \times 64$</td>
<td>$14 \times 14 \times 512$</td>
</tr>
<tr>
<td>Conv / s1</td>
<td>$1 \times 1 \times 512 \times 1024$</td>
<td>$7 \times 7 \times 1024$</td>
</tr>
<tr>
<td>Conv dw / s2</td>
<td>$3 \times 3 \times 1024 \times 64$</td>
<td>$7 \times 7 \times 1024$</td>
</tr>
<tr>
<td>Avg Pool / s1</td>
<td>Pool $7 \times 7$</td>
<td>$7 \times 7 \times 1024$</td>
</tr>
<tr>
<td>FC / s1</td>
<td>$1024 \times 1000$</td>
<td>$1 \times 1 \times 1024$</td>
</tr>
<tr>
<td>Softmax / s1</td>
<td>Classifier</td>
<td>$1 \times 1 \times 1000$</td>
</tr>
</tbody>
</table>

using BGD and achieved a 7.32% error rate for image classification on the ImageNet data set[22]. VGG has also been used in similar projects where the task was to identify fruits[23]. The released smaller version of the network contains 16 weight layers and over 130 million of parameters to tune[25]. Images with a fixed size of $224 \times 224$ are used as input to the network. The only preprocessing done to the input data is subtracting the mean RGB value of the data set from the pixels. Following the input layer is a stack of convolutional layers using a kernel size of $3 \times 3$, five max-pooling layers and three fully connected layers. The first two fully connected layers use 4096 neurons and the last one use 1000 neurons, one for each class in the ImageNet data set.
Table 2.4: Outline of VGG models evaluated in the VGG paper. Version D is the version used in this report. (from Very Deep Convolutional Networks For Large-Scale Image Recognition, 2015)

<table>
<thead>
<tr>
<th>ConvNet Configuration</th>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 weight layers</td>
<td></td>
<td></td>
<td>13 weight layers</td>
<td></td>
<td>16 weight layers</td>
<td></td>
</tr>
<tr>
<td>input (224 × 224 RGB image)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-64</td>
<td>conv3-64</td>
<td>LRN</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td></td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>soft-max</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 3

Methodology

The project is divided into two phases, the experimentation phase and the implementation phase. The experimentation phase aims to find the most suitable network for this project. It lays the foundation of how the system will perform in the end. The implementation phase aims to describe how the software and hardware are integrated to form an identification system.

3.1 Tools

Hardware

Raspberry Pi

To replicate a real life scenario where a self-serving system has limited processing power, a Raspberry Pi has been selected. A Raspberry Pi operates in a way similar to a regular computer but on a fraction of the cost and size. It is a great development platform for creating prototypes and trying concepts.

![Figure 3.1: A Raspberry Pi model 3 B.](image_url)

It is based on a 64bit Quad Core 1.2GHz CPU and has 1GB of RAM available. Additionally, the Raspberry Pi has a CSI and a DSI port for connecting a Raspberry Pi camera module and a touchscreen display. Additionally, the Raspberry Pi also has 40 general purpose pins to connect various hardware. The processor has a wireless LAN that enables an internet connection. Furthermore, CNN’s and various deep learning frameworks has been benchmarked to use on the mini computer[21].

The system could be run on a regular computer. However, to get an individually functioning system, not depending on external sources, a Raspberry Pi was chosen for this project. Furthermore, to run the system on the mini computer would lower the costs and occupy minimum amount of space if placed in a physical store.

Camera

The camera used for this project is the Raspberry Pi Camera Module v2 and is the official product from the Raspberry Pi Foundation. The camera has a
8-megapixels resolution and is compatible with the Raspberry Pi without any drivers which enables a quick setup. The camera is connected via a ribbon cable to the DSI port on the Raspberry Pi.

Display
The touchscreen display used for this project is called Raspberry Pi Display 7" multitouch. The display is a 800 by 480 pixels display and connects to the processor via an adapter board. The adapter board handles power and signal conversion to the Raspberry Pi. The display is connected to the GPIO port to get power from the processor and a second connection is required to the DSI port to visualize the data.

Software
Python
Python is a loosely typed language and emphasizes code readability. The language has access to enormous open source libraries created by the Python community and companies. The greatest reason why Python was chosen was the code readability, as mentioned above. Using a language that emphasize this property enables developers to ease into a subject without having to struggle with syntax. Additionally, Python was chosen because of access to complex mathematical libraries. Matrices and vectors are widely used in image classification. NumPy\(^1\) is a Python library that provides an easy way of creating large N-dimensional arrays and matrices along with mathematical functions to operate on these. Furthermore, Python is a commonly used programming language to use within machine learning softwares such as Tensorflow, Caffe\(^2\) and PyTorch\(^3\).

Tensorflow
Tensorflow\(^4\) is an open source software for numerical computation. It was originally created to conduct machine learning and deep neural networks research. Tensorflow provides neural network architectures and scripts to retrain the networks for users who wants to apply them in different contexts.

Node.js
Node.js is an open source platform to develop server-side and networking applications. It is written in JavaScript and provides libraries of various JavaScript modules which simplifies the development of web applications. Node.js has the ability to handle data streaming, I/O bound and API based applications. Having Node.js simplifies the issue when different applications needs to communicate with each other.

Postman
Postman is a tool created to ease for front end developers by simulating the response from a server. Postman is working as a mock-up server and allows users to view responses without creating the back end. While developing the back end, the front end was developed in parallel thanks to Postman. Furthermore, to speed up the developing process not spending time waiting for the image classifier to send a request, Postman handled the fake HTTP requests, and debugging in a smart way was enabled.

---

\(^1\)http://www.numpy.org/
\(^2\)http://caffe.berkeleyvision.org/
\(^3\)http://pytorch.org/
\(^4\)https://www.tensorflow.org/
Training

Using transfer learning, networks pre-trained on the ImageNet data set are re-trained as described in their respective papers[8][25][27]. The papers describe using SGD (Inception) or BGD (MobileNet and VGG) for minimizing the loss function as well as the preprocessing performed to the images. Network architectures and open source scripts for retraining are provided by Tensorflow. The training was executed on a NVIDIA GeForce 1060 graphics card, privately owned by one of the participants in the group, to minimize the retraining time.

Data Set

The data set is divided into 10 different classes. Each class represents one type of fruit and vegetable. The chosen classes are apple, avocado, banana, bell pepper, clementine, kiwi, orange, pear, potato and tomato. These classes are chosen because some fruits and vegetables have similar appearances and are frequently bought in retail markets. Limitations to the data set has been done in order to not make the project too extensive. These limitations are that all types of a fruit or vegetables reside under the same class. This means all types of apples reside under the apple class and similar for each fruit. Images for the data set are collected from ImageNet. Each class consists of approximately 400 images. This subset of images create a base data set to train the networks. Example images from each class of the data set can be found in the appendix.

Beyond using a single data set fetched from ImageNet, pictures taken in an environment similar to a retail store has been added to the subset to create a second data set. The additional images consists of all 10 classes and 30 images for each class. The developers of Tensorflow recommends to retrain networks with images similar to the working environment5. The additional images represents roughly 7-8% of the total data set. For retraining the networks, the second data set was used. For this project, images of fruit and vegetables has been taken without being placed in plastic bags.

---

5[https://www.tensorflow.org/tutorials/image_retraining](https://www.tensorflow.org/tutorials/image_retraining)
3.2 Implementation

This section will go deeper into the steps the system takes to display a result. The image capture and classification will be referred to the back end, they build the backbone of the system. The graphical user interface is referred to as the front end, it is what the user will interact with. The work flow is described from back end to front end and starts with the activation mechanism and ends at the graphical user interface.

Workflow

The system is started by launching the individual modules separately. Each module works independently from the other and will keep on running regardless if any of the other modules are unavailable. The back end is launched by starting a python script from the command line tool using the 'python' command. The Node.js middleware and React front end are launched in a similar way using the 'node' and 'npm' command respectively. Once all modules in the system are started, the back end waits for an activation mechanism to start. As the back end receives the starting signal, from the activation mechanism, an image is captured from the connected camera. This image is passed to the convolutional neural network for classification. The result from the classification is sent to the middleware using an HTTP post request. Once the middleware receives the post request, it redirects the data to the front end. The front end is constantly waiting for a response from the server. When the front end receives the data, an automatically push to the Result-page where the data from the image classification is presented.

![Figure 3.4: The basic work flow of the system.](image)

Activation Mechanism

The activation mechanism to the system is a 5 kg double bending beam load cell. The load cell is connected to a HX711 amplifier which in turn is connected to the pins of the Raspberry Pi. From the amplifier, the ports DT and SCK are connected to GPIO 5 and GPIO 6 on the Raspberry Pi respectively. The system is activated by placing an object on the load cell. If the measured weight of the object exceeds 200 grams, the system will proceed to capture an image.

Image Capture

The camera is placed on top of the system and tilted forward towards a flat surface in front of the system. The purpose of this is to minimize the interference from other objects in the surroundings that otherwise could impact the results.
of the classification. The interference are objects, patterns and colours not relevant in the task of classifying the main object. In other words, everything not part of the vegetable or the fruit. To further decrease the interference in the image a resolution of 640 by 480 was chosen. This resolution gives a user some leeway for where the object is placed in the image as well as cutting away interference in the surroundings. Once the camera is active, it waits for the activation mechanism to trigger. When the system is triggered, an image from the camera is captured and saved to a numPy array with a size equal to the resolution. The numPy array is then passed to the classifier.

Classification

When the classifier is initializing, it loads the pretrained model specified as an argument. This is by far the most computational heavy operation and is only done once every initialization. Furthermore, the classifier also defines image preprocessing operations in the initialization phase. This will prevent an issue where these operations are recreated for every image and thus creating a memory leak. Once the initialization is complete, the classifier is ready to receive images one at a time. The image is provided to the classifier as a numPy array in a function argument. Before any processing of the image takes place, an HTTP post request is sent to Node.js. The purpose of the request is to inform the front end that a classification process is about to begin. After the request is issued, the image is transformed into a Tensorflow object called tensor. The tensor applies the image preprocessing operations before propagating it through the network. The image preprocessing involves subtracting the image mean to every pixel and resizing the image to fit the input layer of the network. After the image is propagated through the network, an array is extracted from the output layer. This array contains the probabilities of the object in the picture for each of the classes the network is trained to recognize. The top five values are extracted from the array and staged to be transferred to the user interface.

HTTP post

During the staging process, the array containing the results from the classifier is encoded into a JSON object. The JSON object is then added to the HTTP post request and sent to Node.js. Once the respond is sent, the returning status code is evaluated. Should anything else other than a status code 200 return, a second post request is sent. If the second post request does not receive a status code 200, the system assumes something is wrong on the other end and returns to the waiting state to capture an image.
Node.js

Node.js serves as a middleware between the front and back end. Using the socketIO\textsuperscript{6} library, the Node server sends data to the front end via a socket channel established on launch. The task of the middleware is to receive HTTP calls and respond to them accordingly. When an HTTP post request is received, the server simply redirects the data via the socket to the front end. As a security mechanism, only HTTP requests which originate from the local host is accepted. This ensures that external sources can not interfere with the front end from the middleware.

Graphical User Interface

The graphical user interface is coded to work as a browser application. It includes several components to make it easy for any developer to understand. The GUI is written in the text editor Atom including formatters and packages to ease the coding and auto indent the code, specifically to use with React. The code is kept simple and clear and follows programming “rules of thumb”. Names are adjusted and chosen for the purpose of the component. The maximum length of any component is 200 rows to keep the code airy and avoid mistakes leading to bugs or possible flaws. The user interface is written in JavaScript with the React library. It is styled with classes created with Cascading Style Sheets (CSS) combined with inline styling.

The content of the application is rendered every time the code is saved which enables updates in real time. Possible errors and warnings are printed in the terminal of the computer where the application runs. More specific errors and warnings are printed in the console of the web browser and displays every time the content is saved. This property provides effective development and since almost every change is saved and updated, errors and mistakes are easily noticed.

3.3 Network Evaluation

This section describes the method and setup for how the networks are evaluated and tested. The networks are evaluated by two properties. Propagation time, which is the time it takes for an image to be classified, and accuracy, which is how accurate the prediction is.

Accuracy

To properly evaluate the accuracy of the networks, a setting close to the working environment of the final product had to be created.

Since the camera was already tilted forward, as mentioned in section 3.2, a flat and relatively equally colored surface had to be established. Once the

\textsuperscript{6}https://socket.io
environment was set, images of fruits and vegetables captured by the mounted camera was propagated through the network. The results of each classification was logged to a text file and later compiled. A total of 10 tests was conducted on each network. Each test yielded 10 samples of accuracy on fruits or vegetables from different angles and in various amounts. This resulted in 100 samples of accuracy for each network.

Propagation Time
To evaluate the propagation time of the networks, series of 100 images was captured and classified. The propagation time is the time between start and end of the classification. Each test of a network was run five times and yielded 500 samples of time propagation per network. The content of each image was not of relevance and thus not saved since it did not affect the outcome the measured time. Additionally, only one network was loaded per test to prevent filling the working memory and affecting outcome of the propagation time. Each sample was written to a log file and later compiled.

3.4 System Testing

Usability Evaluation
Three heuristic evaluations has been performed, one on the existing systems in the retail store, and two on the produced system:

Heuristic Evaluation 1: Test performed on existing systems in the retail market. The system uses manual identification of the fruit.

Heuristic Evaluation 2: Test performed on progressing system to discover flaws during the development

Heuristic Evaluation 3: Test performed on finished system.

The tests has been performed on five test persons since the method only requires 3-5 persons to find 70% of the usability flaws. The heuristic evaluation has following steps to be completed:

1. Recording of the procedure starts
2. The individual is asked to identify a fruit
3. The recorded material is analyzed

The usability tests are performed without any information given to the user. The user is simply requested to identify a fruit or vegetable with the goal of printing a label. No information is given which could lead to any clues or guidance. The result has been analyzed by following the interactions between the individual and the computer to find any possible flaws.

Furthermore, another kind of usability test of the system has been performed with the help of fifteen individuals. The individuals are of different age, gender and background to get as many insights as possible. The test was created by the project members and is called the "Prelaunch Test". The test was performed when the project members felt the graphical user interface was finished. Since the project members is the creators of the system, the procedure of identifying a product is clearly precieved, hence there is no experienced usability flaws according to the creators. To find possible flaws not detected by the creators and to get inputs from the users, the Prelaunch Test was performed.

The Prelaunch Test included following questions and tasks:
1. First impression evaluation by the user

2. The user is requested to find the specific fruit or vegetable he or she has chosen. No description is given. The user can either search for the product or put it in front of the camera.

3. Evaluation of the system consisting of free comments and two questions:
   - Do You have any comments?
   - Which system do You prefer if You have to choose between this one or the existing systems used in the retail business?
   - How can the system be improved?
Chapter 4

Results

4.1 Network Architectures

The networks are evaluated in terms of accuracy and propagation time. Data for evaluating these networks was collected by performing the two types of tests described in section 3.3. The results presented in this section are from the final versions of MobileNet and Inception after three iterations of retraining. Only the relevant diagrams will be shown in this section. A full list of all diagrams can be found in appendix. VGG was not successfully trained, hence no results of the network.

Propagation Time

Between the two networks there is a large difference in propagation time. The average propagation times for 500 images are 3.3 seconds for Inception and 0.43 seconds for MobileNet as shown in Table 4.1. This makes the average propagation time of Inception 7.67 times slower than MobileNet. In this section, two types of diagrams are shown. The first type of diagram displays the propagation times from five different tests. The y-axis of these diagrams represents the propagation time in seconds. The x-axis represents an image in the sequence of 1 - 100, where 1 is the first image taken in the test and 100 the last. The second type of diagram displays a regression analysis of the average propagation time for each image in the sequence. Each sample is calculated from five images with the same sequence number. The x- and y-axis represents propagation time and image number respectively, just as in the first diagram.

Table 4.1: Average propagation time 500 images.

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>3.30</td>
</tr>
<tr>
<td>Mobilenet</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Inception

Figure 4.1 displays 500 propagation samples taken in five tests using Inception. The first image taken in all tests has significantly longer propagation time than the rest of the images. Test 2 to 5 have a fluctuating pattern in the first 50 images where test 1 has the same pattern in the later 50 images. However, this fluctuating pattern later pans out and becomes insignificant. The regression analysis of the average propagation time per image displays a convergence to a
value close to 3.10 seconds, as shown in Figure 4.2. This value falls just below the total average of 3.3 seconds.

![Figure 4.1: Propagation times for Inception.](image)

Figure 4.1: Propagation times for Inception.

![Figure 4.2: Regression analysis of Inception’s average propagation time per image.](image)

Figure 4.2: Regression analysis of Inception’s average propagation time per image.

**MobileNet**

MobileNet’s propagation samples are displayed in Figure 4.3. All five tests display similar behaviour and does not differ from each other. Unlike Inception, MobileNet’s propagation time is stable after the first image. A regression analysis (Figure 4.4) confirms that the propagation time is stable at about 0.5 seconds which falls just above the average propagation time of 0.43 seconds.

![Figure 4.3: Propagation times for MobileNet.](image)

Figure 4.3: Propagation times for MobileNet.
Figure 4.4: Regression analyses of MobileNet’s average propagation time per image.

Accuracy

The confusion matrices presented below displays what the classifier commonly misinterprets a fruit or vegetable for. Four types of color markings are used. Green, which are classes where the network performs with accuracy higher than 90%. Blue, classes that are predicted to be correct more than half of the time but not viable enough to get a green marking. Red, classes the network fails to predict. Finally, yellow is used to mark classes that the network misinterprets a type of fruit for. A networks top 1 accuracy can be calculated by dividing the sum of the diagonal by the number of tests performed. This confusion matrix displays the top 1 accuracy for the network. Additionally, a confusion matrix for the top 3 predictions are also displayed to show good the network is at placing the correct label among the top three ranks. To further investigate the accuracy of the networks for some classes, a cumulative matrix curve, CMC, to the associated class will be presented. The CMC displays how good the network is at placing the correct fruit in the top 1 to 5 rank respectively. The x-axis show a rank from 1 to 5 and the y-axis shows the accuracy for that rank.

Inception

Table 4.2 displays the confusion matrix for Inception. Highlighted in green are apple, avocado, banana, pear, potato and tomato. Marked in blue is orange which is often misinterpreted as a clementine. Finally, marked in red are clementine and kiwi. The top 1 accuracy of Inception in 76%. As displayed in the Figure, Inception has a problem distinguishing an orange from an clementine, but an actual clementine is often misinterpreted as a tomato. Figure 4.5 displays the CMC for the orange class. It shows that even though orange was misinterpreted as a clementine, the correct label was always among the top 2 ranks during the test.

Table 4.2: Inception top 1 confusion matrix.
The CMC for the clementine class is shown in Figure 4.6. The diagram emphasizes that Inception is bad at classifying clementines at a top 1 rank. However, a 100% percent is reached when displaying top 3 rankings of each prediction.

Kiwi is by far the most difficult class for Inception to label. Out of 10 images, the network never managed to place the kiwi label among the top 1 ranks. Not even at a top 5 ranking is the network able to have a 100% accuracy for kiwis.

The confusion matrix for Inceptions top 3 accuracy is displayed in Table 4.3. Comparing the top 1 accuracy confusion matrix to the top 3 accuracy confusion matrix it can be said that Inceptions performs well. The overall accuracy rises from 76% to 96%. The only class difficult to label is the kiwi class. However, the accuracy of the kiwi class has risen from 0% to 60%.

**MobileNet**

The performance of MobileNet, in terms of accuracy, is quite similar to the performance of Inception. A confusion matrix for MobileNet can be seen in Figure 4.4. The top 1 accuracy for MobileNet is the same as for Inception,
76%. Marked in green are avocado, bell pepper, orange, pear, potato and tomato. Marked in blue are apple and banana. These two classes still perform quite well, even though marked in blue. Apple is misinterpreted for a pear two times and banana is misinterpreted three times, one of the times as an apple and twice for a pear. The classes MobileNet has difficulties interpreting are clementine and kiwi. When comparing to Inception, MobileNet is actually better at classifying oranges and pears. However, MobileNet performs worse or equally good in all other categories.

A look at the CMC (Figure 4.8) for clementine shows that MobileNet reaches a 100% accuracy if the top 5 labels are displayed. The network misinterpreted the clementine a majority of the times for a tomato.

Figure 4.8: MobileNet error rates for clementine

Just as Inception has difficulties labeling kiwi, MobileNet follows in the same lines but for the worse. Out of 10 images, MobileNet succeeds in a 60% accuracy when taking the top 5 ranks into consideration.

The confusion matrix for the top 3 predictions can be seen in table 4.5. MobileNet still has difficulties with the same classes as for the top 1 predic-

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Actual Class</th>
<th>Apple</th>
<th>Avocado</th>
<th>Banana</th>
<th>Bell Pepper</th>
<th>Clementine</th>
<th>Kiwi</th>
<th>Orange</th>
<th>Pear</th>
<th>Potato</th>
<th>Tomato</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Avocado</td>
<td>3</td>
<td>10</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Banana</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bell Pepper</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Clementine</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Kiwi</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Orange</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pear</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Potato</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tomato</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 4.9: MobileNet error rates for kiwi

These classes are clementine and kiwi. However, the overall accuracy has increased. From a top 1 accuracy of 76% to 97%.

Table 4.5: MobileNet top 3 confusion matrix.

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Apple</th>
<th>Avocado</th>
<th>Banana</th>
<th>Bell Pepper</th>
<th>Clementine</th>
<th>Kiel</th>
<th>Orange</th>
<th>Pear</th>
<th>Potato</th>
<th>Tomato</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>10</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Avocado</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Banana</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bell Pepper</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Clementine</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Kiel</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Orange</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Pear</td>
<td>10</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Potato</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>3</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Tomato</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
4.2 System Design

Hardware

The hardware consists of a Raspberry Pi, display and case, camera and camera case. The display case contains a storage for the Raspberry Pi which makes it easy to conceal the processor, which in turn makes the design look clean. Following images shows how the appearance of the system with the graphical user interface displayed.

![The front of the system displaying GUI](image1.jpg)

Figure 4.10: The front of the system displaying GUI

![The back of the system](image2.jpg)

Figure 4.11: The back of the system

Interface

The design of the GUI is created with the goal of appearing simple and conspicuous. It is supposed to lead the user in the right direction with few interaction calls. To avoid confusion, the user has limited options to take. At the start page (see Figure 4.12), there are three possible actions to take which eases a quick decision. The user can either put the product on the activation mechanism which triggers the classifier, search for the product or chose the product directly if it is displayed at the default page. The products displayed at the default page is the most frequently bought products and are chosen manually.
Code Structure

The graphical user interface constitutes of compact components to make the code easy to develop and maintain. Figure 4.13 visualizes the flow of the graphical user interface.

The base of the application constitutes of a component called App. The App-page contains the components and routes to the other pages displayed in 4.13. Routes are created with the help of a library called React Router Dom \(^1\) and are essential to navigate between pages. In the code of the App-component, routes to the Identification-page and the Home-page are created. The Route-objects includes a component and a path. To create a Route, JSX-syntax is used:

```jsx
<Route path = "/identification" component = {Identification} />
```

Furthermore, the App-component includes code to be able to receive data from the Node.js server. When the activation mechanism is triggered, a post providing information that the classifier is labeling an image, is sent to the App-component. When sending a note from the server, a function called `componentDidMount` is immediately invoked. The function pushes to the Identification-page which in turn waits for a second response from Node.js. Thereafter,

\(^1\)https://www.npmjs.com/package/react-router-dom
the Identification-page renders the IdentificationResults-component and the results, sent from the classifier via Node.js, is rendered on the screen. In Figure 4.14 the flow for when identifying an object with the help of the classifiers is shown.

![Flowchart displaying the identification process from back end to front end](image)

The Home-page consists of the SearchField-component. The Home-page is rendered inside of the App-component as long as there is no response or data sent from the server. The SearchField-component is mainly built on states and includes the logic of the search function. To search for a fruit or a vegetable, there is a variable constantly being checked for changes, called a state. If the input from the user is recognized in any name of any product stored, the products which includes the combination of the written letters is shown on the screen. If the input text is "BA" the state will update to "BA" and the program will render products including "BA", such as banana (see Figure 4.15. In conclusion, the user must not search for a component with a button click, the results will be rendered constantly after every input change.

![SearchField-component with text input "BA"](image)

The IdentificationResults-component includes the identified products sent from the image classifier and renders if they have an accuracy of 5% or more. At this page, the user is requested to click on the desired product. If the classifier fails and the actual product is not displayed, the user has the option to search. The user will then be sent back to the Home-page.

An essential building block is called ProductItem. ProductItem includes name, price, label and a picture. It is also including an onClick-function. The onClick-function will tell the user that a label is being printed when clicked. This function would print the label in the real-case scenario. However, since this project has delimitations, it only functions to communicate with the user. When creating a ProductItem, JSX-syntax is used:
"name", "src", "picture", "price" and "label" are called properties of the ProductItem-component and can be fetched at other components by calling "this.props.name" where "this" refers to actual component.

Other pages are called Printing and Identification. These components are telling the user that some process is running and helps the user to understand that some interaction was performed. The Printing-component is rendered when the user clicks on a product. The Identification-component is rendered when the classifier is performing its labeling. Both components displays a Loading-component:

![Loading-component](banana.jpg)

Figure 4.16: Identifying object

The Loading-component is fetched from a React library. Instead of creating an own component, the Loading-component is used and enables time-efficiency.

### 4.3 Usability

**Prelaunch Test**

The Prelaunch Test concluded that there was some usability flaws in the graphical user interface. The most frequently usability flaw occurred when the user was supposed to print the label. When the system finished to identify the fruit or vegetable and the result was presented on the display, some of the users did not click the product to print the label. The users assumed that the label was going to print automatically when the correct fruit or vegetable was identified. However, the users who got two or more products as a result, did click the product.

According to the Prelaunch Test, the discovered usability flaws was following:

- No clear printing request
- No clear feedback
The test users provided some valuable input such as:

- Add an instruction button
- The header-component looks like its clickable because of rounded corners

**Heuristic Evaluation**

Heuristic evaluations has been performed on both systems. The evaluations concludes which of the nine guidelines the graphical user interface breaks.

**Heuristic Evaluation 1**

According to the test persons, the existing systems lacks some of the guidelines to be user friendly. In total, five of the ground rules mentioned in section 2.2 were discovered broken:

- Simple and Natural Dialogue
- Speak the User’s language
- Be consistent
- Minimize the User’s memory load
- Provide shortcuts

The first guideline discovered broken, was Simple and Natural Dialogue. According to the results of the test users, the ground rule was broken several times. Comments from users and observations of the behaviour concludes following problems identified (see Table 4.6).

<table>
<thead>
<tr>
<th>Problem identified</th>
<th>Breaking guideline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Looking for the search-button, do not want to go through each alternative displayed as an image</td>
<td>Simple and Natural Dialogue</td>
</tr>
<tr>
<td>Attention is automatically drawn to where least amount of text is presented</td>
<td>Simple and Natural Dialogue</td>
</tr>
<tr>
<td>Empty buttons/products missing label or image</td>
<td>Simple and Natural Dialogue</td>
</tr>
<tr>
<td>Too much and unnecessary text</td>
<td>Simple and Natural Dialogue</td>
</tr>
</tbody>
</table>

In Table 4.6, one of the problems identified was ”Empty buttons/products missing label or image”. This problem occurred repeatedly and disrupted a natural flow and simple appearance of the system.

Other guidelines which was broken was following:

<table>
<thead>
<tr>
<th>Problem identified</th>
<th>Breaking guideline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Products not organized or structured in a clear way</td>
<td>Speak the User’s language</td>
</tr>
<tr>
<td>When searching, it is not clear what the ‘C’ button does</td>
<td>Speak the User’s language</td>
</tr>
</tbody>
</table>
Heuristic Evaluation 2
According to the heuristic evaluation, the resulted system broke three of the guidelines.

- Speak the User’s language
- Provide feedback
- Provide clearly marked exits

The most frequently broken guideline was *Provide Feedback*. Some of the users wanted clearer instructions about what process was running. Several users pointed out that they expected a clearer message of when the identification was performed.

Table 4.10: Guideline broken: Provide Feedback

<table>
<thead>
<tr>
<th>Problem identified</th>
<th>Breaking guideline</th>
</tr>
</thead>
<tbody>
<tr>
<td>No clear instructions</td>
<td>Provide Feedback</td>
</tr>
</tbody>
</table>

Another guideline which was frequently broken was *Provide Clearly Marked Exits*. This occurs when the user is supposed to click the product to get a label.

Table 4.11: Guideline broken: Clearly Marked Exits

<table>
<thead>
<tr>
<th>Problem identified</th>
<th>Breaking guideline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does not tell the user that the identification is done</td>
<td>Provide Clearly Marked Exits</td>
</tr>
<tr>
<td>No clear printing instruction</td>
<td>Provide Clearly Marked Exits</td>
</tr>
</tbody>
</table>

Heuristic Evaluation 3
The third heuristic evaluation was performed on five test persons. Three of the volunteers had not tested the system before. According to the result, no guidelines were broken and all test persons completed the identification within seconds.
Performance

The fastest, slowest and average time of identifying a fruit or vegetable in both systems are compared. The time is measured to when the user puts the product on the scale to when the label is printing. This test is performed on 10 users.

Table 4.13: Time comparisons between existing and resulted system

<table>
<thead>
<tr>
<th></th>
<th>Existing System</th>
<th>Resulted System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slowest</td>
<td>28.0 s</td>
<td>20.0 s</td>
</tr>
<tr>
<td>Fastest</td>
<td>7.21 s</td>
<td>5.77 s</td>
</tr>
<tr>
<td>Average</td>
<td>14.6 s</td>
<td>10.10 s</td>
</tr>
</tbody>
</table>
Chapter 5

Discussion

To begin with, we are satisfied with the outcome of this project and the experiences we have gained. The project was in level with the degree of difficulty expected, but more comprehensive than expected.

The goals for this project has been to create a system more user friendly compared to the existing systems. To classify a graphical user interface as user friendly, is almost impossible. That is a conclusion every individual has to do herself. However, there are methods to find some usability flaws in a graphical user interface.

Comparing the usability of two systems gives intentions of if one of the systems is more user friendly compared to the other one. The idea of creating a system implemented with image classification aroused when using the existing systems. We experienced them as:

- more complicated than necessary
- to many options available
- no natural flow
- unnecessary amount of interactions

We wanted to create a system which we preferred to use over the existing systems, and where other people shared our opinion.

5.1 Results

5.1.1 Usability Evaluation

Compared to the existing systems, the heuristic evaluation indicated of a positive result. The users liked the design and thought there was improvements of the appearance compared to the existing ones. The first plan was to make one evaluation of the existing system and another evaluation of the created system. However, we did not know if our finished system was easy for anyone else to understand. Therefore, we decided to make a Prelaunch Test and a second heuristic evaluation to see if the system needed improvements. We performed the tests, made some changes and considered the system as finished.

The Prelaunch Test and Heuristic Evaluation 2 was shown to be very useful. Our system lacked some functions and the users had some valuable inputs which made us change in the design. The most common comment made about the system was that the display was too small. A consequence of a small display limited the font size and the product item size. Implementing a bigger display is a part of further development. The Prelaunch Test and the Heuristic Evaluation made us realize that there was several usability flaws in the system.
However, according to the Prelaunch Test, 86% percentage mentioned they would use our system instead of the existing ones in the retail business.

The Heuristic Evaluation concluded that there was fewer of the guidelines broken compared to the existing system. However, the produced system included usability flaws which could not be found in the existing system. This is probably because we focused on solving the problems in the existing system and by doing so, we created new ones.

The guideline Minimize the User’s memory load was discovered broken in the existing systems due to the comment “Users should not need to remember whether the fruit is ecological or not”. However, the resulted system would break this guideline as well since the produced system does not have the ability to see the difference of a ecological fruit or a regular fruit.

To conclude, the Heuristic Evaluation indicated that there was no usability flaws. Nevertheless, the usability tests helps the GUI creators to find about 50-70% of the usability flaws. This means there is a risk that the system includes usability flaws not discovered by testing. As Nielsen states in his study of testing several user interfaces:

“So even in the best case only half of the problems were found, and the general case was rather poor. Actually, even these numbers are not all that bad. Even finding some problems is of course much better than tiding no problems, and one could supplement the heuristic method with other usability engineering methods to increase the total number of problems found.”

Our interface may include usability problems. However, the problems are reduced and as Nielsen proposes, we created another usability test. For the final product, other kinds of usability tests will be implemented.

Networks

The results from the networks are as expected. Since MobileNet is developed for mobile applications, it makes sense for the network to have less propagation time. Inception is developed for the Large Scale Visual Recognition Challenge held by ImageNet and has more focus on accuracy. However, the differences in accuracy between the networks is not as large as the difference between the propagation times. MobileNet propagates images significantly faster with almost the same accuracy. It can also be discussed if more accurate predictions than MobileNet is needed. The confusion matrix for the top 3 predictions show that MobileNet has a top 3 accuracy of 97%. However, even though the top 3 accuracy is great, MobileNet still has difficulties in predicting clementines and kiwis. Compare this to Inceptions top 3 confusion matrix and we see that the overall accuracy is slightly less, but has more classes with 100% accuracy.

When measuring propagation times, the first image of each test for both networks had a significantly longer propagation time. This is speculated, but not confirmed, to be because the application is not loaded into the cache memory. Once the first image has propagated, the network has all the operations loaded into cache and thus can access them faster.

Initially, VGG was also a network that would be compared to MobileNet and Inception. However, it turned out to be to large of a challenge to train a VGG network. Tensorflow did not provide open source scripts to retrain the network which led to that the project group had to write the script themselves. To write the retraining script, a deeper understanding of how to use the Tensorflow python library had to be gained. An attempt at retraining VGG gave no result. It ended up to be too much work for an already large project. Although VGG was not retrained, the results can be speculated upon. VGG is even larger than Inception which means the propagation time of VGG would probably have been longer than the propagation time of Inception. However, the propagation time of VGG would probably have led to better accuracy. Nevertheless, since the difference in accuracy between Inception and MobileNet is small compared to
the propagation, a similar relation between Inception and VGG is expected. In conclusion, VGG would probably have a longer propagation time but a small improvement in accuracy.

**Dataset**

The data set turned out to be quite good even though only a small percent of the images were from the working environment. The difficulties the networks have in differentiating clementine and orange is because the two fruits are very similar in appearances. Adding more images to these classes would probably not improve the accuracy by much. Additional input such as weight could instead be taken into consideration to better differentiate these two fruits. As for kiwi, the accuracy could be improved by creating a new set of images for this class. Currently, a majority of the images depicts kiwis being cut open and showing their green insides. These images does not depict what the kiwi looks like in the working environment and the network will thus not recognize the fruit.

**5.2 Further Development**

**User Interface**

As three to five test subjects find close to 70% usability problems with the heuristic evaluation of a user interface, more tests has to be performed. After five test subjects, the heuristic evaluation reaches the point of diminishing returns[14].

**Optimize the Network**

Performing retraining on data sets from its actual environment could get the network more accurate. Using data sets fetched from ImageNet resulted in pictures of fruits and vegetables in varying environments. Since the classifier is suppose to work in a retail store, it will never encounter images of for example a forest. Therefore, training images that differ from the working environment could probably confuse the classifier to what parts of the image are features of a fruit or vegetable. Training on data sets from the actual environment would optimize the network and is recommended in several tutorials and installation guides for TensorFlow and OpenCV. Published work described in section 2.1 only use images taken in the working environment when retraining the network [1][23]. However, the method of transfer learning is still applied to decrease the number of images that has to be captured. Retraining a network from scratch, without applying transfer learning, is also a possibility but seems to only marginally improve accuracy[30].

If we collected images from the environment the system is supposed to perform in, we could probably increase the accuracy. Furthermore, if more images are collected, classes can be split into subclasses containing different types of a fruit or vegetable. For example, the apple class could be split into subclasses such as Granny Smith, Pink Lady and Royal Gala which are all types of apples. However, there is a risk that splitting a class into subclasses for each type of fruit or vegetable turns out to be too challenging for the network to classify. It is also almost impossible to predict the outcome of the retraining, even though millions of training samples are provided. Perhaps a more predictable behaviour can be achieved by implementing a series of networks. The first network only decides what kind of fruit it is. The task of classifying the subset of fruit is left to one of many succeeding networks specialized on a single kind of fruit. Methods like these are discussed in *Deep fruit detection in orchards*[1] and *Deepfruits: A fruit detection system using...*
Deep neural networks[23] shows reasonable qualitative performance for a system with similar functionality.

Another method of increasing accuracy could be to take multiple images and classify each one. A form of voting system is then implemented after the classification of each image. The class that has the best ranking amongst all classifications will become the final answer. A variant of this method could be to take a single image and apply different preprocessing techniques to each image and then implement the voting. An obvious downside of this method is the increased computation needed compared to only classifying a single image. Additional computation leads to an increased time to use the system which might only provide a worse user experience.

Finally, more complex methods of fruit identification exists. Faster Region-based CNN (R-CNN) has been shown to give accurate results in orchards[1][23] and could be a better alternative to a CNN.

Data Transfer

One of the nine guidelines when developing an interface is to always notify the user of what is happening. If some process of the system fails to complete or simply crashes, the user should be notified of this event. In the system, there is no way to notify a user of errors that happened in the back end. This is because of the way the back end sends data to Node.js. By send data through HTTP requests, no communication channel is established where both parts are aware of each other. Node.js simply redirects data received. Should the back end by any means not be able to send data to Node.js, it is impossible to tell that something in the back end went wrong. From the perspective of Node.js, the middleware is simply unaware that anything has attempted to issue a HTTP request. Node.js is thus simply unable to convey information to the user that something went wrong.

The problem can be solved by redesigning the way the back end sends data to the Node.js. A socket implementation, as done between Node.js and the front end, would enable the possibility to watch for disconnects on either end of the channel. If a disconnect is registered, both the back end and Node.js are aware of the event and can handle the disconnect.

Database

The project was supposed to include a database to fetch fruits and vegetables from. However, since the project was bigger than expected, the database became the least priority due to its low importance. The database is a part of further development and will be integrated in the future.

Hardware and Physical Appearance

According to the usability tests, a bigger display was desirable. Due to the strict budget of this project, a small touch screen was chosen. Using a 13-inch screen instead, would give the user an interface displaying larger buttons and bigger font size. However, a larger display may affect the processing time.

The case used for the screen and Raspberry Pi lacks some obvious security. At the back of the system, there is visible cables between the display and the Raspberry Pi. Furthermore, the cable between the camera and the processor is visible and easy to remove if wanted. The camera is attached to the case with a small pressure and removed with low traction.
Policies, Laws and Regulations

Laws and Regulations

To get this product into business, several concerns needs to be taken into consideration. Some of the most important concerns are the laws and regulations. In may 2018, the General Data Protection Regulations (GDPR) will be enforced and replaces the old regulation PUL. The major change concerns companies dealing with sensitive data such as personal code numbers, name, bank details, photos or e-mail addresses. Since this product includes a camera to take photos, there might be a risk of catching a persons’ face in the picture.

The purpose of following the regulations of camera usage within an organization is to protect the integrity of people. This product will not be placed in an environment with free access for the public, it will be placed inside a store. In Sweden, the Swedish authority Länsstyrelsen\footnote{http://www.lansstyrelsen.se/} demands a report of camera surveillance and would be enough for this product to be accepted on the market. Whether it is the responsibility of the product owner or the store owner, is an investigation to do in the future.

Sustainability and Policy

As future engineers, we have the responsibility to create and develop with the aim to be as sustainable as possible. To contribute to a sustainable development, we need to have the ability to design and manage the product with regard to people’s conditions and needs. Furthermore, follow the goals of social and economic guidelines of the society. Pursuing these guidelines of sustainable development, the following points will be a part of the policy and responsibility.

Ecological Sustainability

The footprint of the product will be calculated and compared to existing products. The footprint calculates the impact an organization has on the environment and is a guideline of if a business is using more of earth’s resources than exists. In our case, it would include the production of the hardware, the shipment and transports of the hardware and the energy used to maintain and develop the system. The aim is to operate the business with least possible footprint and compensating for the gas emissions of the production.

Furthermore, the transportation when delivering or selling the product to retail business has an effect on the environment. Contributing less to gas emissions, alternative fuel vehicles will be chosen. Since gas emissions is a global problem and essential to reduce for the future, it plays a big role for us if this product will be launched on the market. The environmental responsibility includes recycling of waste, renewable energy use and systematic work to follow and develop energy use. Moreover, selecting producers and factories having environmental goals over selecting the most affordable suppliers is an action to take.

Social Sustainability

To fulfill the strive for being a business taking social responsibility, relationships between customers, other business, manufacturers and the people within the organization will be held with the goal of being supportive and cooperative. People whom we come in contact with will be treated likewise no matter of origin, sexual orientation, gender, skin color, religion or opinions.

\footnote{http://www.lansstyrelsen.se/}
Economic Sustainability

The product development will be operated to pursue economic stability. Most important is to handle material and human resources with the aim of being of prolonged disposal. The development will be operated to achieve constant growth but must not be at the expense of social or ecological valuations.
Chapter 6

Conclusion

Using computer vision to improve the identification process of fruits and vegetables by self-service systems in the retail market has been a success. MobileNet provided fast identification results with accurate predictions. Some fruits and vegetables was harder to label than others and the reason for this can be traced back to the data set used when retraining the network. The graphical user interface solved some of the usability problems in the existing systems but also created new ones. Overall, the number of usability problems has decreased. The system still needs further development in the back end and front end to cover performance and usability problems. This development would have been done by the project group if more time was available. However, the project has come an unexpectedly long way considering the scale of the project and that the project group had almost no experience in at the start.

The rest of this chapter will be dedicated to answer the questions at issue and evaluate the goals and functional requirements.

Goals

Shorten the time it takes to identify a product

Average, shortest and longest time compared identifying a fruit or vegetable was better with the produced system. Hence, the goal was reached.

Minimize the number of usability problems which occurs in existing systems.

The amount of usability problems occurred in the existing systems was five. The amount of usability problems of resulted system was zero. Hence, the goal was reached.

Functional Requirements

The system should, with the help of an image classifier and a camera, identify at least five different number of product

The system uses MobileNet as a classifier and can classify 10 different kinds of fruits or vegetables with various accuracy. Images for the classifier are provided by the Raspberry Pi Camera Module v2 and is connected to the Raspberry Pi.

The system should have GUI for the user to interact with

A graphical user interface has been developed using React. The interface has solved some of the usability problems in the existing systems. Further devel-
The system should be able to fetch products from a database
The system should include an activator to activate the software
Questions at Issue
Is the Raspberry Pi suitable for managing neural networks?
Which CNN is suitable for the system?
Which problems occur with the chosen CNN?
How do we handle a situation where the classifier provide incorrect results?
What flaws of the usability was found in the existing systems and reduced in the resulted system?

- Simple and Natural Dialogue
- Be Consistent
- Provide Shortcuts
References


Appendices
## Time Plan

![Time Plan]

### Figure 1: Visualization of time spent on each task.
CMC Result Diagrams
Data set

Images from ImageNet

Self collected images
24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales

DESCRIPTION

Based on Avia Semiconductor’s patented technology, HX711 is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.

The input multiplexer selects either Channel A or B differential input to the low-noise programmable gain amplifier (PGA). Channel A can be programmed with a gain of 128 or 64, corresponding to a full-scale differential input voltage of ±20mV or ±40mV respectively, when a 5V supply is connected to AVDD analog power supply pin. Channel B has a fixed gain of 32. On-chip power supply regulator eliminates the need for an external supply regulator to provide analog power for the ADC and the sensor. Clock input is flexible. It can be from an external clock source, a crystal, or the on-chip oscillator that does not require any external component. On-chip power-on-reset circuitry simplifies digital interface initialization.

There is no programming needed for the internal registers. All controls to the HX711 are through the pins.

FEATURES

- Two selectable differential input channels
- On-chip active low noise PGA with selectable gain of 32, 64 and 128
- On-chip power supply regulator for load-cell and ADC analog power supply
- On-chip oscillator requiring no external component with optional external crystal
- On-chip power-on-reset
- Simple digital control and serial interface: pin-driven controls, no programming needed
- Selectable 10SPS or 80SPS output data rate
- Simultaneous 50 and 60Hz supply rejection
- Current consumption including on-chip analog power supply regulator: normal operation < 1.5mA, power down < 1uA
- Operation supply voltage range: 2.6 ~ 5.5V
- Operation temperature range: -40 ~ +85℃
- 16 pin SOP-16 package

APPLICATIONS

- Weigh Scales
- Industrial Process Control

Fig. 1 Typical weigh scale application block diagram
HX711 Amplifier Datasheet

**VCC**: 2.7-5.5V

**VBG** = 1.25V

**AVDD** = VBG \( \frac{R1 + R2}{R1} \)

Default: Closed - Data rate set to 10SPS
Open jumper to set to 80SPS

Increases noise per read

**E+**

**E−**

**A−**

**A+**

**Red**

**Black**

**White**

**Green or Blue**

Common Load

Cell Colors:

**IO**: 2.7-5.5V

**Shield**

**Yellow**

**VDD**: 2.7-5.5V

**VCC** vs **VDD**:

**VCC** is the main supply voltage, while **VDD** sets the digital logic voltage reference and should be connected to microcontroller supply voltage, or shorted to **VCC**.
Adam Olsson - Computer Science Engineer

Frida Femling - Computer Science Engineer