universität
uulm

Faculty of Engineering, Computer Science and Psychology
Institute for Databases and Information Systems

Implementation of an iOS app for image-based classification of selected substances in organic waste with the help of an AI system

Bachelor Thesis at Ulm University

Submitted by:
Marlene Mika
marlene.mika@uni-ulm.de
1042495

Reviewer:
Prof. Dr. Manfred Reichert

Supervisor:
Dr. Marc Schickler

2023

Version October 24, 2023

Satz: PDF-LaTeX 2$_\varepsilon$

Abstract

The rapid development of Artificial Intelligence (AI) has led to groundbreaking innovations that are already an integral part of our everyday lives. As AI becomes more prevalent, exciting prospects are opening up to make our daily lives even more efficient and convenient. Nevertheless, we are facing new challenges, including the increasing amount of waste and its pollution. To address these issues, we developed an iOS app in this thesis that uses AI technologies to recognise and classify objects into the categories "organic waste" and "non-organic waste". With the help of a study, we were able to find out that waste separation is also an important topic for the population and that such an app is desirable in cases of uncertainty regarding disposal. Thus, a reduction of foreign substances within the organic waste can be supported. The accuracy of the AI plays a major role in gaining the trust of the users.

Contents

List of Figures

List of Tables

List of Code Snippets

Acronyms

AI Artificial Intelligence. iii, viii, ix, 1, 2, 3, 6, 7, 8, 9, 12, 13, 14, 17, 18, 19, 36, 37, 41, 46, 69, 70, 71, 72, 73, 74

ANN Artificial Neural Network. vi, ix, 2, 17, 21, 24, 27, 36, 46, 73

AP Average Precision. ix, 35, 40

CNN Convolutional Neural Network. vi, ix, 27, 30, 31, 32, 36, 73

DNN Deep Neural Network. ix, 25, 26

FN False Negatives. ix, 34

FP False Positives. ix, 34

GPUs Graphical Processing Units. ix, 46

GUI Graphical User Interface. ix, 50

IDE Integrated Development Environment. ix, 50

IoU Intersection over Union. ix, 34

mAP mean Average Precision. vi, ix, 33, 34, 35, 36, 40, 47, 49, 64

NMS Non-Maximum Suppression. ix, 35, 49

ReLU Rectified Linear Unit. ix, 22, 23, 28, 30

ROI Region of Interest. ix, 32

RPN Region Proposal Network. ix, 32

SIEBW State Institute for the Environment of Baden-Württemberg. ix, 5, 41

SSD Single Stage Detector. vi, ix, 32, 33, 36, 37, 38, 73

SSMD Single Stage MultiBox Detector. ix, 37, 38, 39, 40

TN True Negatives. ix, 34

TP True Positives. ix, 34

TSD Two Stage Detector. vi, ix, 32, 33, 36

UI User Interface. ix, 50, 53, 54, 64

YOLO You Only Look Once. vi, ix, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49

1 Introduction

The rapid development of Artificial Intelligence (AI) in recent years has undoubtedly led to a real breakthrough. Chatbots like ChatGPT are a prime example of the diverse applications of AI that suddenly appeared in our everyday lives and revolutionised our interactions with digital systems. Yet AI is by no means new to our lives. Even before ChatGPT, there were applications such as voice assistant Amazon Alexa or AI-controlled computer opponents in video games that can adapt to the player's abilities.
With the current state of development of AI, we are on the doorstep of even more exciting possibilities. Advances in machine learning, neural networks and deep learning have expanded the capabilities of AI systems to handle complex tasks in any walk of life.

However, while AI is becoming more present in many areas of our lives, we are also facing new challenges. One of these challenges concerns the increasing amount of organic waste and the foreign matter it contains due to incorrect disposal, which pollutes our environment. Within only 3 years, the amount of organic waste has increased from 9.9 million tonnes to 11.2 million tonnes. The proportion of foreign matter is above the desirable tolerance limit, which reduces the recycling rate.

These facts make it all the more urgent to develop innovative solutions that enable the management of increasing organic waste volumes and the reduction of foreign matter in this waste stream. In this context, a promising solution is emerging: the development of an iOS app that combines both facts and helps consumers dispose of their organic waste more efficiently while promoting clean waste separation.

In this context, we conduct the conception and implementation of such an iOS app within this thesis. For this purpose, in chapter 2 we first take a closer look at the development of the popularity of AI and the development of organic waste generation in recent years, in order to then get an idea of which functionalities are desirable in

such an app.
Through a requirements analysis of our app idea, we then define the functional and non-functional requirements for the implementation in chapter 3.
For the use of an AI we need a thorough understanding of how it works and how an AI can be used in our app idea. Therefore, in chapter 4 we will focus on how AI works and how it can be used in our app idea. Here, we will focus on different aspects such as Machine Learning, Artificial Neural Networks and Deep Learning in order to develop a deep understanding of the technologies that will run our app.
In the following chapter 5, we turn to the actual implementation of AI and iOS app. For this, after choosing a suitable AI architecture, we explain the composition of the dataset and then train the AI to recognise organic and non-organic waste objects and evaluate the performance. Following this, we present the architecture of the implemented app and the implementation of the functional and non-functional requirements.
In the following chapter 5, we turn to the actual implementation of AI and iOS app. For this, after choosing a suitable AI architecture, we explain the composition of the dataset and then train the AI to recognise organic and non-organic waste objects and evaluate the performance. Following this, we present the architecture of the implemented app and the implementation of the functional and non-functional requirements.
With the help of a study in chapter 6, we will collect feedback on the app as well as information on the personal relation to waste disposal of the study participants. Furthermore, we will investigate the acceptance of the participants regarding the AI, as Artificial Intelligences can still generate erroneous outputs.

2 Motivation

This chapter focuses on the current situation in the field of organic waste in Germany. In the course of the chapter, we will see that the volume of organic waste has increased continuously in recent years. Furthermore, we will realise that the purity of organic waste is not sufficient to ensure a high recycling rate, which shows a need for action.
Furthermore, we will look at the development of the popularity of Artificial Intelligence (AI) in the past decades. By analysing search trends, we will see that there has been a significant increase in interest in AI over the last year, especially due to the introduction of the chatbot ChatGPT.
We will then combine these two aspects to develop the idea of an iOS app that aims to improve the purity of organic waste. This app will be used as part of a user study to investigate user acceptance of AI and whether users are more likely to value quality or quantity when classifying objects.

2.1 Organic Waste Situation in Germany

With regard to organic waste disposal in Germany, there has been an increasing trend in recent years. Since 2018, there has been a continuous increase in the total volume of organic waste. This increase is exemplified by the following statistical data (see also Table 2.1): In 2018, the volume of organic waste generated throughout Germany was 9.9 million tonnes, according to the Federal Statistical Office. Within just two years, this amount increased by an additional 0.7 million tonnes, and one year later, the Federal Republic again recorded an increase of 0.6 million tonnes, bringing the total volume to 11.2 million tonnes in 2020. The consequence of this development is that more organic waste is also generated per individual.

In 2021, the annual amount was 134 kilograms per person, whereas three years earlier it was 120 kilograms [1, 2, 3].

Year	2018	2019	2020	2021
Collected organic waste (in million)	9.9 t	10.2 t	10.6 t	11.2 t
Increase compared to previous year	-	0.3 t	0.4 t	0.6 t
Amount per person	120 kg	122 kg	128 kg	134 kg

Table 2.1: Organic waste generation by households in Germany from 2018 to 2021 [1, 2, 3]

The separate collection of organic waste enables the recycling of the organic substances and nutrients it contains. These are produced in the form of composts and fermentation residues and are used in horticulture and agriculture.

For the recycling of the waste, the German Federal Environment Agency presents three different processes, depending on the composition of the organic waste:

1. Highly structured and slightly moist plant material is used to produce compost and is transported to composting plants for this purpose.
2. Solid to liquid organic waste that is easily degradable and has little structure, such as food waste, can be used to produce biogas or fertiliser. For this purpose, it is processed in fermentation plants.
3. The utilisation of dry woody plant material in biomass cogeneration plants serves to generate energy and to produce fertiliser in the form of ash.

In 2020, according to the Federal Environment Agency, about 1,000 composting and 100 pure organic waste fermentation plants recycled the collected waste. More than half of the waste could be composted, but without energy recovery. The Federal Environment Agency therefore aims to increase recycling using the fermentation method, which can also produce biogas.

Certain limit values and requirements under the Organic Waste Regulation, set by the Federal Ministry of Justice, are binding for compost and fermentation residues. These regulations serve to limit the maximum permissible content of heavy metals in compost and fermentation residues to ensure that they can be used safely. Furthermore, it is prescribed that both the pasteurisation of compost and fermentation

residues and the killing of seeds and germinable plants must take place in order to ensure hygienic quality [4].

Since 2017, there has also been a limit on the amount of foreign matter in the products produced. Accordingly, a maximum of 0.1 per cent by mass of the dry substance may consist of mouldable plastics such as foil components and 0.4 per cent by mass of other foreign substances such as glass or metal [4].

Compliance with these limits should, of course, not be seen as a maximum tolerance. Rather, the primary goal is to keep organic waste as pure as possible and to minimise incorrect discards and the associated introduction of foreign matter. This helps to keep the recycling rate at the highest possible level.

In 2017, the State Institute for the Environment of Baden-Württemberg (SIEBW) carried out a data collection which, among other things, analysed the proportion of foreign matter in organic waste in two different districts, the Schwarzwald-Baar district and the Ludwigsburg district. The results showed that the amount of foreign matter in the Schwarzwald-Baar district was 2.5 per cent by weight and in the Ludwigsburg district 2.0 per cent by weight. According to the guidelines of the Federal Compost Quality Association, organic waste is considered to be sufficiently pure if the foreign matter content is less than one percent by weight. In addition, other undesirable materials such as bags made of biodegradable plastics or packaged food were identified. The use of such materials is due to the disposal of wet kitchen and food waste.
Based on these results, SIEBW called for urgent action regarding the foreign material content in organic waste in order to improve the quality of the collected organic waste [5].

Due to the continuous increase in organic waste generation per person since 2017 (see Table 2.1), there is a risk that the foreign matter content will also increase, which may also affect the recycling rate. This development underlines the need to take urgent measures to reduce the amount of foreign matter in organic waste and to continuously improve the quality of collection and recycling.

2.2 Rising Popularity of Artificial Intelligence

In recent years, interest in Artificial Intelligence has increased significantly. Looking at the progression of worldwide interest in the term "Artificial Intelligence" on Google Trends over the last five years, it is clear that there has been an impressive increase since the end of November 2022. This remarkable increase is likely to be mainly due to the release of the chatbot ChatGPT, which made its debut during this period [6]. An interesting observation is the fact that the interest trend of ChatGPT on Google Trends is very similar to the "Artificial Intelligence" one (see Figure 2.1) [7].

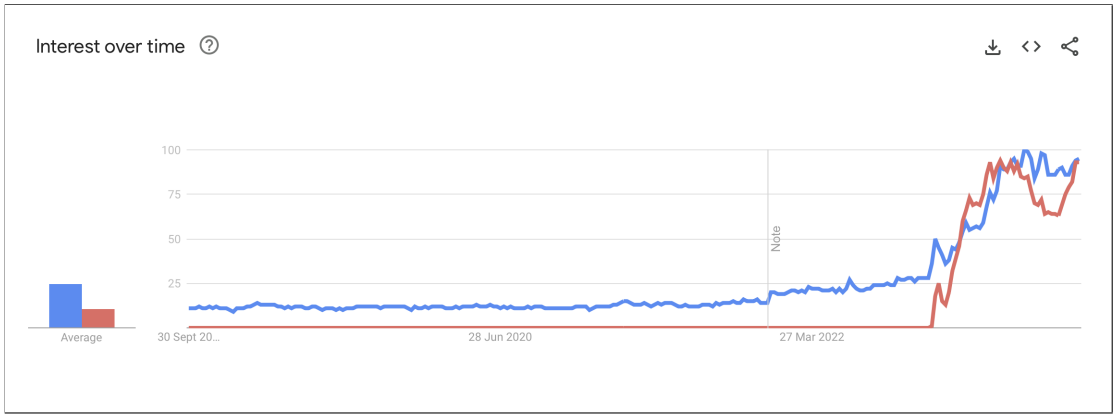


Figure 2.1: Global interest plot of "Artificial Intelligence" (blue) and "ChatGPT" (red) from September 2018 to September 2023 [7]

Both the general topic "AI Chat Open Assistant Chatbot" and the search term "chatgpt" show the largest increase in worldwide search frequency on Google Trends. Google Trends classifies this as a "breakout" and justifies this with the novelty of this topic [8].

In fact, it is clear from Figure 2.1 that Artificial Intelligence is by no means a novelty, as there was already a mild interest in this topic before ChatGPT was introduced. In fact, AI has been an integral part of our everyday lives for quite some time. Examples include spam filters in email inboxes, computer-controlled opponents in video games and also voice assistants such as Amazon Alexa, which are based on AI algorithms [9, 10, 11].

Given the numerous possible uses of AI and its growing importance in human-machine interaction, it is becoming increasingly clear how versatile AI technologies

are. This is reflected in an increased public attention to developments in the field of AI, which emphasises the relevance and impact of AI technologies on our present society. With the continued development and integration of AI systems, innovative application areas and the prospect of enhanced human-machine interaction continue to open up. This promises a future where AI will play a significant role in various industries and areas of life.

2.3 Idea: AI-supported app for improving organic waste purity

Due to the increasing amount of organic waste and the associated risk that the proportion of foreign matter will also increase, the need for education is also growing in order to restore the cleanliness of the waste and thus be able to generate a higher recycling rate. Due to the growing popularity and the diverse application possibilities of AI, the idea arises to develop an iOS app that links the aforementioned aspects. The following two use cases have been elaborated to be the highest relevance for the user:

- **Use Case 1: Checking organic waste for incorrect discards**
  In this use case, a user asks the app to check his organic waste for possible incorrect disposal. The user takes a photo of the organic waste, which is then analysed by the integrated AI. The AI identifies only the foreign matter that is detected in the photo and highlights it in red in the display. In addition, the user is provided with an accompanying information text to learn about the foreign matter detected.
- **Use Case 2: General classification of objects**
  This use case involves a user asking the app to clarify the correct disposal of an item. Here, the user opens the live view of the camera and points it at the item in question. The app's AI performs a classification and marks the item either red (for non-organic waste) or green (for organic waste). Parallel to this, an information text is also provided here to inform the user about the classification of the item.

The functionalities of the app are to be completed by the following points:

- In both use cases, there is a choice between using the live view of the camera and taking a single photo for analysis.
- In addition, the application allows the selection of an existing photo from the gallery to perform the analysis in both examples.
- In the context of use case 2, it is also possible to analyse several objects at the same time for classification.
- For this thesis, the number of recognisable objects is limited to 20 due to the amount of data required. 10 organic waste products and the most typical 10 non-organic waste products should be recognisable.

In order to test the feasibility and relevance of this app idea, we set ourselves two central research questions that will be answered in the course of this thesis:

- **RQ1: Would such a waste classification system be accepted by the population?**
  This question looks at the willingness of users to accept a waste classification system even if it may have potentially incorrect classifications.
- **RQ2: When it comes to the objects that can be classified by the AI, does quantity or quality matter more?**
  This question aims to clarify whether the quality of the AI for classification (e.g. through high accuracy) or rather the quantity (number of recognisable objects) plays a role for the users.

For the implementation of the app idea, the first step is to carefully conduct a requirements analysis to ensure that all desired functionalities of the app can be successfully integrated. In the run-up to the presentation of the implemented app, we will deal in detail with the topic of Artificial Intelligence in order to be able to make a suitable choice of an AI technology based on this. Finally, after we have gained some insights into the finished app, we will address our research questions which will be investigated and answered within a user study.

3 Requirements Analysis

In the previous chapter, the consideration of the organic waste situation in Germany as well as the rising popularity of AI led to the idea of developing an app to facilitate the separation of organic waste and non-organic waste. Based on the description of 2.3, we now identify the required use cases that describe the user's interaction with the system. Based on these use cases, the associated functional and non-functional requirements are then derived. These requirements specifically describe the functions and tasks of the app to ensure that it meets the needs of the users.

3.1 Use Cases

In the following, we define the use cases that represent the interactions between the user and the system.

Use Case 1: Selection of image data for input
For object recognition within the app, an input is needed that is to be analysed. For this purpose, the user can choose from the following three options:

Use Case 1a: Take a new picture with the camera
This option opens the camera and the user can take a new photo.

Use Case 1b: Select an existing picture from the gallery
If the user selects this option, the picture gallery of the device opens. From there, the user can select an existing photo.

Use Case 1c: Switch on live view of the camera
This option also opens the camera, but there is no option to take a photo. Instead, object recognition is performed in real time.

Use Case 2: Choice of recognition mode
The user can choose from two different detection modes that adjust the analysis of the input material:

Use Case 2a: Detect only non-organic waste
In this mode, only objects that the system classifies as non-organic waste are marked.

Use Case 2b: Detect both organic and non-organic waste
With this mode, objects that are recognised as non-organic waste as well as those that are recognised as organic waste are marked.

3.2 Functional Requirements

Based on the use cases and the description of the planned functionalities in 2.3, the functional requirements can now be formulated. These define which tasks the system should fulfil.

ID	FR1
Name	Selection option between the capture options
Description	The app shall offer the user the central view in which the three capture options are displayed: "Take new photo with camera", "Select existing photo from image gallery" and "Live preview".
References	FR2, FR3, FR4

ID	FR2
Name	Taking a new photo with the camera
Description	The app shall allow the user to take a new photo with the camera of their mobile device and then use this photo to perform the analysis.
References	–

ID	FR3
Name	Select an existing photo from the image gallery
Description	The app shall give the user the possibility to select an already existing photo from the image gallery and to select and use this photo for the analysis.
References	–

ID	FR4
Name	Using the live preview of the camera
Description	The app shall provide a live view of the camera that allows the user to capture and analyse objects in real time. However, in this view there shall be no option to take a photo by pressing a release button.
References	FR13

ID	FR5
Name	Selection option of the mode for the sole detection of foreign matter in organic waste
Description	The app shall provide the user with an option to select the mode that allows for the detection of non-organic waste only. In this mode, the focus shall be on the identification of non-organic material in the organic waste in order to identify and sort out these foreign materials to ensure the purity of the organic waste.
References	FR7

ID	FR6
Name	Selection option of the mode for the general classification of objects into "organic waste" and "non-organic waste"
Description	The app shall provide the user with a possibility to select the mode that allows the general classification of detected objects into "organic waste" and "non-organic waste". In this mode, the focus is on education and prevention to inform the user about the correct waste separation and to prevent foreign matter from entering the organic waste.
References	FR8

ID	FR7
Name	Mode for the exclusive detection of foreign matter in organic waste
Description	The app shall support the mode that is exclusively focused on the detection of foreign materials in the organic waste. In this mode, the focus is on identifying and marking non-organic materials in the organic waste to enable correct separation and to ensure the quality of the organic waste.
References	–

ID	FR8
Name	Mode for the general classification of objects into "organic waste" and "non-organic waste"
Description	The app shall provide the mode that enables the general classification of detected objects into "organic waste" and "non-organic waste". In this mode, the focus is on education and prevention to inform the user which objects are suitable as organic waste and which are not.
References	–

ID	FR9
Name	Display the analysed image with the detected objects in corresponding bounding box, confidence and information text
Description	The app shall display the analysed image to the user with the detected objects marked accordingly, depending on the detection mode, to indicate organic waste and non-organic waste. In addition, information about the confidence of each detected object and an information text about the detected objects shall be displayed.
References	FR14, FR15, FR16

ID	FR10
Name	AI for image processing and object recognition
Description	The app shall use an Artificial Intelligence that supports image processing and object recognition. The AI forms the core of the app's analysis function.
References	–

ID	FR11
Name	Integratability of the AI on mobile devices
Description	The AI shall be integrated locally in the app to perform image processing and object recognition on the device itself. This is to ensure that data processing is done without the need for an online connection and the associated delays.
References	FR10

ID	FR12
Name	Detection of multiple objects simultaneously
Description	The AI shall be able to recognise several objects simultaneously on one image. This enables an efficient and fast examination of objects.
References	FR10

ID	FR13
Name	Object detection in real time
Description	The AI shall support image processing and object detection in real time. This gives the user instant feedback while pointing the camera at objects.
References	FR10

ID	FR14
Name	Highlighting of the detected object in the corresponding colour
Description	The AI shall mark detected objects on the screen. Objects classified as organic waste shall be marked in green colour, while objects classified as non-organic waste shall be marked in red colour. This visual marking shall facilitate recognition for the user.
References	FR10

ID	FR15
Name	Information about the confidence of the detected object
Description	The app shall display information about the confidence with which the AI has identified a detected object. This shall provide the user with insights into the reliability of the recognition.
References	–

ID	FR16
Name	Information text about detected object(s)
Description	The app shall provide an information text to educate the user about the identified objects and their correct disposal.
References	–

ID	FR17
Name	Additional information for the different modes
Description	The app shall provide a help function for the different detection modes, where there is a guide with additional information and explanations. This is to ensure that the user can use the functions of the app optimally and correctly.
References	–

ID	FR18
Name	List of recognisable objects
Description	The app shall present a list of recognisable objects that can be recognised by the AI. This provides transparency about which objects can be analysed.
References	–

ID	FR19
Name	Terms of use
Description	The app shall contain clear terms of use that define the conditions and rules for the usage of the app by the user.
References	–

ID	FR20
Name	App disclaimer
Description	The app shall have a disclaimer that clarifies the legal responsibilities and risks associated with the use of the app and provides legal clarity for users.
References	–

ID	FR21
Name	Privacy policy
Description	The app shall have a privacy statement that regulates the processing of images for analytics and privacy protection in accordance with applicable data protection regulations. This is important to ensure user trust in the app.
References	–

3.3 Non-functional Requirements

Final for the requirements analysis, the non-functional requirements for the system are discussed. These requirements refer both to quality criteria for specific functionalities and to requirements for the overall system.

ID	NR1
Name	Precision
Description	The app shall have high accuracy (>85%) in foreign matter and organic waste detection to minimise misclassification.

ID	NR2
Name	Usability
Description	The user interface of the app shall be user-friendly and inviting to use. Only then can it be ensured that the purpose of the app is fulfilled.

ID	NR3
Name	Reliability
Description	The app shall interpret user requests correctly so that the desired action is performed correctly.

ID	NR4
Name	Maintainability
Description	The system shall be designed in such a way that it is possible to troubleshoot or extend the system by documenting each method in the code.

ID	NR5
Name	Data privacy
Description	The app shall process and store all analysed images only locally on the mobile device, without transferring them online or to an external server, to ensure the security and confidentiality of the images and information and to protect the privacy of the users.

4 Artificial Intelligence Fundamentals

Section 2.2 already noted the emerging popularity of AI, especially through the chatbot ChatGPT, and some other application examples were also mentioned. This shows that AI technologies are increasingly present in our everyday lives. However, this raises the question of what actually lies behind the term Artificial Intelligence and how AI works.
In this chapter, we first define the term Artificial Intelligence and look at its historical development. We then turn to fundamental concepts such as machine learning. One of the most important subfields of machine learning is deep learning, which is often realised with the help of Artificial Neural Networks. We deepen our understanding of the basic structure of these networks and take a look at the learning process. An outstanding example of an Artificial Neural Network is the Convolutional Neural Network, which is mainly used in image classification. We will familiarise ourselves with its architecture before concluding with the topic of object recognition. Here, Convolutional Neural Networks play a crucial role as they enable the classification and localisation of objects.

4.1 Definition and History

According to the Oxford English Dictionary, AI describes "the capacity of computers or other machines to exhibit or simulate intelligent behaviour" [12]. This definition raises the fundamental question of what exactly is meant by intelligent behaviour. Alan Turing, a British mathematician, was already preoccupied with this idea in 1950, and developed the concept of the Turing Test based on this consideration. In

the Turing Test, a human referee acts via electronic communication and randomly asks questions to two participants, one of whom is human and the other a computer. The Turing test is considered passed if, after a series of questions and answers from both participants, the referee cannot tell which of the two is the computer and which is the human. In this case, the computer in question is considered "intelligent" [10, 13].

Paaß and Hecker criticise, however, that the Turing test in this form is not sufficient to prove the intelligence of the machine and must be extended. It should be noted that at the time Alan Turing introduced that test, AI was not yet particularly advanced and the idea of such an intelligence test was more of a theoretical milestone [10]. However, Turing was by far not the only AI researcher.

The development of Artificial Intelligence began in 1950 when Marvin Minsky and Dean Edmonds developed the first computer with a neural network [11, 14]. In 1956, the term "Artificial Intelligence" was introduced for the first time at a workshop at Dartmouth College organised by Marvin Minsky together with John McCarthy [10, 11, 13, 14, 15, 16]. The purpose of this workshop was to explore the characteristics of intelligence in detail in order to find out to what extent machines can imitate intelligence [13].

In the following decades, significant milestones were reached that have contributed significantly to the development of today's AI. These include the introduction of the perceptron by Frank Rosenblatt [10, 14], which is considered the precursor of machine learning [11]. Expert systems that focused on solving specific problems were used for the first time, for example DENDRAL for the recognition of chemical compounds [10, 11, 14]. Also, the first programmes for Natural Language Processing emerged, which enabled the solution of problems within the natural dialogue format, such as STUDENT for solving mathematical problems formulated in natural language [11], or ELIZA, one of the first chatbots [13, 15].

In the early 1970s, optimism prevailed in AI research, but this was followed by a period of fading interest in the following years, which became known as the "AI winter". This was attributed to the limited capabilities of AI as well as the lack of computing capacity at the time [11, 15, 16].

It was not until the 1980s that AI research flourished again with advances in neural network training (e.g. backpropagation) [11, 14] and further development of expert

systems. Despite their widespread use, expert systems were not able to establish themselves in the long term because their learning ability was limited and adaptation was complex [11, 15].

The 1990s brought the first intelligent agents, which found a wide range of applications, especially with the appearance of the internet [14, 15]. The emergence of Big Data, a term used to describe the enormous amount of data that suddenly became available through the internet, opened the path for the field of data analysis [14]. These techniques are still used today to display personalised content and make recommendations [11]. The first AI-based robots [14] and speech recognition software were also developed [16].

With the continuous increase in computing power, storage capacity [17] and advances in various fields, it has been possible to use ever more powerful AI technologies. Artificial Intelligence has since become increasingly integrated into our everyday lives, whether through spam filters in email inboxes or virtual assistants such as Alexa or Siri [9, 11].

Since the 2010s, the focus has increasingly shifted to the area of machine learning [15], which enables learning through experience and serves to save detailed programming effort [11].

4.2 Machine Learning

Like humans, who learn to ride a bicycle by experience or learn new terms by showing examples, this system of learning is attempted to be applied to computers [10]. A distinction is made between three different types of algorithms that are applied to enable so-called machine learning:

- **Reinforcement Learning**
  This learning method aims to learn the optimal strategy for a given problem. The basis for this is an incentive or reward function to be maximised. The system is not explicitly told which action is optimal for which situation. Instead, it receives feedback on the action taken after certain time intervals. The feedback is given with the help of rewards and punishments. A concrete application example of this approach is learning to play chess:

For this, the developer specifies the state of the environment (positions of the chess pieces) and lists all possible actions (possible chess moves) based on the environmental conditions (rules of the game) [15]. After each action performed by the system, the environment reacts (opponent plays) and the system receives new information about the state and a reward (e.g. positive points for good move/victory) or punishment (e.g. negative points for a bad move/defeat) [10]. Thus, the goal of the system is to find the moves that maximise the incentive function to ultimately win the game [15].

- **Unsupervised Learning**
  In this learning method, the system analyses available data and independently searches for patterns and regularities, grouping the data based on similarities [10]. An illustrative example of this is a collection of animal pictures:
  In this scenario, the system does not know which animal is in each picture, and it independently tries to identify categories. In doing so, the categories may not necessarily be ordered by animal type (e.g. dogs, cats), but may be grouped by other characteristics such as colours (e.g. black, white or brown animals).
  This learning method is used in various fields of application, such as compression methods. Here it is used to identify and filter out unimportant data components, which can make files smaller [15].

- **Supervised Learning**
  In the supervised learning method, the system receives clear instructions about what information it is supposed to learn. For this purpose, large amounts of data are provided that have already been annotated, i.e. labelled. After processing this data, the system develops patterns that enable it to make independent decisions when presented with new, unlabelled data. An example is a system that has to distinguish between pictures of cars and houses. To do this, it is provided with many pictures of both categories along with appropriate annotations about whether it is a house or a car [10]. After the model has been trained using this so-called training data, it is evaluated using a test data set to assess the performance of the system [15].

4.3 Artificial Neural Network & Deep Learning

Deep Learning is a subcategory of Machine Learning that is characterised by the use of an Artificial Neural Network (ANN). Unlike conventional machine learning algorithms, which often use flat models, deep learning allows the use of networks with multiple layers to capture complex patterns and abstractions in data.
The basic idea behind an ANN is to simulate the concept of information processing in the human brain. In our brain, excitations are picked up by neurons through dendrites and transmitted to the nucleus where the information is processed. The resulting excitation is then transmitted via the axon and synapses to the dendrites of other neurons [18]. An artificial neuron has a very similar structure (see Figure 4.1):

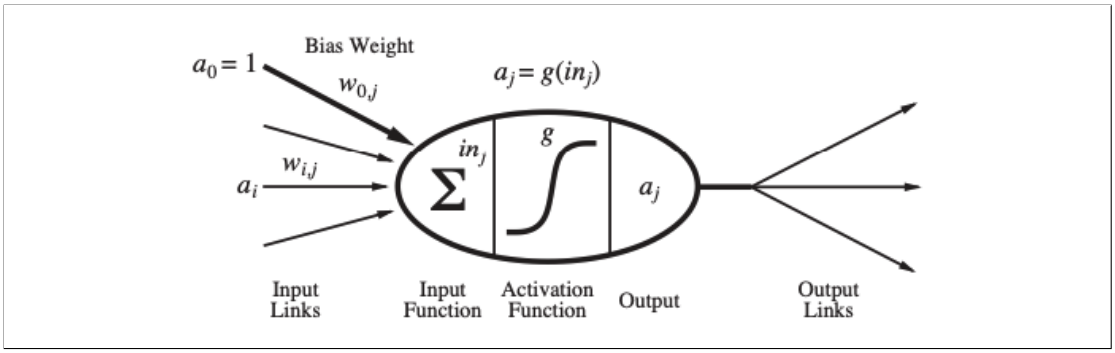


Figure 4.1: Representation of an artificial neuron [14]

- **Input links**: An artificial neuron has input links consisting of weighted connections $w_{i,j}$ to previous neurons a_i. In addition, it can also take direct input values from the network's environment. A standard input $a_0 = 1$ is often added, which is multiplied by an associated weight $w_{0,j}$.
- **Input function**: The neuron's input function sums the weighted inputs of all inputs, including the associated weights. This is done according to the following formula:

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i$$

- **Activation function**: To calculate the output value of neuron j, the following activation function is applied:

$$a_j = g(in_j) = g(\sum_{i=0}^{n} w_{i,j} a_i)$$

- **Output**: The output a_j forwards the calculated state of the activation function to the following neuron(s).

The activation functions can be implemented in different ways. Three commonly used approaches are the linear function, the threshold function and logistic functions [14].

As a linear function, the identity is usually used, which directly forwards the network input. The rectifier function also differentiates between negative and non-negative values, so that it returns 0 for numbers ≤ 0 and the identity for numbers > 0. This results in the function:

$$f_{ReLU}(x) = max(0, x)$$

The graph associated with the function can be seen in Figure 4.2a.
This so-called Rectified Linear Unit (ReLU) function has proven successful especially in Convolutional Neural Networks and is often used in this application.

The threshold function requires that a neuron must exceed a predefined threshold θ in order to be activated. This is represented by the following formula:

$$f_{threshold}(x) = \begin{cases} 1, x \geq \theta \\ 0, else \end{cases}$$

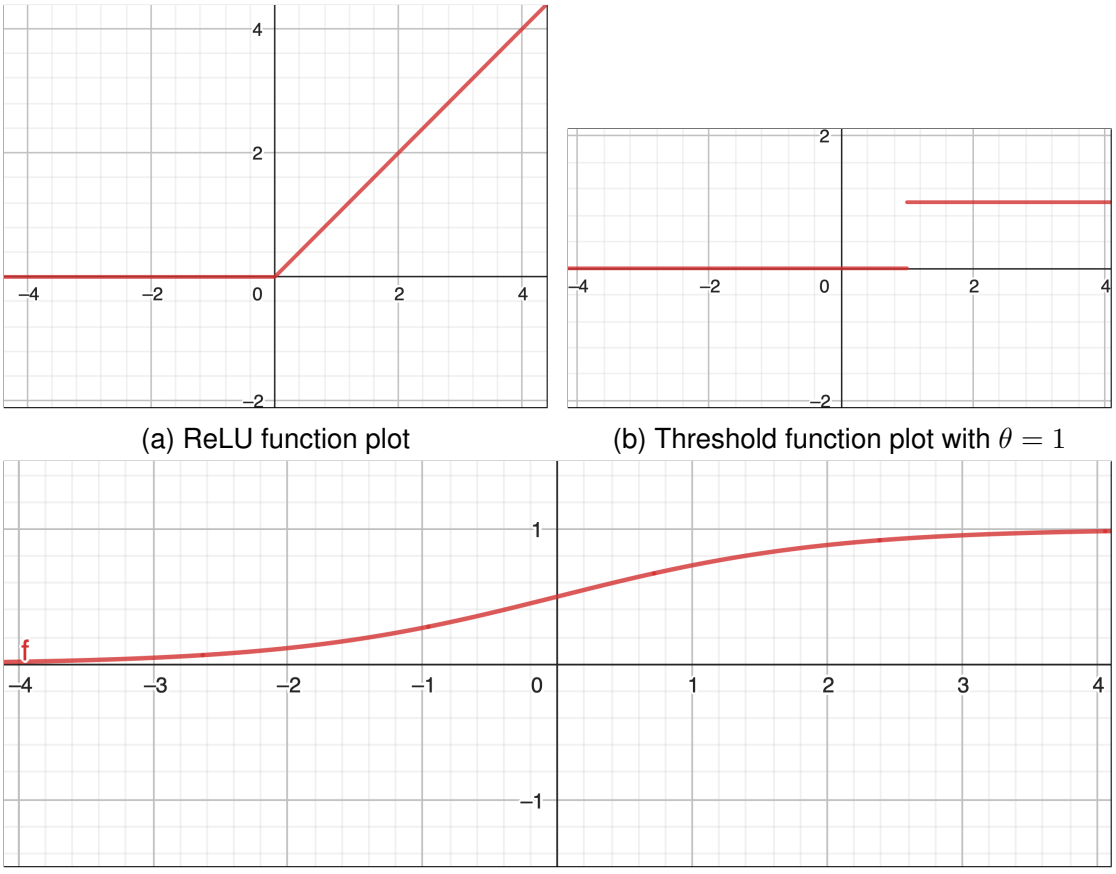
The graph associated with the function can be seen in Figure 4.2b.
If the threshold θ is exceeded, the neuron outputs the value 1, otherwise the value 0. Threshold functions are mainly used in simple networks.

Logistic functions, on the other hand, are differentiable, unlike the threshold function, which facilitates the learning process of the neurons. This differentiability plays

an important role in the learning process. Furthermore, logistic functions enable the generation of finely graded predictions. A frequently used logistic function is:

$$f_{logistic}(x) = \frac{1}{1 + e^{-c \cdot x}}$$

The graph associated with the function can be seen in Figure 4.2c. The parameter c influences the steepness of the curve at the transition from 0 to 1. Alternatively, the $tanh$ function can be used if an activation of the neuron in the range (-1,1) is desired. Neurons that use a logistic function are also called sigmoid perceptrons. This function is most commonly used with smaller, feed-forward neural networks [18].



(a) ReLU function plot

(b) Threshold function plot with $\theta = 1$

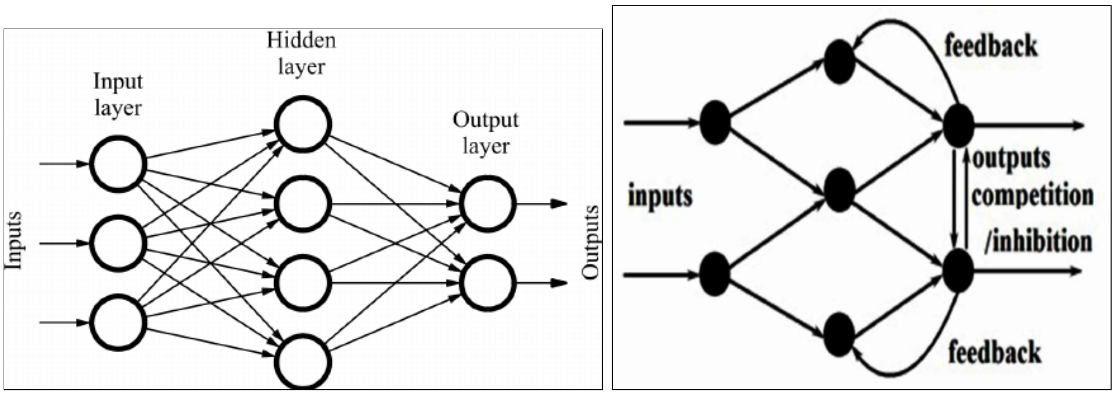
(c) Logistic function plot with $c = 1$

Figure 4.2: Function plots of activation functions, made with GeoGebra[1]

[1] GeoGebra Graphing Tool Website: `https://www.geogebra.org/graphing?lang=en`

Linking several such neurons results in an Artificial Neural Network [14] [18]. These networks can be divided into two basic architecture types:

- **Feedforward network**
  In this architecture, connections between neurons occur in a one-way direction, creating a directed acyclic graph in which no loops exist. This means that the output of one layer has no effect on the same layer. Each neuron receives input from previous layers and forwards its output to subsequent layers. This architecture represents a function of its current input and has no internal state other than weights [14]. Feedforward networks are particularly used in pattern recognition [19]. Such an architecture is shown in Figure 4.3a as a model for illustration.
- **Feedback network**
  In contrast, this architecture allows signals to be fed back in both directions by introducing loops in the network [19]. In the process, outputs are fed back into their own inputs [14]. Feedback networks are extremely powerful and can be very complex. Due to their dynamic nature, their state changes continuously until they reach an equilibrium point. They remain at this point until the input changes and a new equilibrium must be found [19]. Feedback networks therefore support short-term memory [14]. Figure 4.3b shows a corresponding architecture as an illustrative model.



(a) Feedforward neural network model [20] (b) Feedback neural network model [21]

Figure 4.3: Artificial Neural Network architectures

The feedforward network is arranged in so-called layers, where each unit only receives input from units in the immediately preceding layer [14]. There are three basic types of layers:

- **Input layer:** This layer consists of the input neurons, which represent the input vector. This input vector is then passed on to the following hidden layers.
- **Hidden layers:** The hidden layers include all layers of neurons that are located between the input and the output layer [18]. The vectors generated are internal and not visible. These layers are occasionally not arranged linearly, but can also have parallel or diagonal connections [10].
- **Output layer:** The output layer represents the output vector of the neural network. The neurons in this layer are not connected to other neurons [18].

Furthermore, a distinction is made between single-layer and multi-layer architectures. Single-layer architectures consist exclusively of an input and an output layer and are sometimes also referred to as perceptrons. In contrast, multi-layer architectures include at least one or more hidden layers that are not connected to the output of the network [14].

With the introduction of additional layers, not only the mapping capacity increases, but also the number of parameters, the required data and the required computing power. A network with many hidden layers is called a Deep Neural Network (DNN). DNNs today can have hundreds of layers and millions of parameters.

These networks are capable of processing a variety of input types, including images or videos represented by vectors, matrices or tensors. During training, the input is transformed step-by-step to produce the desired output. Each neuron receives a vector as input and computes a new vector as output. The challenge is that the resulting hidden layer vectors are not visible, especially during training [10].

For example, let a training set of data $Train = \{(x^1, y^1), ..., (x^n, y^n)\}$, where x is the input and y the desired output vectors. Now we want to make sure that the neural network produces as exactly as possible the output vector that is stored in y^k, where $y^k \in Train$ with the corresponding input vector $x^k \in Train$, where $k \in (1, n)$. Indeed, there exists a loss function $loss = L(\hat{y}, y)$, which describes the loss between the desired output vector and the actual output vector of the neural network. The

actual output vector $\hat{y} = f_2(w_2 h_1)$ results from the product of the resulting vector of the hidden layer's activation function $h_1 = f_1(w_1 x)$ with the weighted vector w_2. h_1 in turn results from the yielding vector of the activation function of the input layer with the input x. Based on the equations just mentioned, the loss can also be represented as follows: $loss = L(f_2(w_2 f_1(w_1 x)), y)$. Figure 4.4 provides a visual representation of the DNN that has just been defined.

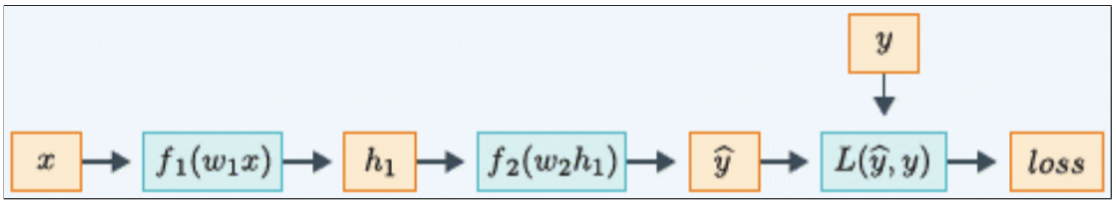


Figure 4.4: Visual representation of the Deep Neural Network, used for explaining the backpropagation method [10]

We also define $u_1 = w_1 x$, which results in $h_1 = f_1(u_1)$, and $u_2 = w_2 h_1$. Now we want to change the weights w_1 and w_2 in such a way that the loss value is minimal. This is done with the backpropagation method:

First, we propagate forward by calculating a predicition for each layer of the network in turn and store all the values, such as x, u_1, h_1, u_2 and $\hat{y}$. This allows us to calculate the error $loss$ at the end. Next, during backpropagation, we calculate the gradient of the error with respect to the weights w_1 and w_2. This results in

$$\frac{\delta L}{\delta w_1} = \frac{\delta L}{\delta \hat{y}} \cdot \frac{\delta \hat{y}}{\delta u_2} \cdot \frac{\delta u_2}{\delta h_1} \cdot \frac{\delta h_1}{\delta u_1} \cdot \frac{\delta u_1}{\delta w_1}$$

and

$$\frac{\delta L}{\delta w_2} = \frac{\delta L}{\delta \hat{y}} \cdot \frac{\delta \hat{y}}{\delta u_2} \cdot \frac{\delta u_2}{\delta w_2}$$

[10].

This gradient tells us how the error would change if the weights were adjusted. Using this gradient and a learning rate α which controls the size of the adjustment steps, we then change the weights, for example by the function:

$$w_{i_{new}} := w_{i_{old}} - \alpha \cdot \frac{\delta L}{\delta w_i}$$

The gradient is subtracted from the old weight to change the weight in the direction that minimises the error [22].
This complete process is now carried out until a certain criterion is reached, e.g. a set number of runs of the dataset (also known as epochs) or reaching a certain level of error [14].

The calculation of the gradient requires at least as much time for each training example as the pure forward propagation. This can lead to considerable computational effort for large data sets. To minimise this effort, batches are used. A batch refers to the number of training examples run before the weights are adjusted [10].

4.4 Convolutional Neural Network

One possible application of feedfoward ANNs is a Convolutional Neural Network (CNN), which supports the processing of grid-based data, including images [22]. The most important application here is object classification, with the aim of automatically identifying different classes of objects in these images. CNNs basically consist of three different types of layers besides the usual input and output layer:

- **Convolution layer**
  In CNNs, images are represented as two-dimensional grids. In order to extract important features from these images, so-called kernels are used that have a certain size, for example 3x3 or 5x5. These kernels contain predefined values.
  The application of these kernels to the input matrix is done step by step. The kernel is first placed in the upper left corner of the input matrix. Then the values of the input matrix are multiplied by the kernel values and the results are summed up. The result of this calculation is stored in a result matrix. An example of the convolution is shown in Figure 4.5.
  Then the kernel is shifted by one position and the process is repeated (see Figure 4.6). This is done until the entire input matrix is covered (see Figure 4.7) [10]. The result matrix, also called feature map, is slightly smaller than the input matrix and contains important features from the image.
  In a CNN, several such kernels are typically used, each capturing different features such as edges or borders [22]. The result matrices of these kernels

are combined into a result tensor. In addition, the result matrices are often transformed by an activation function, such as the ReLU function [10]. This step is important to optimise the extraction and highlighting of features in the images [18].

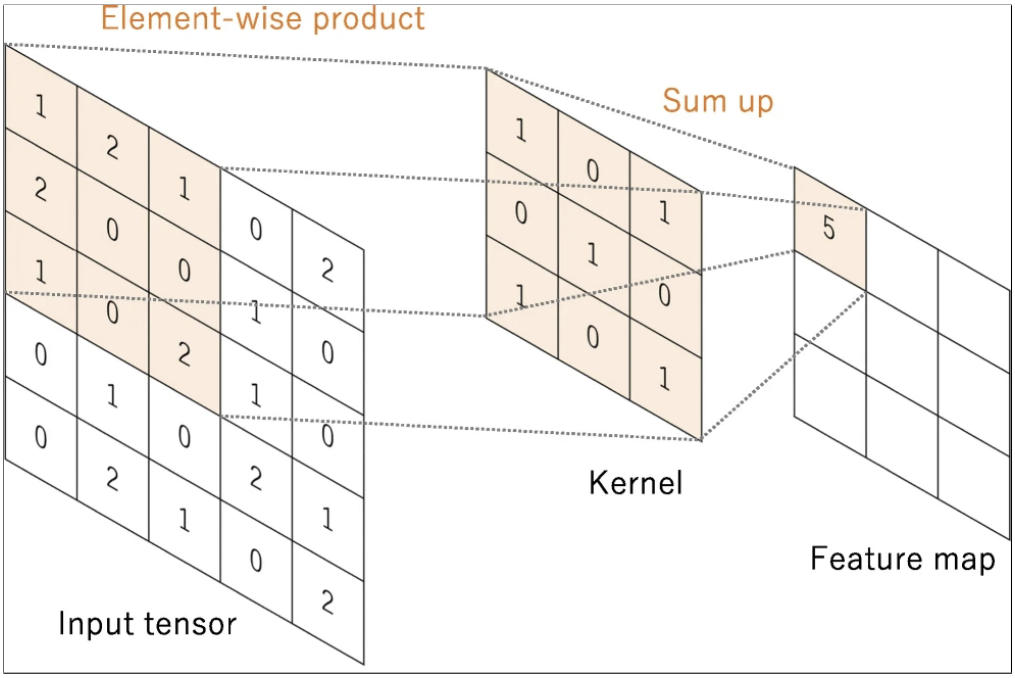


Figure 4.5: First convolution operation on input image with 3x3 kernel [22]

- **Pooling layer**
  The pooling layer is used to reduce the number of feature values, which is particularly advantageous when there are a large number of kernels. Here, several neighbouring features are combined into groups. In this process, the exact position of a feature is often less important than its rough location in the image. Max-pooling is a common method in which the maximum is determined for square sub-areas, for example 2x2 squares, and stored in a result matrix, while the other values are ignored. The selection of the maximum value depends on the respective input image. Figure 4.8 shows an example how the max-pooling process chooses the maximum for each 2x2 area. This approach has three major advantages:

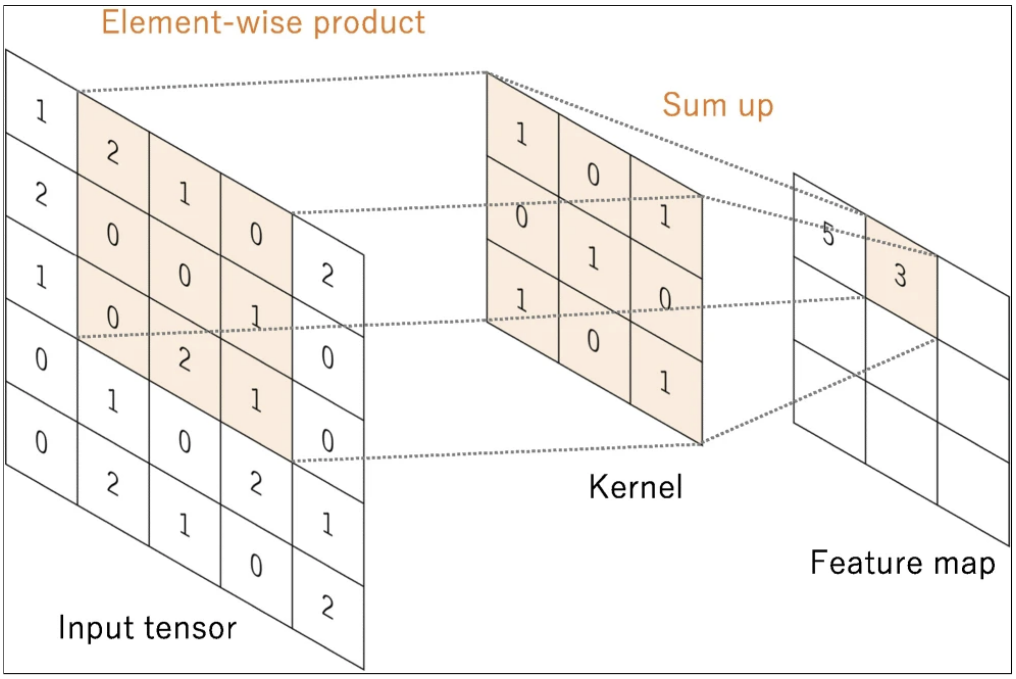


Figure 4.6: Second convolution operation on input image with 3x3 kernel [22]

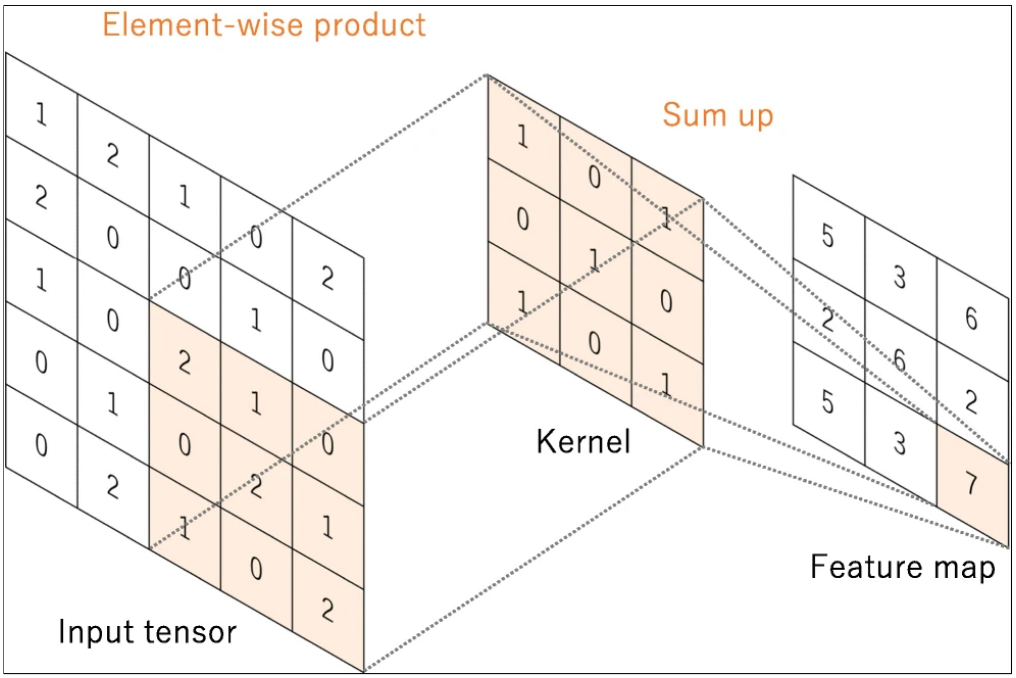


Figure 4.7: Final convolution operation on input image with 3x3 kernel [22]

  - Features detected in CNNs show remarkable invariance to shifts in the input image. This means that the network is able to identify patterns even if they do not occur at exactly the same position in the image.
  - The choice of kernel size in the first two layers has a significant impact on the detected features. Small kernels enable the identification of fine details and smaller patterns, while larger kernels facilitate the detection of larger objects. Applying pooling operations complements this process by further magnifying the detected patterns, which in turn increases the network's ability to identify larger objects in the image.
  - In addition, applying pooling has the advantage of reducing computational power and memory requirements. This is particularly beneficial as it helps to improve the efficiency and speed of the network [10].

  Global average pooling, another pooling algorithm, averages all values per feature map and stores this value. The depth of the feature maps is preserved. This process is usually carried out once before the fully connected layers. layers. The advantages of this are:

  - Reduction in the number of parameters that can be learned
  - Acceptance of inputs with variable size

This sequence of layers can be repeated several times. However, before the output layer, there is (at least) one more

- **Fully connected layer**
  In this layer, the neurons are connected to all the neurons from the previous layer to produce the final output. In classification tasks, especially in the context of images, the output is in the form of probabilities for all classes present. Typically, the last fully connected layer has as many output nodes as there are classes. Each of these layers uses an activation function, such as ReLU [22].

Figure 4.9 shows a CNN consisting of three convolutional and pooling layers and four fully connected layers. This structure enables a classification task. The possible classes on which the CNN is trained include horses, zebras and dogs. The network generates a probability value for each class as output, which enables it to recognise and classify an object of these classes in the image.

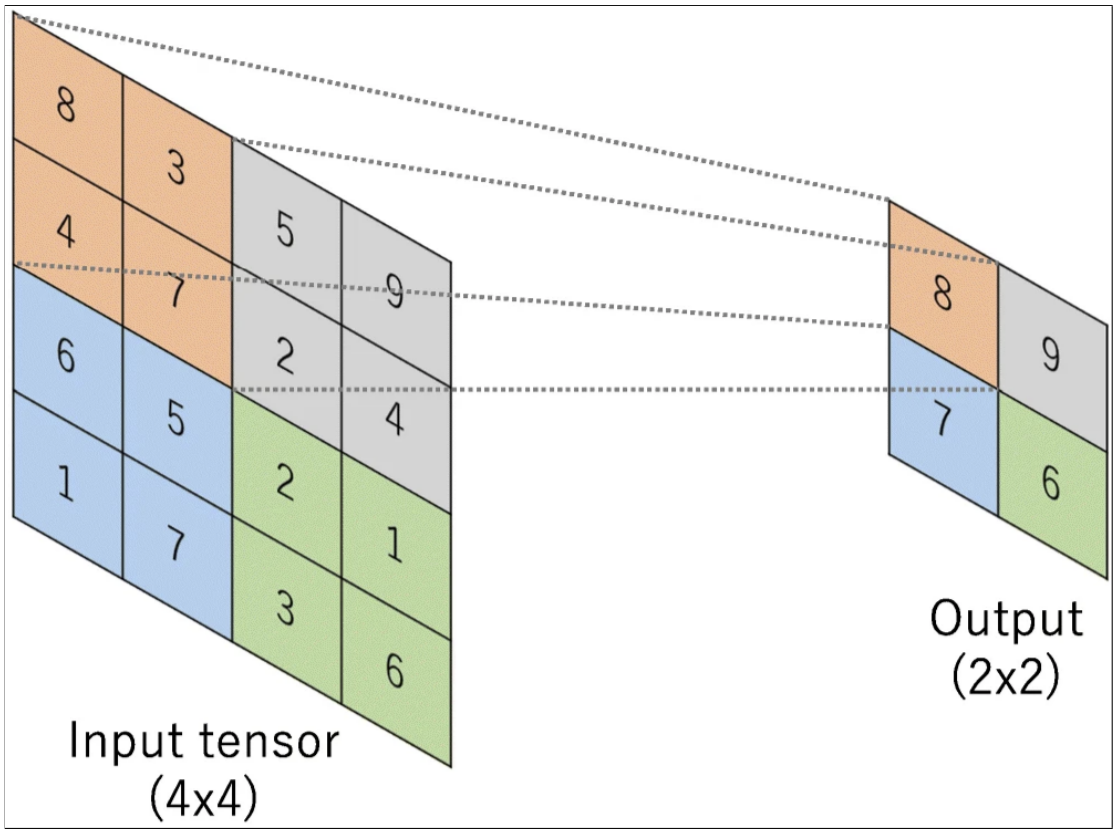


Figure 4.8: Max-pooling example [22]

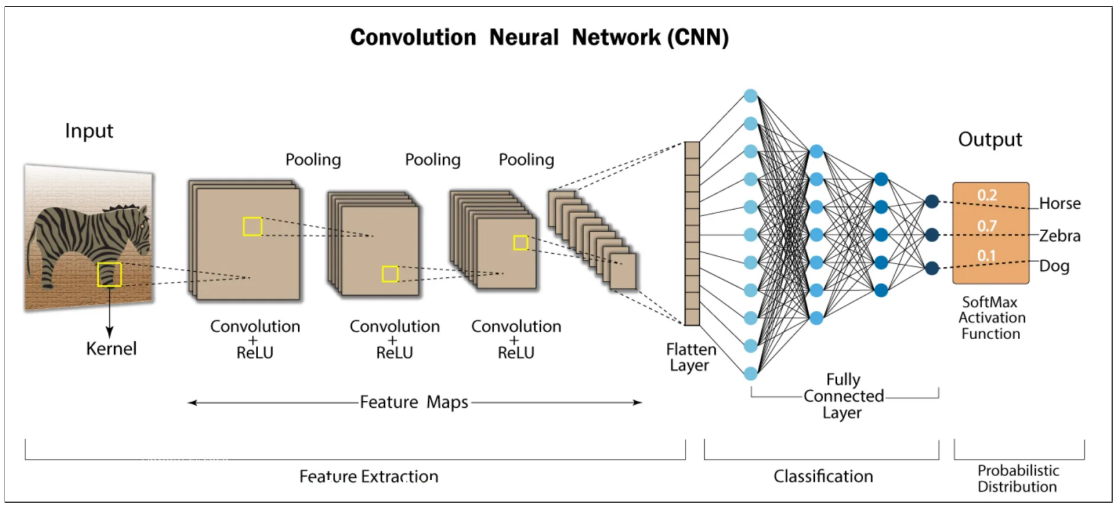


Figure 4.9: CNN example with 3 convolution layers, 3 pooling layers and 4 fully connected layers in order to classify either a horse, zebra or dog in a picture [23]

4.5 Object Detection

The combination of localisation and classification of objects on an image is often referred to as "object detection". In this approach, not only is it detected whether an object of a certain class is on the image, but also at which position it is located [24]. This enables the detection and localisation of multiple objects within an image, even if they have different sizes [25]. So-called "bounding boxes" are used to indicate the position and boundary of the recognised objects [10]. A distinction is made between two types of object detection models:

- **Two Stage Detector (TSD)**
  TSD models are two-stage detectors divided into two stages, often with a Region of Interest (ROI) pooling layer between them. In the first stage, also known as the Region Proposal Network (RPN), potential bounding boxes for objects are proposed. In the second stage, using the ROI pooling process, features are extracted from the candidate boxes to perform classification and bounding box matching. TSDs are characterised by high accuracy in locating and detecting objects.
- **Single Stage Detector (SSD)**
  Unlike TSD models, SSDs propose bounding boxes directly from input images and do not require a separate RPN layer. This makes them more time-efficient and enables their use in real-time object detection applications.

In both architectures, the output for each detected object is usually in the form of the associated class (cls) and the associated position (loc). The basic structures for TSDs (a) and SSDs (b) are shown in Figure 4.10. The term "backbone" refers to the use of CNNs for feature extraction and enabling classification. When adapting to specific requirements, such as accuracy or efficiency, the replacement of the final fully connected layers with specially designed layers often occurs. This process occasionally requires a trade-off between speed and accuracy [26].

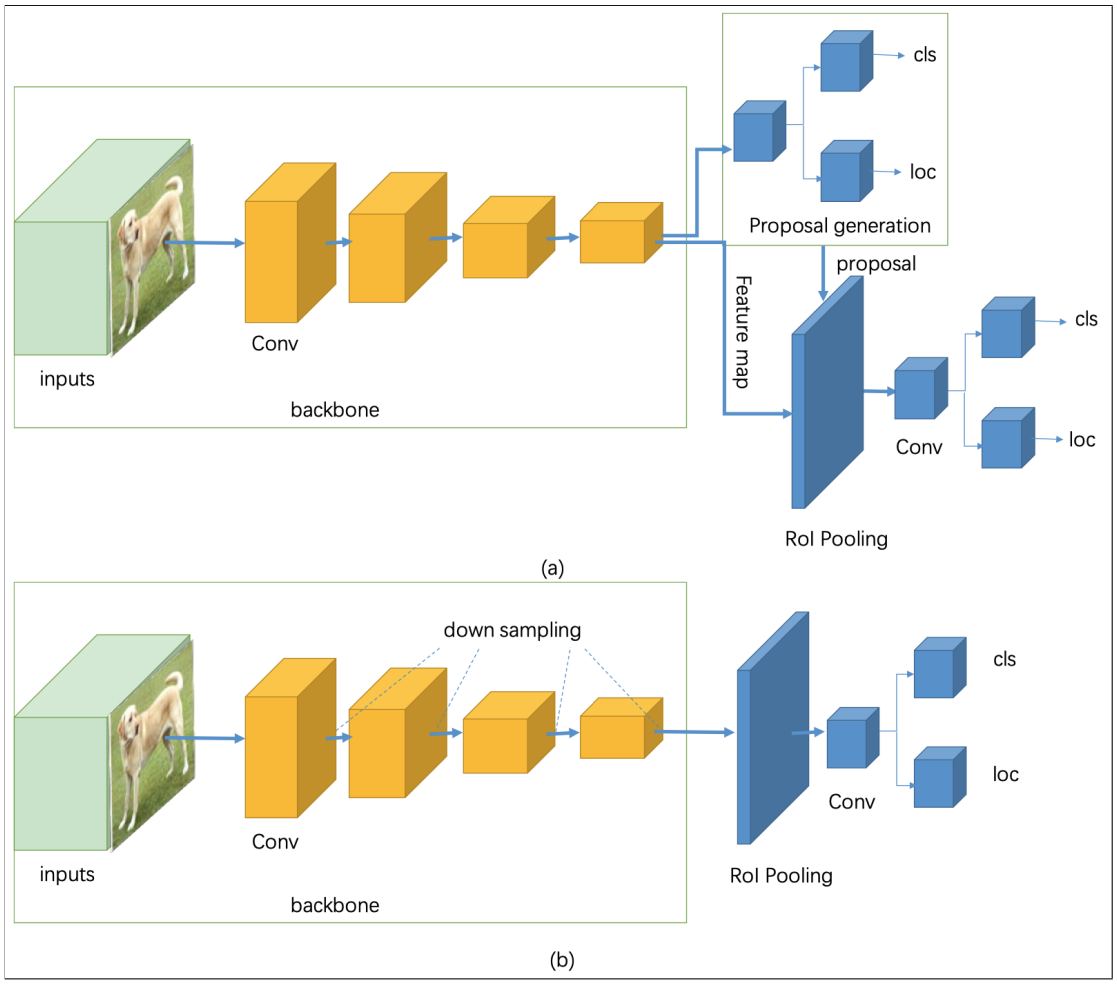


Figure 4.10: Basic architectures of TSDs (a) and SSDs (b) [26]

Evaluating the performance of an object detection model is crucial to assessing its accuracy and optimising it if necessary. A common measure for assessing accuracy is the so-called mean Average Precision (mAP). The mAP is calculated using the following formula:

$$mAP = \frac{1}{N} \sum_{i=0}^{N} AP_i$$

N represents the number of object classes, and AP_i represents the average precision for each individual class, which can be calculated as follows:

1. **Intersection over Union (IoU)**
   During the training process, IoU is used to measure the overlap of the detector's identified bounding box ($bbox$) with the actual ground truth box (gt). This is done by dividing the area in which the detected and actual bounding boxes overlap by the total area of the detected and actual bounding boxes. The calculation formula is:

   $$IoU = \frac{bbox \cap gt}{bbox \cup gt}$$

   A higher IoU value indicates a more accurate localisation.

2. **Setting thresholds for class assignment**
   A threshold value is then defined that determines from which overlap a detected object in the image is considered acknowledged. To find out the mAP@0.5:0.95, the image is scanned for several thresholds, starting at 0.5 and ending at 0.95 with an increment of 0.05. Alternatively, there is the mAP@0.5, which indicates the accuracy of the model exclusively for the threshold value 0.5. However, this measure only provides a general assessment of the model accuracy compared to the detailed mAP@0.5:0.95.

3. **Calculation of the Confusion Matrix**
   The Confusion Matrix can now be created for each threshold value. It includes the following fields:

   - **True Positives (TP):** This category includes all bounding boxes whose IoU value is above the corresponding threshold set. True Positives are the correctly recognised objects.
   - **False Positives (FP):** False Positives are the bounding boxes whose IoU value is below the corresponding threshold set. They represent the incorrect bounding boxes.
   - **True Negatives (TN):** This category includes those areas where neither the model nor the ground truth bounding boxes detected objects. In other words, these are correctly detected absences of objects.
   - **False Negatives (FN):** If the model does not recognise an object and therefore does not draw a bounding box even though it is present, these cases are called False Negatives.

4. **Calculation of precision and recall**
   Using the values from the Confusion Matrix, precision (P) and recall (R) are calculated by:
   $$P = \frac{TP}{TP + FP}$$
   and
   $$R = \frac{TP}{TP + FN}$$
   Precision indicates how reliable the positive predictions of the model are, while recall indicates how many of the actual objects were correctly detected.

5. **Calculation of the area under the PR curve**
   Finally, for each selected threshold value, the corresponding precision and recall values are plotted in a coordinate system. The area under the resulting PR curve is calculated with:
   $$AP = \int_0^1 P(R)\, dR$$
   This value represents the Average Precision (AP) for the corresponding class which can then be used to calculate the mAP [27].

If several bounding boxes are drawn for one object, Non-Maximum Suppression (NMS) is used. The main goal of NMS is to ensure that only one bounding box is drawn for each detected object to avoid overlaps and duplicate detections. This is a crucial step in object detection to clean up the results and ensure that only the most appropriate bounding box is kept for each detected object [28].

4.6 Conclusion

This chapter provided a comprehensive introduction to Artificial Intelligence and Machine Learning. It covered definitions, historical developments and the different types of Machine Learning. Deep Learning, especially the architecture of Artificial Neural Networks, was covered in detail. Within this topic, important concepts such as feedforward and feedback networks, activation functions and backpropagation were introduced. One example was the Convolutional Neural Network for image processing. In object recognition, differences between Single Stage Detector and Two Stage Detector were explained, as well as the calculation of mAP for model evaluation.
This understanding will be useful in the upcoming chapter when it comes to selecting and implementing a suitable AI solution for our iOS app for organic and non-organic waste detection.

5 Implementation

This chapter is dedicated to the implementation of the idea described in Section 2.3 and the resulting functional and non-functional requirements (see Chapter 3). We will first present the implementation of AI for object recognition. Based on a choice made between the two leading architectures, we will then train the model using the dataset we have compiled ourselves. We will then evaluate the performance of the model based on the results after training. Finally, the trained model will be exported for use on Apple devices.
The second half of this chapter focuses on the implementation of the actual iOS app. After presenting some technical features and the general architecture, we will go into detail about the implementation of the functional requirements. Finally, we will evaluate the fulfilment of the non-functional requirements.

5.1 AI Model

In this section we look at the implementation of AI to enable object recognition in the app. We will first make a choice between the two architectures, YOLO and SSMD, based on related work. Then we will assemble a dataset that will be used for subsequent training of the model. The results of this training will serve as the basis for evaluating the performance of the model. Finally, we will describe the steps necessary to successfully integrate the model into the app.

5.1.1 Selection

For the selection of the AI model for implementing our app idea, we focus on Single Stage Detectors. This is due to FR13, which requires real-time object detection.

According to Licheng Jiao et al., You Only Look Once (YOLO) and Single Stage MultiBox Detector (own abbreviation SSD, we use SSMD due to confusion with Single Stage Detector (SSD)) are among the most representative examples of SSDs [26]. We will first give an insight into their functionality. Afterwards, we will look at related work in order to make an equipped decision about which one is best suited for the implementation of our app.

YOLO

In YOLO, the initial convolutional layers of the neural network are used to extract image features, while the last layers of the CNN predict the probabilities and coordinates for detecting objects. The image is divided into a grid with dimensions SxS. The grid cell in whose area the centre of an object occurs is responsible for the detection of the object.

Up to B bounding boxes are provided in each grid cell to detect potential objects. A bounding box is characterised by the values b_x and b_y, which represent the coordinates of the centre of the box relative to the respective grid cell, and the values b_w and b_h, which indicate the width and height of the box relative to the total size of the input image.

A confidence value p_c is calculated for each bounding box. Essentially, it is a probability that indicates whether or not an object exists in this grid cell. If the model believes that there is no object, a low value (0) is given.

In addition to the confidence values, each grid cell for each bounding box gives the probabilities for C classes, where C represents the number of available classes. These probabilities indicate the likelihood with which the detected objects in this grid cell belong to the different classes.

Thus, per grid cell there are B outputs y, which have the following structure (when

having $C = n$ classes):

$$y = \begin{bmatrix} b_x \\ b_y \\ b_h \\ b_w \\ p_c \\ c_1 \\ \dots \\ c_n \end{bmatrix}$$

Altogether, this results in a tensor with the size: $S \times S \times B \cdot (5 + C)$ [29].

SSMD

The fundamental difference between YOLO and SSMD is their approach to object recognition. While YOLO is limited to a single scale of feature maps, SSMD works on multiple scales of feature maps to enable the recognition of objects of different sizes.

In SSMD, additional convolutional layers are added at the end of the base network. These layers are progressively smaller and allow predictions for the detection of objects of different sizes. For each of these convolutional layers, special convolutional filters are used to make predictions for object detection. These predictions include both class affiliation probabilities and bounding box information.

In addition, "standard boxes" are used in conjunction with the feature maps. Standard boxes are predefined bounding box shapes that are placed on the different layers of the feature maps. These boxes are used to represent potential positions for detected objects and form an important basis for the predictions of SSMD. For each of the k standard boxes at a specific location on the feature maps, the following information is calculated:

- **Class scores:** these scores represent the probability that a detected box belongs to a particular class. There are c class scores predicted for each of the k standard boxes.
- **Bounding Box Shifts:** These shifts indicate how the standard box must be adjusted to better match the detected bounding box. There are 4 predicted

shifts for each of the k standard boxes, relative to the original shape of the standard box.

The output of this process is a tensor with $(c + 4)kmn$ output values for a feature map of size $m \times n$ with p channels [30].

Final Choice based on Related Work

Previous research, notably the work of Deep Patel et al., investigated different waste detection architectures, including YOLOv5M and SSMD. Evaluation using the mAP@0.5 criterion showed that YOLO performed significantly better, with a mAP@0.5 of 0.613 compared to SSMD, which scored only 0.511 [31].

A comparable study was conducted by Liu Bohong and Wang Xinpeng, evaluating different architectures for waste detection. Here, YOLOv4 was again shown to be superior to SSMD, with a mAP@0.5 of 64.33 compared to 63.56 for SSMD [32].

Both Deep Patel et al. and Liu Bohong and Wang Xinpeng emphasised in their work that YOLO is suitable for real-time object detection [31, 32]. Based on this result and the better mAP values, we also decided to use the YOLO architecture.

The latest version of YOLO, YOLOv7, has a shorter inference time and offers improved AP compared to previous versions [33]. In addition, models are available in different sizes, depending on the requirements of each device. This includes the `tiny` model, which is particularly suitable for edge devices and mobile phones.

YOLOv7 is freely available in a public GitHub repository and is provided free of charge. Training tutorials are also available, and an export script allows the trained model to be converted into formats such as CoreML that can be used on Apple devices [34].

Now that the decision to use the YOLO architecture has been made, we will detail our approach to implementing and evaluating the solution. In the coming sections, we will go into detail about data collection, training, the results obtained and export.

5.1.2 Data Collection and Preparation

Before the usage of the object detector can be considered, it is necessary to train it appropriately to be able to recognise the desired objects correctly. In Section 2.3, we decided on the identification of a total of 20 object classes. These classes include the ten most common organic waste products as well as ten common impurities in organic waste. In defining the organic waste object classes, we followed the guidelines of the Disposal Services Ulm [35] and the waste management company Alb-Donau-Kreis [36]. For the object classes of non-organic waste, we looked at the data collection of SIEBW to see which typical foreign substances turned up during the investigation [5]. Finally, we define the following object classes:

Organic waste object classes	**Non-organic waste object classes**
• apple	• battery
• bone	• can
• egg carton	• ceramic
• egg shell	• cigarette
• feather	• face mask[2]
• foliage	• glass
• orange peel	• pill
• paper towel	• plastic bag
• soil	• plastic cup
• sunflower	• stone

Table 5.1: Object classes that shall be detected with the AI

James Skelton recommends organising around 2,000 images with labels per object class for optimal results when using YOLOv7 [37]. The online database Roboflow[3], also known as Roboflow Universe, was used to collect this extensive set of images.

[2] self defined

[3] Roboflow Website: `https://universe.roboflow.com`

On Roboflow, ready-to-use datasets are available that have already been annotated with bounding boxes and the corresponding classes. These datasets can be exported in various formats, including YOLOv7, or integrated into individual datasets. For integration, there is the flexibility to use the entire dataset or selectively chosen images, for example images of a certain object class. A significant strength of this platform is the possibility of data augmentation. This technique allows the dataset to be augmented by modifying existing images through rotation, flipping or targeted quality degradation to train the model more efficiently. In addition, Roboflow offers the possibility to create your own datasets. Users can upload desired images and independently annotate objects with bounding boxes and classes using an annotation tool. When splitting data sets for export, it is also possible to specify how the data is split and whether the images should be scaled to a specific format. In our case, for each object class, it was decided to use 70% of the images for training, 20% for validation and 10% for testing, scaling the images to a 640x640 format. This procedure is justified because the division ensures a balance between training data, validation data and test data and thus enables a reliable evaluation of the model. The scaling is done because YOLOv7 is optimised for square images.

The training data is provided to the model to enable learning patterns and features in the data. The validation data is used to monitor the performance and progress of the model during training without being trained on the data. They are used exclusively for evaluation. Finally, the test data is used to evaluate the performance of the trained model after training and validation are complete. This test data presents unknown data to the model to ensure that it is able to generalise patterns and has not simply memorised the training data.

Table 5.2 provides an overview of all used data sets, as well as augmentation measures and object class affiliation. It also includes the number of images and annotations.

Object Class	#Images	#Annotations	Augmented?	Dataset Source(s)
apple	2,216	2,257	x	[38, 39, 40]
bone	1,003	1,214	x	[41, 42]
egg carton	634	6,037		[43]
egg shell	891	3,106	x	[44, 45, 46]
feather	2,159	3,621		[47, 48]
foliage	1,692	2,863	x	[49]
orange peel	1,624	2,658	x	[50]
paper towel	1,592	2,285	x	[51] & own pictures
soil	1,973	1,973	x	[52, 53]
sunflower	1,123	3,274		[54]
battery	1,528	3,080	x	[55]
can	1,395	2,027	x	[56]
ceramic	1,444	2,058	x	[57]
cigarette	2,120	3,123		[58]
face mask	1,830	1,995	x	[59, 60]
glass	1,732	2,204	x	[61]
pill	2,827	2,982	x	[62]
plastic bag	2,909	2,920	x	[63]
plastic cup	1,018	2,567		[64]
stone	2,246	2,318	x	[65, 66]

Table 5.2: Composition of the dataset

Table 5.2 also shows that in some cases 2,000 images may not always have been reached, even using augmentation techniques. Nevertheless, the classes in question have a significant number of annotations. This is due to the fact that multiple bounding boxes may exist on individual images.

Roboflow provides the exported data set in the required structure of YOLOv7. A `data.yaml` file is created that specifies the paths to the respective image folders for training, validation and testing. In addition, the number of object classes in the data set is specified in this file under `nc`. The class names are listed in a string array with the name `names`. Code Snippet 5.1 shows the `data.yaml` file for the generated data set.

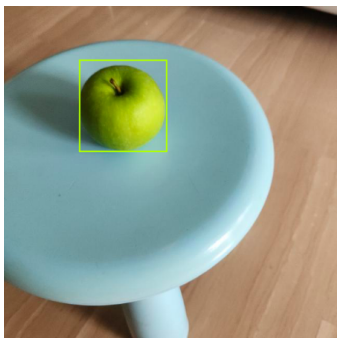
```
train: ../dataset/train/images
val: ../dataset/valid/images
test: ../dataset/test/images

nc: 20
names: ['feather', 'flower', 'egg', 'kitchen paper',
        'apple', 'foliage', 'soil', 'egg carton',
        'orange', 'bone', 'cigarette', 'pill',
        'plastic cup', 'plastic bag', 'glass', 'face
        mask', 'ceramic', 'can', 'battery', 'stone']
```

Code Snippet 5.1: data.yaml file

The labels folder, which is on the same level as the image folders for training, validation and testing, contains the annotations for the corresponding images. The annotation files have exactly the same name as the corresponding image files, but are saved in .txt format. These files contain all the bounding boxes previously drawn within the respective image.

In these text files, the number corresponding to the object class is specified first. The model is based on the `names` array from the `data.yaml` file, whereby the array starts at index 0.
As an example: Assuming we have an image called `apple.png` (see Figure 5.1) that has an apple on it and we have marked it with a bounding box, the number of the object class is noted in the `apple.txt` file (which is number 4 for the apple according to the `data.yaml` file). Then the position of the object is specified in a certain format. This format includes the x and y coordinates of the centre of the object as well as the width and height of the object. These values are always in the range of 0 to 1, as YOLO works with relative values. Code Snippet 5.2 shows the corresponding `apple.txt` file for the `apple.png` file in figure 5.1.

With the help of the assembled data set, the model can now be trained to identify the desired objects.



```
4 0.3578125 0.30078125 0.2609375 0.275
```

Code Snippet 5.2: apple.txt file

Figure 5.1: apple.png file

5.1.3 Training Process

The YOLOv7 repository already provides a Python script that can be used to train own models on user-defined data sets [34]. The corresponding command for starting the training is shown in Code Snippet 5.3. In the following, we explain the meaning of the individual parameters in detail:

- `!python train.py` is the command to run the `train.py` script.
- `--img` sets the input size of the images for the training. Since our images were previously scaled to 640x640 pixels, we use these values.
- `--batch-size` defines how many images are processed simultaneously in each step of the training process. In this case we use a batch size of 16 images per training step.
- `--epochs` specifies how often the model is trained over the entire dataset. In our case, we chose 150 runs to adequately train the model.
- `--data` points to the path to the configuration file for the training data. Here we use the `data.yaml` file mentioned earlier (see Code Snippet 5.1).
- `--weights` specifies the path to the previous weights. In this particular case, we use the default weights that YOLOv7 provides in the repository [34].
- `--cfg` points to the configuration file for the YOLOv7 model. This file contains information about the architecture of the model. We decide to use the smallest model, `yolov7-tiny`, which is specially optimised for use on mobile devices.

- `--name` can be used to name the trained model. In this case we have given our model the name "yolov7".
- `--cache-images` causes the images to be cached during training to speed up access to them. This contributes to the efficiency of the training process.

```
!python train.py
    --img 640 640
    --batch-size 16
    --epochs 150
    --data /content/dataset/data.yaml
    --weights /content/yolov7/train.py
    --cfg /content/yolov7/cfg/training/yolov7-tiny.yaml
    --name yolov7
    --cache-images
```

Code Snippet 5.3: Command for starting the training of the model

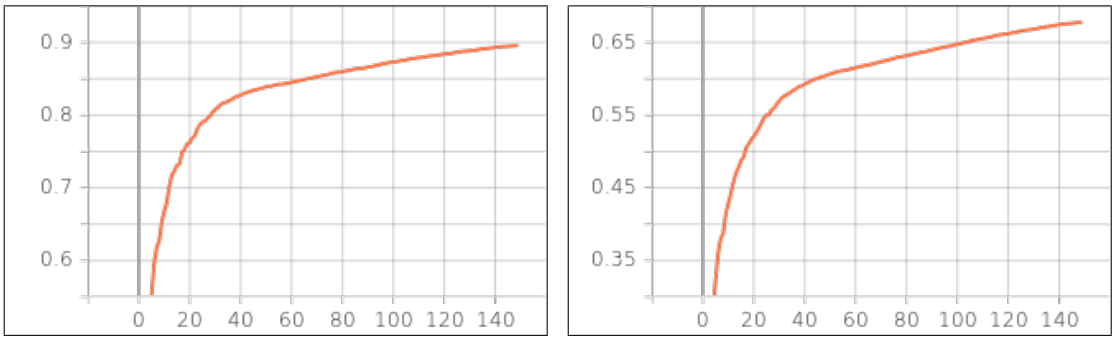
ANN training often requires the use of Graphical Processing Units (GPUs) due to their ability to provide a high number of computational cores [10]. Google Colab[4] provides an online platform that allows Python notebooks to run in a cloud-based environment. This platform supports the provision of powerful GPUs, which in particular accelerates the training of AI models. Furthermore, Google Colab provides the ability to store data in Google Drive, which allows for easy data management and storage [67]. For these reasons, and the fact that our existing hardware was not sufficient to conduct the training, we decided to conduct the training within such a Colab environment.

5.1.4 Results

During the training process, a file called `events.out.tfevents` is created. This file contains records of various metrics and statistics that allow the progress of the model to be tracked over time. Using TensorBoard, the collected information can be visualised, providing detailed insights. These visualisations prove to be highly useful in selecting the most appropriate model from the variants generated. During

[4]Google Colab Website: `https://colab.research.google.com`

the training process, several models are created with different weights, including those created at different times, at the beginning and at the end of the training. Among them is also the model that has the best weights according to the results during the training.
In addition to the file, graphics for the Confusion Matrix, P-, R- and F1-curve are created. We will now take a look at some of these data to evaluate the performance of our trained model.



(a) mAP@0.5 plot during training (b) mAP@0.5:0.95 plot during training

Figure 5.2: Development of the mAP@0.5 and mAP@0.5:0.95 for the trained YOLOv7 model, visualised with TensorBoard. The X-axes represent the progression over the epochs, while the Y-axes represents the associated values.

- **mAP@0.5** (see Figure 5.2a)
  In the context of mAP evaluation using a single threshold, our model achieves an impressive value of 0.8955 after 150 epochs. After around 40 epochs only, the mAP@0.5 was already over 0.8. This indicates that our model learns very efficiently and already shows remarkably high performance at a comparatively early stage of the training process.
- **mAP@0.5:0.95** (see Figure 5.2b)
  Looking at the mAP including a wider range of thresholds, from 0.5 to 0.95 in steps of 0.05, we see a maximum value of 0.6781 after 150 epochs. This value is naturally somewhat lower than that of the mAP@0.5, since the mAP@0.5:0.95 has stricter precision requirements for detection. Likewise, it can be observed that a mAP value above 0.5 is already reached after 40 epochs. This suggests that our model is able to maintain acceptable accuracy over a wide range of thresholds.

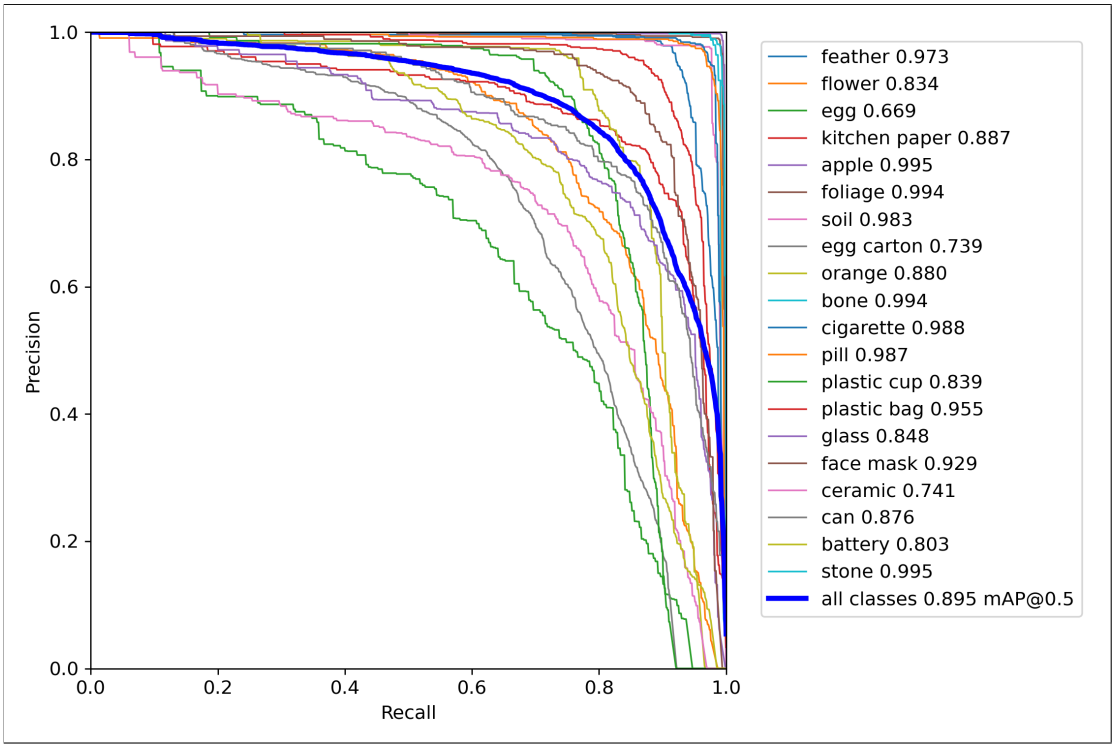


Figure 5.3: Development of the PR curve for the trained YOLOv7 model

- **PR curve** (see Figure 5.3)
  According to the findings of Jiaju Miao and Wei Zhu, an ideal model is characterised by having a precision of 1 for all recall values. In general, it is assumed that the closer the points of its PR curve are to this ideal line, the better the performance of the model [68]. The PR curve of our model shows that it performs very well for some object classes and comes close to the ideal curve. Most object classes also perform well. Unfortunately, for the object classes "egg shell", "ceramic" and "egg carton", a lower precision can be observed compared to the ideal curve. This illustrates that the model is able to identify some object classes with high accuracy, while there is room for improvement for other classes.

Overall, it can be concluded that the model can predict most object classes with high accuracy and performs well across different thresholds. Nevertheless, there is room for improvement, especially for certain object classes. One possible factor for this room for improvement could be the limited number of available training images for certain object classes, especially with regard to the classes "egg shell" and "egg

carton", which have below 1,000 images in the dataset (see also Table 5.2).

Since both mAPs have the highest value in the last epoch and according to NR1 a high accuracy is desired, we use the model with the weights from the final epoch, which is called `last.pt`.

5.1.5 Preparations for embedding in the iOS app

Besides the possibility to train models for individual purposes, the YOLOv7 repository also offers a script `export.py` to convert finished models into various formats, including ONNX, TensorRT and CoreML. The latter format is intended specifically for use on Apple devices. The conversion is done using the command presented in Code Snippet 5.4.

```
!python export.py --weights best.pt
```

Code Snippet 5.4: Export command

Although `export.py` successfully converts our `last.pt` model into the desired `last.mlmodel` format for use on Apple devices, it should be noted that the exported model does not include post-processing measures such as NMS and the detection layer. NMS is responsible for reducing redundant bounding boxes, as explained in Section 4.5. The detection layer processes the output of the neural network and generates the final detected objects in a format that contains bounding box coordinates in absolute values and the corresponding confidence values. In addition, it allows setting a confidence threshold to remove bounding boxes whose confidence values are below this threshold. To integrate these post-processing measures, a re-conversion of the exported `last.mlmodel` is done using the script `YOLOv7CoreMLConverter.py` from the YOLOv7-CoreML-Converter repository[5].

[5]YOLOv7-CoreML-Converter Repository:
`https://github.com/junmcenroe/YOLOv7-CoreML-Converter`

5.2 iOS Application

Finally, this section deals with the development of the iOS app in which the previously developed and trained neural network is integrated. We start with a brief introduction to the development tool used. Then we provide a general overview of the class structure of the app, followed by a detailed presentation of the implementation of the functional requirements. Finally, we check whether our finished app also fulfils the non-functional requirements.

5.2.1 Technical Features

The application was developed using Xcode, the Integrated Development Environment (IDE) developed by Apple. For the implementation, we used version 14.2 of this IDE and the programming language Swift. We designed the User Interface (UI) with the UI framework SwiftUI. The use of the Xcode simulator proved to be extremely helpful, as it made it possible to virtually test the developed app on various Apple devices and identify errors at an early stage. This greatly contributed to the development of an stable and performant app. The finished app is compatible with devices that run at least iOS 16.2 (or newer operating systems).

5.2.2 Architecture

The app consists of three main groups: Object Detection, Camera Interface and Graphical User Interface (GUI). In this structure, the DataModel class functions additionally as a controller and takes over the interface management between the camera, the object recognition, especially the information on recognised objects, and the UI. Another class that does not belong to any group is the coordinator. The corresponding class diagram is shown in Figure 5.4.
In the following, we will describe the main functionalities of the individual classes in order to then go into detail about the implementation of the functional requirements from Chapter 3.

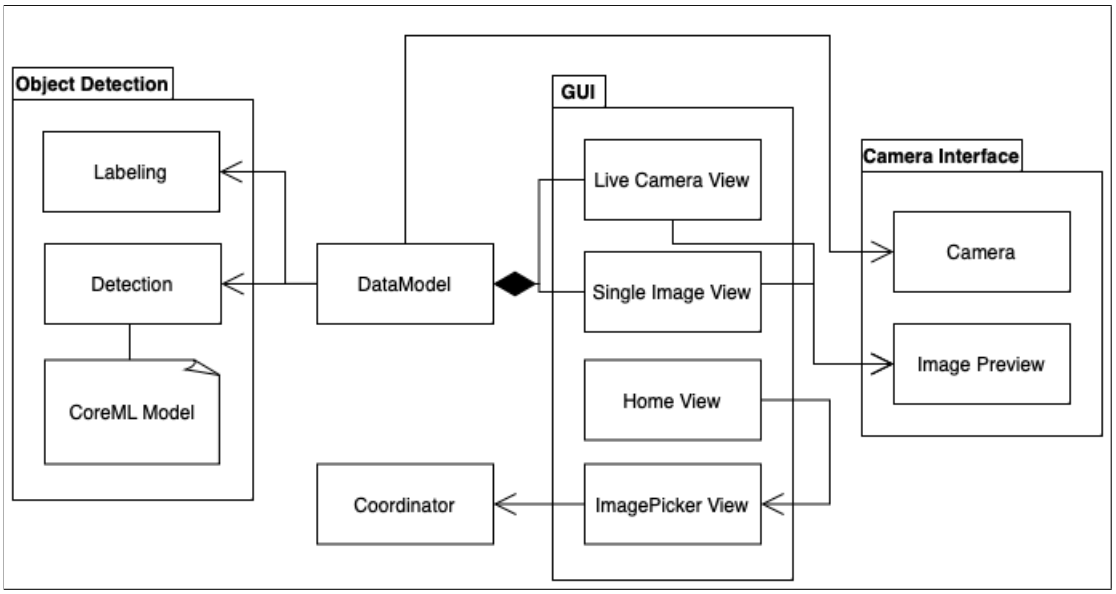


Figure 5.4: Class Diagram, made with draw.io[6]

Object Detection Group

Labeling Class

- draws bounding boxes and displays confidence values
- performs bounding box colour matching

Detection Class

- performs the actual object detection and processing
- converts normalised coordinates (values between 0 and 1) into absolute image coordinates
- stores detected objects
- filters the results for the corresponding use case

CoreML Model

- trained neural network in CoreML format

[6]Draw.io Website: `https://app.diagrams.net`

DataModel Class

- processing of camera and image data
- coordinates the object recognition process
- generates information texts

GUI Group

Live Camera View Class

- displays the live view
- displays detected objects together with information

Single Image View Class

- displays selected photo for analysis
- displays detected objects along with information

Home View Class

- Home screen of the application
- provides choices for different modes and type of input for analysis
- contains information on recognised objects
- displays agreement form to the terms of use, disclaimer and privacy policy when the app is launched for the first time

Image Picker View Class

- allows selection of images from the gallery or taking photos with the camera

Camera Interface Group

Camera Class

- controls the camera, including switching between front and rear cameras
- configures camera output
- allows taking photos
- manages device orientation

Image Preview Class

- used to display pictures or camera views

Coordinator Class

- enables communication and handling of image selection/picture taking actions between the app's UI and the device's camera or photo library

5.2.3 Realisation of the Functional Requirements

When explaining the implementation of the individual functional requirements, we refer as far as possible to a common user guidance within the app.

First app launch: Obtain consent to Privacy Policy, Disclaimer & Terms of Use

Meets requirement(s)	FR19, FR20, FR21
Description	When the app is launched for the first time, a dialogue window is displayed containing the Privacy Policy, Disclaimer and Terms of Use. Each of this information is listed in a fold-out section. To use the app, the user must activate all three switches. Only after activating all three switches the "Done" button is enabled and clickable (see Figure A.1) to close the dialogue window and get to the actual home screen.

Technique	When the app is started, it is checked whether the app is being executed for the first time. In this case, a Boolean system variable `agreedToTAC` is created and set to `false`. This variable is permanently stored in the app as long as it is not deleted. The corresponding dialogue box with the information is only displayed if `agreedToTAC` has the value `false`. After the user has activated all `Switches` and clicked the "Done" `Button`, `agreedToTAC` is set to `true`. This means that the dialogue window is no longer displayed each time the app is started and the user is directly shown the home screen.
Implemented in	Home View

Selection of the two operating modes: Exclusive foreign matter detection and organic waste and non-organic waste detection

Meets requirement(s)	FR5, FR6
Description	Two options are available to the user on the home screen to select the desired operating mode. The option for exclusive foreign matter detection is activated by the button "Check organic waste", while the option "Classify objects" activates the mode for identifying both organic and non-organic waste objects. Figure A.2a shows the home screen of the app.
Technique	Both buttons are realised using a `ZStack`, which enables the layered arrangement of UI elements. A rounded `Rectangle` is placed in the lowest layer, followed by a `Text` with the corresponding name. This implementation was specifically chosen as an alternative to a conventional `Button`, as clicking on this button opens a `Menu` for selection. In this `Menu`, the user can choose between various options, including selecting a photo from the gallery, taking a new photo or using the live view. Activating the `Menu` requires a `label` that eventually represents the `ZStack`.
Implemented in	Home View

Overview of recognisable objects

Meets requirement(s)	FR18
Description	A button called "Object Overview" is available on the home screen (see Figure A.2a). Clicking this button opens a sheet that displays a list of recognisable objects based on Table 5.1. The sheet can be seen in Figure A.2b.
Technique	The `Sheet` is displayed when `showObjects = true`. By default, this value is `false` and is only set to `true` when the corresponding `Button` on the start screen is clicked. Clicking "Done" on the `Sheet` then sets `showObjects` back to `false` and the `Sheet` closes.
Implemented in	Home View

Help function for the different modes

Meets requirement(s)	FR17
Description	Next to the buttons for selecting the modes, as shown in Figure A.2a, there is a smaller button with a question mark symbol. Clicking on this button opens a sheet with information on the operation of the respective mode. The functions are briefly explained and illustrated by pictures (see Figure A.3).
Technique	These buttons were also implemented using a `ZStack` for each. They were embedded in an `HStack` together with the mode selection buttons in order to arrange the mode and help buttons next to each other. The corresponding `Sheet` is only displayed if the value `showInfoCheck` for the "Check Organic Waste" mode or `showInfoClassify` for the "Classify Objects" mode is set to `true`. By default, both values are set to `false` and are changed to `true` by clicking the respective question mark button. They can be set back to `false` by clicking the "Done" `Button`, which will close the `Sheet`.
Implemented in	Home View

Choice of capture option

Meets requirement(s)	FR1
Description	Clicking one of the two mode selection buttons opens a menu listing the following three options: "Take Photo", "Choose image from gallery" and "Live Preview" (see Figure A.4).
Technique	The `ZStack` that acts as a mode selection button is actually a `label`. Clicking on the `label` opens the `Menu` with the three options, displayed as `Button` each.
Implemented in	Home View

Take new photo with camera

Meets requirement(s)	FR2
Description	Clicking on the option to take a photo within the menu for selecting the recording option opens the camera. A photo can then be taken with the camera. After taking the photo, the image can be cropped or retaken if necessary. The user can then confirm the image as the one to be analysed or return to the home view at any time.
Technique	When the `Button` "Check organic waste" is clicked, the following actions are performed: • The variable `sourceType` is set to `.camera`. • The variable `showPhotosPicker` is set to `true`. • The variable `useCase` is set to `1`. As `showPhotosPicker` is set to `true`, a `Sheet` containing the `Image Picker View` is displayed. This `View` allows the management of the camera and the image gallery. Based on the setting `sourceType = .camera`, a `UIImage PickerController` is created that opens the camera to take a photo. The `Coordinator` is responsible for taking and processing the photo. When an image is selected for analysis, the `Coordinator` is used to route this image back to the `Home View`. Here the image is prepared for later analysis. Then `navigateTo` is set to `2` to allow navigation to the `Single Image View`. In this `View` the selected image is displayed. The `useCase` is also passed to the `Single Image View` to indicate the mode "Check organic waste".

	The variable `isActiveBio` is set to `true`. This ensures that the `Single Image View` is not opened until the selected image is fully loaded for analysis. If "Classify objects" mode is selected instead, `useCase` is set to `2` and instead of `isActiveBio`, `isActiveClassify` is set to `true`. The `useCase` ensures that, depending on the selected mode, either only non-organic waste objects (1) or both organic and non-organic waste objects (2) are detected and labelled. When returning to the `Home View`, the selected image is set to `nil`, `isActiveBio` and `isActiveClassify = false`, `useCase = -1` and `navigateTo = -1` are set in order to be able to exclude a misbehaviour of the app.
Implemented in	Home View, Image Picker View, Coordinator

Selecting a picture from the gallery

Meets requirement(s)	FR3
Description	When the user clicks on the button to select the recording option and chooses the option "Select picture from gallery", the picture gallery opens. Here the user can select an image of their choice, crop it and, if required, select another photo to be used for the analysis. At any time, the user can return to the home screen.
Technique	For example, if the user selects "Select image from gallery" in the "Classify objects" mode, the following assignments are made: • The variable `sourceType` is set to `.photoLibrary`. • The variable `showPhotosPicker` is set to `true`. • The variable `useCase` is set to `2`.

	As `showPhotosPicker` is set to `true`, a `Sheet` containing the `Image Picker View` is displayed. This `View` allows the camera and image gallery to be managed. Setting `sourceType = .photoLibrary` creates a `UIImagePickerController` that opens the image gallery to select an image. The `Coordinator` is responsible for processing the photo. When an image is selected for analysis, the `Coordinator` is used to route this image back to the `Home View`. Here the image is prepared for later analysis. Then `navigateTo` is set to `2` to allow navigation to the `Single Image View`. In this `View` the selected image is displayed. The `useCase` is also passed to the `Single Image View` to indicate the "Classify objects" mode. The variable `isActiveClassify` is set to `true`. This ensures that the `Single Image View` is not opened before the selected image is fully loaded for analysis. If instead the "Check organic waste" mode is selected, `useCase` is set to `1` and instead of `isActiveClassify`, `isActiveBio` is set to `true`. The `useCase` ensures that, depending on the selected mode, either only non-organic waste objects (1) or both organic and non-organic waste objects (2) are detected and labelled. When returning to the `Home View`, the selected image is set to `nil`, `isActiveBio` and `isActiveClassify = false`, `useCase = -1` and `navigateTo = -1` are set in order to be able to exclude a misbehaviour of the app.
Implemented in	Home View, Image Picker View, Coordinator

Using the live view of the camera

Meets requirement(s)	FR4
Description	Clicking on the option to activate the live view within the menu for selecting the recording option opens a preview of the camera, but without the possibility of taking a photo. It is possible to switch between outdoor and indoor camera at most and to return to Home View at any time.

Technique	For example, if the user selects "Live View" in "Classify Objects" mode, the following assignments are made: • The variable `navigateTo` is set to `1`. • The variable `isActiveClassify` is set to `true`. • The variable `useCase` is set to `2`. The navigation to the `Live Camera View` is done by `navigateTo` and `isActiveClassify`. In addition, the `useCase` is passed in order to mark only those objects that are relevant for the selected mode. If the mode "Check Organic waste" is selected instead of the mode "Classify objects", the variable `isActiveBio` is set to `true` instead of `isActiveClassify` and the `useCase` is set to `1`. When returning to the `Home View`, `isActiveBio` and `isActiveClassify` are set to `false`, `useCase` is set to `-1` and `navigateTo` is set to `-1` to eliminate any discrepancies in the app's behaviour.
Implemented in	Home View, Live Camera View

Distinction between modes in terms of recognition and marking

Meets requirement(s)	FR7, FR8
Description	Depending on the selected mode, the marking of the corresponding objects takes place. In the "Classify objects" mode, both organic waste and non organic waste objects are marked, while in the "Check organic waste" mode only the latter are marked.
Technique	In the `Detection` class, the `useCase` is requested and passed from the `Home View` to the Display View (either `Single Image View` or `Live Camera View`). The Display View then passes the `useCase` to the `DataModel`, which acts as the controller for object detection. The `DataModel` uses the function `detectAndProcess` of the `Detection` class, which among other things takes the parameter `useCase`.

	Within this method, the `detect` method is called, which is responsible for the actual execution of the object detection and stores the detected objects in a list `[VNObservation]`. In addition, the `processObservation` function is called. This method processes the detected objects by converting them into absolute image coordinates. Furthermore, the method checks the content of the `[VNObservation]` list. If the `useCase` has the value `1`, which means that only non-organic waste objects should be displayed, a `[String]` list is used to check whether organic waste objects are present within the `[VNObservation]` list. If this is the case, they are simply skipped and not added to the `[DetectedObject]` output list. Code Snippet B.1 shows the method `processObservation` just explained, in which the detected objects are filtered depending on `useCase`.
Implemented in	DataModel, Detection

Object detection model

Meets requirement(s)	FR10, FR11, FR12, FR13
Description	The presented model is characterised by a brisk recognition speed, which makes it well suited for real-time applications, including the live view. Furthermore, it is capable of identifying multiple objects simultaneously. This object recognition model has been integrated into the application in the form of an .ml model file, which means that an internet connection is not required.
Technique	See Section 5.1.
Implemented in	CoreML Model

Information about detected object

Meets requirement(s)	FR14, FR15
Description	Each identified object is represented by a bounding box and an associated confidence value. The bounding boxes are used to determine the exact position and size of the detected object on the image. For better differentiation, the bounding boxes are marked green for organic waste objects and red for non-organic waste objects. The confidence value supplements the information by providing additional details on the reliability of the classification.
Technique	The `DataModel` first performs object detection using the `Detection` class and then uses the resulting list of detected objects to draw bounding boxes with confidence values. This is achieved by the `labelImage` function of the `Labeling` class. In this function, the bounding boxes are drawn individually for each recognised object together with the associated confidence values. The colour of the bounding box is assigned based on the `label` of the detected object, using a `[String]` list is used to determine whether it is organic waste or non organic waste. The confidence values are converted to percentages for ease of use and displayed at the bottom of the bounding boxes. Code Snippet B.2 shows the class `Labeling` with the function `labelImage`.
Implemented in	DataModel, Labeling

Information text about detected object(s)

Meets requirement(s)	FR16
Description	In addition to the analysed image with the bounding boxes, there is an information text about the recognised objects in the image. Depending on the use case, the user is informed differently about the recognised objects.

Technique	The `DataModel` receives the detected objects in the form of a list through the `Detection` class. The information text is then created based on the list of detected objects in the method `adjustObservationsInfo`. For this purpose, the function checks in which `useCase` the app is currently located. In addition, it checks whether only organic waste/non-organic waste objects are detected, and the text is adjusted according to the number of objects. The code for the corresponding method can be viewed in Code Snippet B.3.
Implemented in	DataModel

Result screen with drawn bounding boxes, confidence and information text

Meets requirement(s)	FR9
Description	After selecting the image to be analysed, a display is shown containing the analysed image with the bounding boxes for the corresponding objects according to the selected mode. In addition, the confidence for the corresponding object is drawn into the bounding box. Below the analysed image is an information text, which varies according to the type and number of objects and according to the selected mode (see Figures A.6 and A.7b). In the short time between confirming the selection of the image to be analysed and the result view, a loading screen appears with a GIF and a text informing the user that the image is currently being analysed. Figure A.7a shows the corresponding Loading Screen. If the live view is activated instead of analysing a single photo, a screen opens showing the live view of the camera. In this view, the bounding boxes with confidence values are drawn in directly, depending on the selected mode, in case an object was detected. Below this live view is also the information text, which varies depending on the number and type of detected objects and the selected mode (see Figure A.5).

Technique	In the `Single Image View`, it is first checked whether the AI has completed the analysis process. This is indicated by the Boolean variable `showLoadingScreen`. If the AI is still analysing, an information `Text` is displayed indicating that the image is currently still being processed. At the same time, a `LoadingPop.gif` is displayed in which four icons of recognisable objects rotate in a circle. As soon as the AI has finished the analysis, `showLoadingScreen` is updated and the loading animation is replaced by an `Image Preview` (see Code Snippet B.4). The `Image Preview` displays the analysed photo, including the bounding boxes and the confidence of the detected objects according to the selected mode. The `Image Preview` displays the analysed photo, including the bounding boxes and the confidence of the detected objects according to the selected mode. This image is generated by the `DataModel` with the method `handlePhotoPreview`. Here, the image processing is done using the `labelImage` method from the `Labeling` class. Below the `Image Preview`, the information `Text` is displayed in the `Single Image View`, which is also provided by the `DataModel` with the help of the `adjustObservationsInfo` method. The `Live Camera View`, on the other hand, shows the camera view directly, without the previous loading screen. Here, too, the `Image Preview` is used to display the camera view. The information `Text` also appears below the `Image Preview` (see Code Snippet B.5). The marking of the detected objects is done by the method `handleCameraPreviews` in the `DataModel`. Although the basic logic is the same as for `handlePhotoPreview`, many successive `Images` are analysed in the `Live Camera View` and output via a stream (see Code Snippet B.6). This gives the impression that the object recognition is done in real time in the `Camera`.
Implemented in	DataModel, Single Image View, Live Camera View, Image Preview, Camera

The implementation and realisation of all functional requirements was successfully realised. The developed software provides users with all the necessary functions and features to meet the set requirements.

5.3 Assessment of the fulfilment of the Non-functional Requirements

At the end of this chapter we also want to evaluate the fulfilment of the non-functional requirements.

- **NF1: Precision**
  The model has an mAP@0.5 of 89.55%, which means that it correctly detects and locates objects in an average of 89.55% of the cases when there is a 50% overlap between the predicted and actual bounding box. The achieved accuracy of over 85% thus fulfils our requirements.
- **NF2: Usability**
  The UI was deliberately designed in a simple and easy-to-understand design. The app can be used in both Light Mode and Dark Mode (see Appendix A) and supports all phone orientations. However, a comprehensive assessment of the user-friendliness will only be made in the next chapter after extensive tests and evaluations have been carried out by actual users.
- **NF3: Reliability**
  The use of default values ensures that important functions and variables are always initialised and do not contain invalid values. This practice contributes to the robustness of the app by avoiding possible error conditions. For this reason, it is possible to call it a robust implementation.
- **NF4: Maintainability**
  The architecture of the app is similar to the Model-View-Controller pattern. In addition, comprehensive documentation and comments are available at crucial points in the code. This contributes significantly to the maintainability of the app, facilitating bug fixing and adding new features. Our requirements for the maintainability of the app are therefore fulfilled.

- **NF5: Data privacy**
  Images are processed only locally on the user's device. By integrating the AI model into the app, no internet connection is required for data transfers. Our app does not use any network communication. In addition, analysed images are not stored and are automatically deleted after closing the app. Thus, our app also meets data protection requirements.

In conclusion, we are confident that our project meets the defined non-functional requirements and is well suited for the intended purposes.

6 User Survey and Evaluation

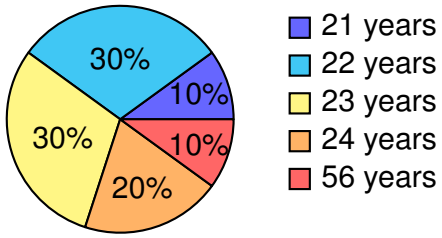
In this chapter, we will focus on the implementation and analysis of our study to evaluate the fully developed app. We will first explain the methodology used. Then we will present the demographic data of the participants. We will continue by sharing the results regarding the participants' personal connection to the topic of waste and waste management. This is followed by a summary of the feedback we received from the participants on the app. Finally, we will conclude this chapter by answering the Research Questions defined in Section 2.3.

6.1 Methodology

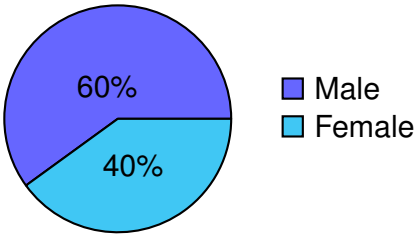
In order to answer the research questions defined in Section 2.3, to collect the personal relation of the participants to the topic of waste and its disposal, and to evaluate the implemented app, a study was conducted. At the beginning of the study, we interviewed the participants to find out about their personal relationship to the topic of waste and its disposal. Afterwards, the participants were given 10 minutes to test the implemented app. All 20 recognisable objects (see Table 5.1 for further information) were placed on a desk. The participants did not receive any instruction on how to use the app, in order to be able to additionally assess the intuitiveness and user-friendliness of the application. After use, feedback and further information were collected from the participants in another interview. Finally, we collected the demographic data of the participants. The entire study was documented using audio recording after the participants had given their consent to the recording.

6.2 Demographic Distribution of Participants

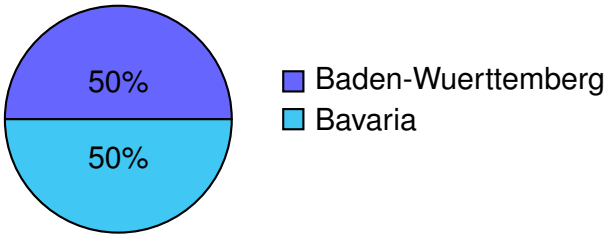
10 participants took part in the study. Their demographic data is shown in the following diagrams.



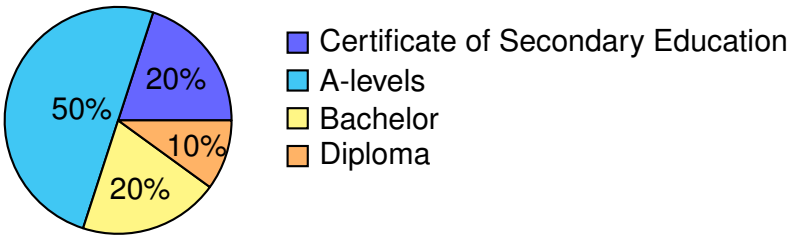
Age of the participants



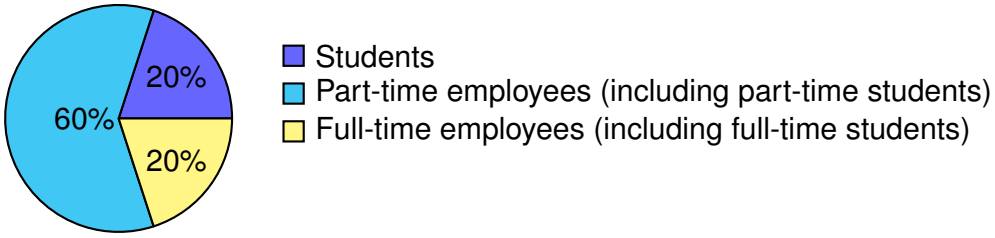
Gender of the participants



State of residence of the participants



Education level of the participants



Professional activity of the participants

6.3 Personal Relation to Waste Disposal

The survey results show that the majority of respondents practice waste separation in their daily lives, indicating a widespread awareness of the need for waste separation. In addition, it is evident that many participants collect organic waste separately, which indicates additional efforts in the area of environmentally friendly waste disposal.

Regarding the cleanliness of waste separation, it is found that most respondents generally keep their waste well sorted. However, there are occasional reports of contamination, especially when waste separation is hindered due to lack of facilities.

Confidence in the proper disposal of waste varies among participants. The vast majority have confidence in the proper disposal of their waste. In case of uncertainty, some respondents consult sources of information such as the internet or imprints on packaging. In some cases, however, the waste is disposed of in the residual waste due to a lack of opportunities for waste separation.

There is considerable variation in the amount of waste produced by participants. Some generate a considerable amount of waste, while others make an active effort to avoid waste. The rest of the interviewees keep to an average level.

In conclusion, the results emphasise the importance of waste separation for the majority of respondents. However, some respondents state that they have not been sufficiently informed about the rules, find differences from place to place a hindrance or consider the mixing of waste categories to be of little use, which leads to them not investing extra time in detailed waste separation.

In addition, respondents express requirements for a waste separation app, highlighting the need for clear allocation of waste categories, identification of waste with

clear allocation details, reliability of the app and ease of use. Some respondents want the app to outperform the internet search in terms of speed.

6.4 Feedback on the App

The feedback on the app opens up valuable insights into the actual use and individual experiences of the users surveyed. Many participants describe the operation of the app as uncomplicated and easy to understand. In particular, the live view and the question mark function are positively highlighted. The simple design of the app and the ability to recognise several objects at the same time also meet with general approval.

However, some of the users surveyed express difficulties in distinguishing between different functions of the app, which can occasionally cause confusion. Criticisms also relate to the speed of the AI reaction, especially for objects held close, as well as the occasional switching of the text when a new object is detected or the detection of an object is omitted.

Regarding the use of the app to check organic waste or to classify objects, there is some variation among the respondents. 10% of the participants prefer to check organic waste, while 60% prioritise the classification of objects. The remaining 30% tend to use both.

The users who make suggestions for improving the app submit a wide range of ideas. These vary from the implementation of a tutorial at the first start to the specification of a certain size for the information area. Some respondents would like to see a supplementary video and examples in the help function, while others point out that symbols should be displayed to help with disposal.

Regarding the question of recommending the app to others, 90% of the users express positive views, especially with regard to its suitability for supporting people without experience in waste separation. However, 10% of the respondents recognise potential for improvement and suggests introducing an incentive system, for example in the form of awarding points for the number of objects scanned.

70% of the respondents consider the possibility of private use of the app, especially in cases of persistent uncertainty about waste separation or disputed objects. The

other 30% consider private use unnecessary, as the system still requires optimisation.

The question of the appropriateness of the app name, namely "WasteGuard", generates different opinions. 30% of the users think that there may be more suitable names, while 70% consider the current name adequate for the task fulfilled.

In summary, the feedback provides valuable information on user satisfaction and possible improvements to the app. It also highlights the diverse usage scenarios and individual user requirements that can be taken into account in the future development of the app.

6.5 Evaluation of the Research Questions

Finally, in this section we present the answers to our Research Questions, which were determined with the help of our study. During and after using the app, we asked specific questions to evaluate and answer the Research Questions.

RQ1: Would such a waste classification system be accepted by the population?

The answer to research question RQ1, which considers the acceptance of an AI-based waste classification system among the population, reveals several key aspects. These aspects influence the willingness of users to accept such a system, even though it may have potentially erroneous classifications.
It turns out that the recognition accuracy of the AI is of crucial importance. Users are more inclined to accept the system if it is able to recognise objects correctly. However, the acceptance of this system is strongly dependent on the frequency of errors. Single errors are more likely to be tolerated, while repeated errors can undermine trust in the system.
Furthermore, the reaction of users to misclassifications is shown to play a role. They tend to attribute these errors to weaknesses in the AI system rather than rejecting the entire system, indicating that users understand the potential of the technology and do not tend to jump to conclusions.

Another interesting aspect is the positive reaction of users to surprising classifications where the AI correctly identifies an object and gives a different classification than the user had assumed. Some of these users consider another source of information when this scenario occurs; if this confirms the classification of the AI when this case occurs repeatedly, the acceptance of and trust in the AI increases.
In addition, it shows that users have confidence in the AI from the start if it answers correctly. This underlines the importance of high recognition accuracy from the beginning.

In summary, a waste classification system could be accepted by the population as long as it has a high recognition accuracy. The provision of clear information, hints and tips as well as comprehensible communication can increase user confidence. However, it should be noted that a certain amount of scepticism and verification of information remains. Such a system could be seen as a useful addition to waste classification without the need for full user acceptance.

RQ2: When it comes to the objects that can be classified by the AI, does quantity or quality matter more?

Research question RQ2 investigates user preferences regarding the importance of quantity versus quality in the classification of objects by AI systems. The evaluated opinions of the users offer valuable insights into this topic.
A small proportion of users place particular emphasis on quantity. They argue that an extensive knowledge base is of great value for AI. From their perspective, the more objects that can be classified, the more useful and informative the system becomes, even if the accuracy is not particularly high.
In contrast, the vast majority of users emphasise the importance of quality in AI classification. They focus on the accuracy and reliability of the system. For them, it is of higher relevance that the AI systems classify the recognised objects with high precision and reliability, even if this is at the expense of quantity.
A small group of users prefers a balanced approach that considers both quality and quantity. They suggest setting a threshold of 70% accuracy, below which the AI is considered unreliable. In particular, for frequently occurring objects, a particularly high accuracy of over 90% should be aimed for, while for less frequent objects, an accuracy of over 70% is considered acceptable. High confidence and the detection

of the largest possible number of objects is considered desirable.

The findings illustrate that the quality of AI classification is of outstanding importance to the majority of users, with a focus on reliable and accurate classification. A smaller proportion of users prefer a balanced approach that considers both quality and quantity. A minority emphasises quantity, considering a comprehensive knowledge base valuable for AI. This underlines the complexity of users' preferences and requirements in the context of AI systems for object classification.

7 Summary and Outlook

In this thesis, an app based on Artificial Neural Networks was developed and used for object recognition to enable the classification of organic and non-organic waste. The motivation for this app arose from the increasing amount of organic waste and the resulting increase in contaminants in it. In connection with the growing interest in Artificial Intelligence (AI) and its diverse application possibilities, the idea to develop an iOS app to improve the purity of organic waste with the help of object recognition was born.

First, different use cases were analysed to derive functional and non-functional requirements. Before implementing the AI technology, a solid understanding of the underlying functionality and suitable architectures was required. In the process, we came across so-called Single Stage Detector (SSD)s, which enable the classification and localisation of objects with the help of so-called Convolutional Neural Networks. We decided to use a SSD architecture for the implementation and trained it with the help of a specially compiled data set in order to be able to integrate it into the iOS app afterwards. The app's functionalities were developed according to the previously defined requirements and then evaluated in a study.

The results showed that the app was found to be particularly useful in situations where there is uncertainty about waste separation. Users rated the app as intuitive, although they also voiced criticisms and suggestions for improvement. In particular, they criticised occasional errors in the AIoutput, which led to some scepticism. We were thus able to establish that a high level of accuracy is a prerequisite for the population to also trust the app, especially the AI. The participants also confirmed this by the fact that the quality of object recognition plays a greater role than the quantity of recognisable objects.

As next steps, measures could be taken to improve the AI accuracy, for example by extending the training dataset and re-training the model. In addition, the application could be extended to other waste categories, such as paper or packaging waste. Developing a version for Android smartphones would extend the reach of the app and provide Android users with the same clean waste separation benefits. In addition, the app could be further developed by, for example, introducing a reward system for the number of items collected or the cleanliness of the waste to encourage use. Cooperations with waste management companies and local authorities could promote the integration of the app into existing waste management systems and thus improve the efficiency and sustainability of waste management.

Bibliography

[1] Statistisches Bundesamt. *457 Kilogramm Haushaltsabfälle pro Kopf im Jahr 2019: 2 Kilogramm mehr als 2018*. URL: https://www.destatis.de/DE/Presse/Pressemitteilungen/2020/12/PD20_511_321.html (visited on 10/01/2023).

[2] Statistisches Bundesamt. *Neue Rekordmenge an Haushaltsabfällen im Jahr 2021*. URL: https://www.destatis.de/DE/Presse/Pressemitteilungen/2022/12/PD22_546_321.html (visited on 10/01/2023).

[3] Statistisches Bundesamt. *Aufkommen an Haushaltsabfällen: Deutschland, Jahre, Abfallarten*. URL: https://www-genesis.destatis.de/genesis/online?sequenz=tabelleErgebnis&selectionname=32121-0001&zeitscheiben=2#abreadcrumb (visited on 10/02/2023).

[4] Umweltbundesamt. *Bioabfälle*. URL: https://www.umweltbundesamt.de/daten/ressourcen-abfall/verwertung-entsorgung-ausgewaehlter-abfallarten/bioabfaelle (visited on 10/01/2023).

[5] Michael Kern et al. *Sortenreinheit von Bioabfällen*. LUBW Landesanstalt für Umwelt Baden-Württemberg. 2018. URL: https://pudi.lubw.de/detailseite/-/publication/67195 (visited on 10/02/2023).

[6] OpenAI. *Introducing ChatGPT*. URL: https://openai.com/blog/chatgpt (visited on 09/30/2023).

[7] Google Trends. *Explore "Artificial Intelligence" and "ChatGPT" on Google Trends*. URL: https://trends.google.com/trends/explore?date=today%205-y&q=%2Fm%2F0mkz,ChatGPT&hl=en-GB (visited on 09/30/2023).

[8] Google Trends. *Explore - Google Trends*. URL: https://trends.google.com/trends/explore?hl=en-GB (visited on 09/30/2023).

[9] Sumit Das et al. “Applications of artificial intelligence in machine learning: review and prospect”. In: *International Journal of Computer Applications* 115.9 (2015). DOI: 10.5120/20182-2402.

[10] Gerhard Paaß and Dirk Hecker. *Künstliche Intelligenz: Was Steckt Hinter der Technologie der Zukunft?* Springer Fachmedien Wiesbaden, 2020. DOI: 10.1007/978-3-658-30211-5. URL: https://doi.org/10.1007/978-3-658-30211-5.

[11] Martijn Kuipers and Ramjee Prasad. “Journey of Artificial Intelligence”. In: *Wireless Personal Communications* 123.4 (Oct. 2021), pp. 3275–3290. DOI: 10.1007/s11277-021-09288-0. URL: https://doi.org/10.1007/s11277-021-09288-0.

[12] “artificial intelligence, n.” In: *Oxford English Dictionary*. Oxford University Press, July 2023. DOI: 10.1093/oed/3194963277. URL: https://doi.org/10.1093/oed/3194963277.

[13] Irene Teich. “Meilensteine der Entwicklung Künstlicher Intelligenz”. In: *Informatik Spektrum* 43.4 (June 2020), pp. 276–284. DOI: 10.1007/s00287-020-01280-5. URL: https://doi.org/10.1007/s00287-020-01280-5.

[14] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach*. 3rd ed. Pearson, 2009.

[15] Peter Buxmann and Holger Schmidt. “Grundlagen der Künstlichen Intelligenz und des Maschinellen Lernens”. In: *Künstliche Intelligenz*. Springer Berlin Heidelberg, 2021, pp. 3–25. DOI: 10.1007/978-3-662-61794-6_1. URL: https://doi.org/10.1007/978-3-662-61794-6_1.

[16] Rockwell Anyoha. *The History of Artificial Intelligence*. URL: https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/ (visited on 10/05/2023).

[17] Joachim Reinhart et al. *Künstliche Intelligenz - eine Einführung: Grundlagen, Anwendungsbeispiele und Umsetzungsstrategien für Unternehmen*. 1st ed. Vogel Communications Group, 2021. ISBN: 978-3-8343-3511-1.

[18] Uwe Lämmel and Jürgen Cleve. *Künstliche Intelligenz: Wissensverarbeitung - Neuronale Netze*. 6th ed. Carl Hanser Verlag GmbH & Co. KG, München, 2023. ISBN: 978-3-446-47881-7.

[19] Christos Stergiou and Dimitrios Siganos. *Neural Networks*. Institute of Informatics, Sokhumi State University. 2014. URL: https://srii.sou.edu.ge/neural-networks.pdf (visited on 10/15/2023).

[20] Fadja Nguembang et al. “Vision Inspection with Neural Networks”. In: Dec. 2018.

[21] Kandarpa Sarma. “Learning Aided Digital Image Compression Technique for Medical Application”. In: Dec. 2015, pp. 400–424. ISBN: 978-1-4666-8654-0. DOI: 10.4018/978-1-4666-8654-0.ch019.

[22] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9.4 (2018), pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. URL: https://doi.org/10.1007/s13244-018-0639-9.

[23] Nafiz Shahriar. *What is Convolutional Neural Network - CNN (Deep Learning)*. URL: https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5 (visited on 10/18/2023).

[24] Zhong-Qiu Zhao et al. “Object Detection With Deep Learning: A Review”. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (2019), pp. 3212–3232. DOI: 10.1109/TNNLS.2018.2876865.

[25] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. “Deep Neural Networks for Object Detection”. In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/f7cade80b7cc92b991cf4d2806d6bd78-Paper.pdf.

[26] Licheng Jiao et al. “A Survey of Deep Learning-Based Object Detection”. In: *IEEE Access* 7 (2019), pp. 128837–128868. DOI: 10.1109/access.2019.2939201. URL: https://doi.org/10.1109%2Faccess.2019.2939201.

[27] Nafiz Shahriar. *Mean Average Precision (mAP) Explained: Everything You Need to Know*. URL: https://www.v7labs.com/blog/mean-average-precision (visited on 10/18/2023).

[28] Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. “Non-maximum Suppression for Object Detection by Passing Messages Between Windows”. In: *Computer Vision – ACCV 2014*. Springer International Publishing, 2015, pp. 290–306. DOI: 10.1007/978-3-319-16865-4_19. URL: https://doi.org/10.1007/978-3-319-16865-4_19.

[29] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[30] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2. URL: https://doi.org/10.1007%2F978-3-319-46448-0_2.

[31] Deep Patel et al. “Garbage Detection using Advanced Object Detection Techniques”. In: *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*. 2021, pp. 526–531. DOI: 10.1109/ICAIS50930.2021.9395916.

[32] Liu Bohong and Wang Xinpeng. “Garbage Detection Algorithm Based on YOLO v3”. In: *2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*. 2022, pp. 784–788. DOI: 10.1109/EEBDA53927.2022.9744738.

[33] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-yuan Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. July 2022. DOI: 10.48550/arXiv.2207.02696.

[34] Wong Kin-Yiu et al. *yolov7*. GitHub Repository. URL: https://github.com/WongKinYiu/yolov7 (visited on 10/20/2023).

[35] Entsorgungs-Betriebe der Stadt Ulm. *Biomüll*. URL: https://www.ebu-ulm.de/abfall/muelltrennung-biotonne.php#:~:text=Was%20wollen%20Sie%20Entsorgen%3F&text=FÃijr%20alle%20organischen%20AbfÃďlle%20aus,und%20zu%20hochwertigem%20Kompost%20verarbeitet. (visited on 10/19/2023).

[36] Abfallwirtschaft Alb-Donau-Kreis. *Biotonne im Alb-Donau-Kreis*. URL: https://www.aw-adk.de/abfallarten/biogut/ (visited on 10/19/2023).

[37] James Skelton. *How to train and use a custom YOLOv7 model*. URL: https://blog.paperspace.com/yolov7/ (visited on 10/19/2023).

[38] SDP2. *Apple Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/sdp2/apple-1fsqz (visited on 10/20/2023).

[39] FruitFinding Robot. *Fruit Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/fruitfinding-robot/fruit-boah1 (visited on 10/20/2023).

[40] Apples. *Red and Green Apples Single Classification Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/apples-1lrlt/red-and-green-apples-single-classification (visited on 10/20/2023).

[41] ENGR407. *Bone Detection Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/engr407-rkihh/bone-detection-guuif (visited on 10/20/2023).

[42] Paddle Paddle Yolo. *Trash Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/paddle-paddle-yolo/trash-vtjsd (visited on 10/20/2023).

[43] Kozhukhov Alexander. *toloka_eggs Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/kozhukhov-alexander/toloka_eggs (visited on 10/20/2023).

[44] Paddle Paddle Yolo. *Trash Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/paddle-paddle-yolo/trash-vtjsd (visited on 10/20/2023).

[45] Keziah Angelica Amamio. *COMPOSTINATOR Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/keziah-angelica-amamio/compostinator (visited on 10/20/2023).

[46] ChuTa. *RUb Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/chuta-j6bnc/rub-cjcyg (visited on 10/20/2023).

[47] down and feather. *down and feater detection Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/down-and-feather/down-and-feater-detection (visited on 10/20/2023).

[48] Alinabelko. *feathersv1-dataset*. GitHub Repository. URL: https://github.com/feathers-dataset/feathersv1-dataset/tree/master/data (visited on 10/20/2023).

[49] Siddharth Singh Chouhan. *A Database of Leaf Images: Practice towards Plant Conservation with Plant Pathology*. 2019. DOI: 10.17632/HB74YNKJCN.1. URL: https://data.mendeley.com/datasets/hb74ynkjcn/1.

[50] garbage. *orange-peels Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/garbage-ci9br/orange-peels (visited on 10/20/2023).

[51] Machine Learning. *GarbageAI Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/machine-learning-r4n0c/garbageai (visited on 10/20/2023).

[52] soilph. *soil.ph Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/soilph-yku76/soil.ph (visited on 10/20/2023).

[53] keng. *Nitrogen Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/keng-kva3g/nitrogen-h0jkc (visited on 10/20/2023).

[54] Craig laboni. *flowers Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/craig-iaboni/flowers-6ooub (visited on 10/20/2023).

[55] cw. *Batteries Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/cw/batteries-hrxj7 (visited on 10/20/2023).

[56] home. *asd2 Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/home-yuoa8/asd2-h43nr (visited on 10/20/2023).

[57] silvia.hernandez01@ustabuca.edu.co. *EtiquetadoFredy Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/silvia-hernandez01-ustabuca-edu-co/etiquetadofredy (visited on 10/20/2023).

[58] garbage. *cigarette butts Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/garbage-ci9br/cigarette-butts-1z3jq (visited on 10/20/2023).

[59] Yang. *mask Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/yang-szmop/mask-rah2h (visited on 10/20/2023).

[60] Palm. *mask_detect Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/palm-gytxh/mask_detect-aihy5 (visited on 10/20/2023).

[61] asdasd. *Glass Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/asdasd-boz3q/glass-xrghl (visited on 10/20/2023).

[62] Sesac. *Pill Detection Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/sesac-49prp/pill-detection-0ixnz (visited on 10/20/2023).

[63] school2. *plastic bag Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/school2-iqyyf/plastic-bag-cr8xz (visited on 10/20/2023).

[64] WasteCreative. *WasteCreative Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/wastecreative/wastecreative-cusat (visited on 10/20/2023).

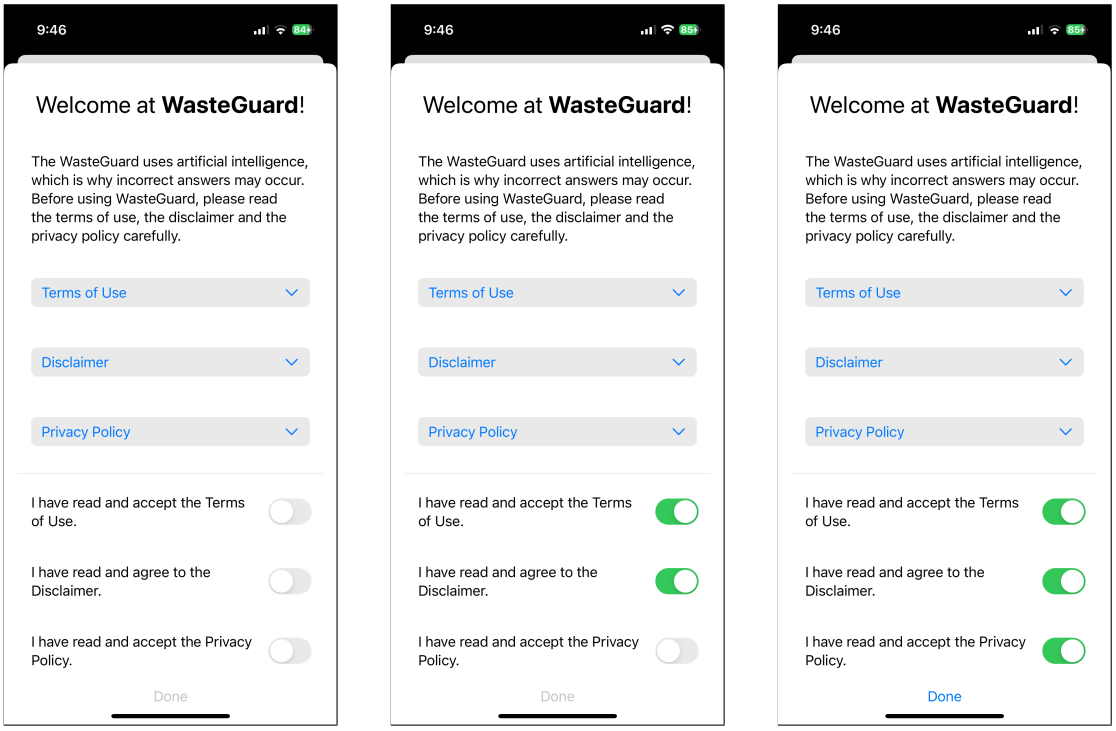
[65] Chris Conrad. *Rocks Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/chris-conrad-le7vi/rocks-s0gnn (visited on 10/20/2023).

[66] SNU. *SURFACEv1 Dataset*. Open Source Dataset. URL: https://universe.roboflow.com/snu-treib/surfacev1 (visited on 10/20/2023).

[67] Google Colab. *Google Colaboratory*. URL: https://colab.research.google.com (visited on 10/20/2023).

[68] Jiaju Miao and Wei Zhu. "Precision–recall curve (PRC) classification trees". In: *Evolutionary Intelligence* 15.3 (Apr. 2021), pp. 1545–1569. DOI: 10.1007/s12065-021-00565-2. URL: https://doi.org/10.1007/s12065-021-00565-2.

Appendix A: App Screenshots

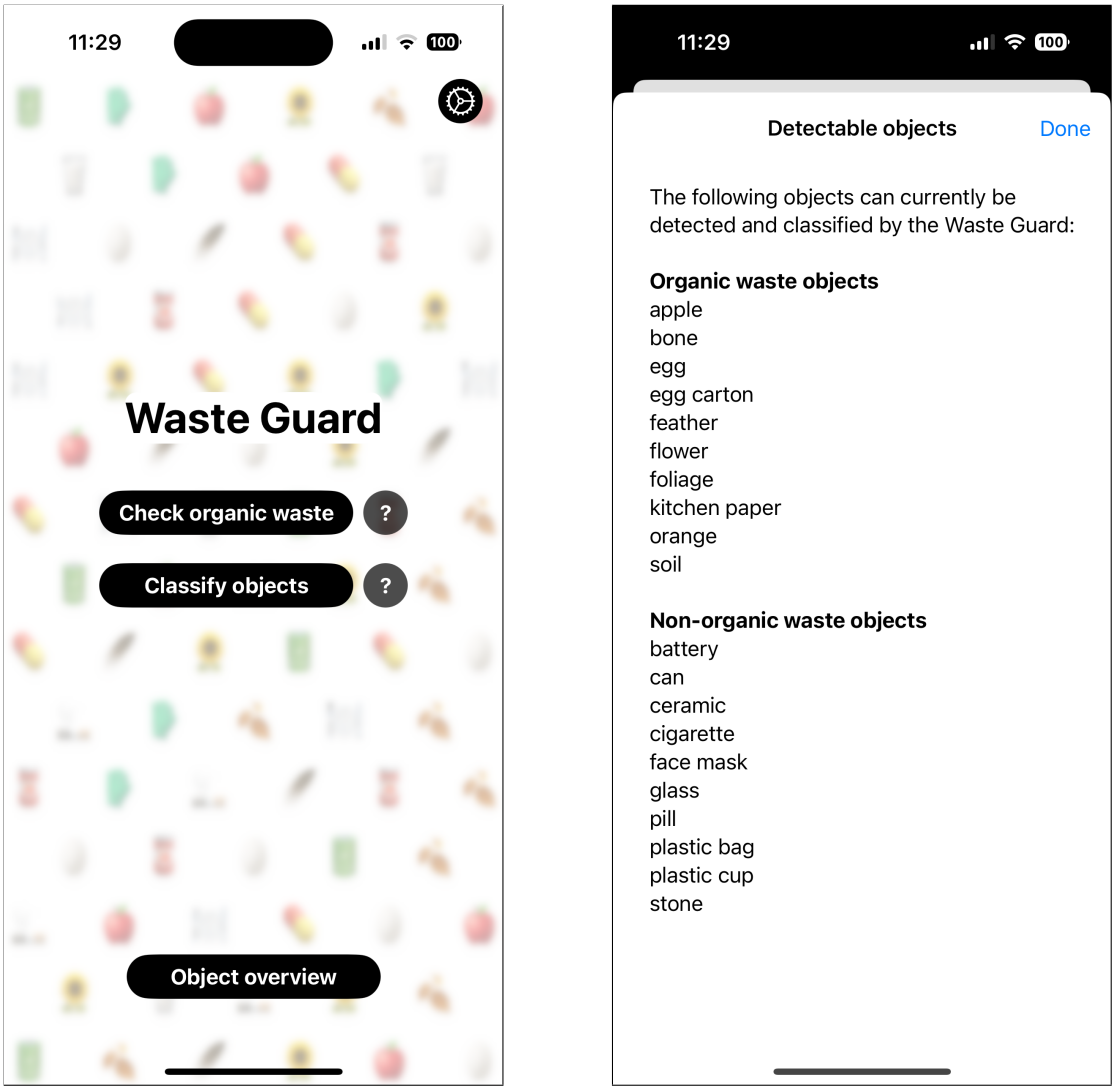


(a) No switch activated, "Done" button is disabled

(b) Two switches activated, "Done" button is disabled

(c) All switches activated, "Done" button is enabled & can be clicked

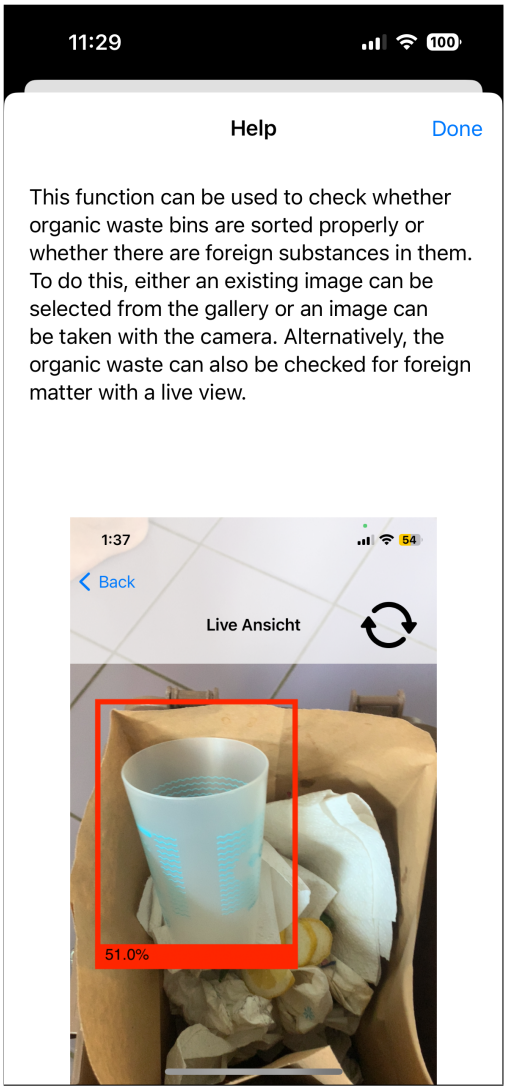
Figure A.1: Dialog window for accepting Terms of Use, Disclaimer & Privacy Policy (System appearance is set to light)



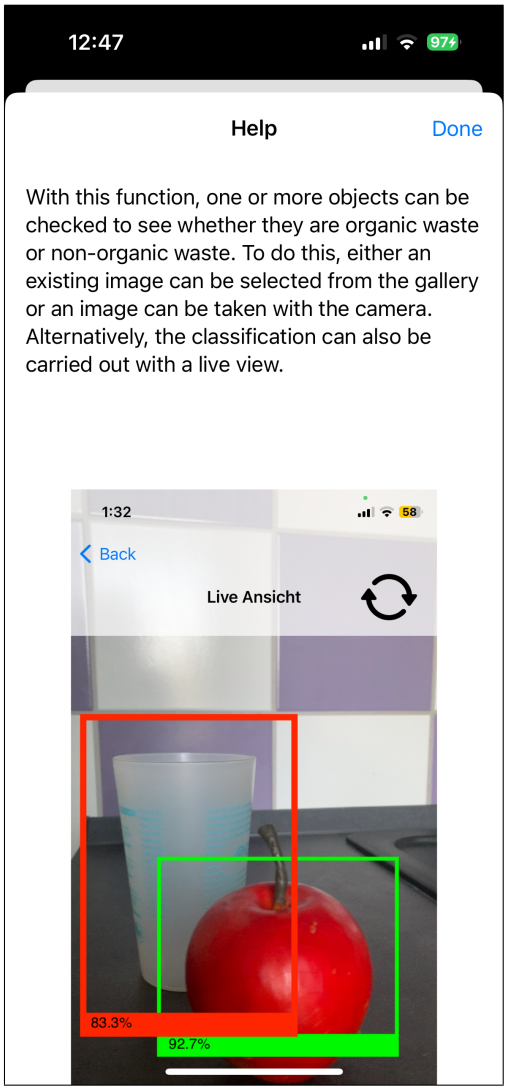
(a) Home screen

(b) Object overview of detectable objects

Figure A.2: Home screen and Object overview of the app (System appearance is set to light)

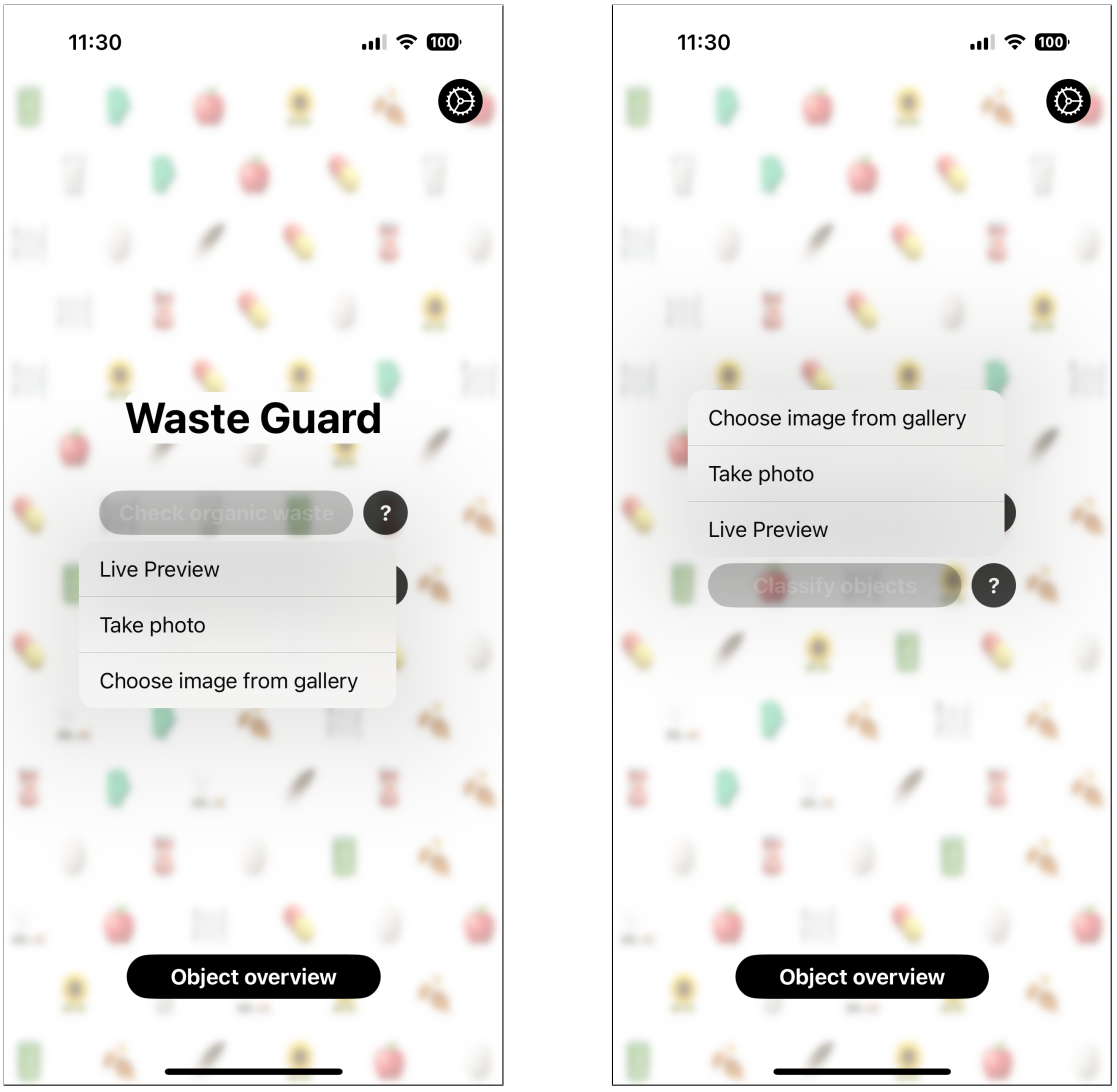


(a) Information sheet for the "Check organic waste" mode



(b) Information sheet for the "Classify objects" mode

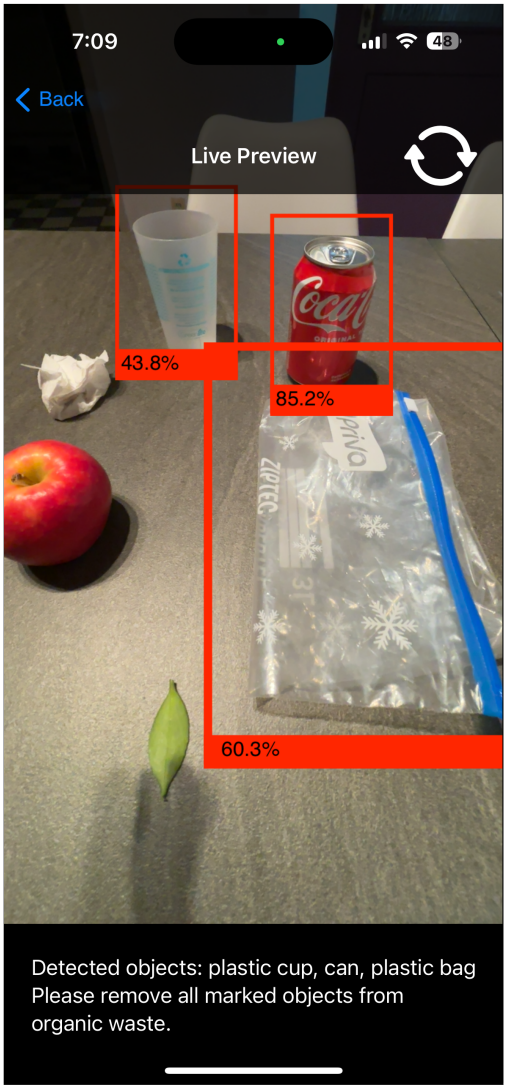
Figure A.3: Information about the different modes available (System appearance is set to light)



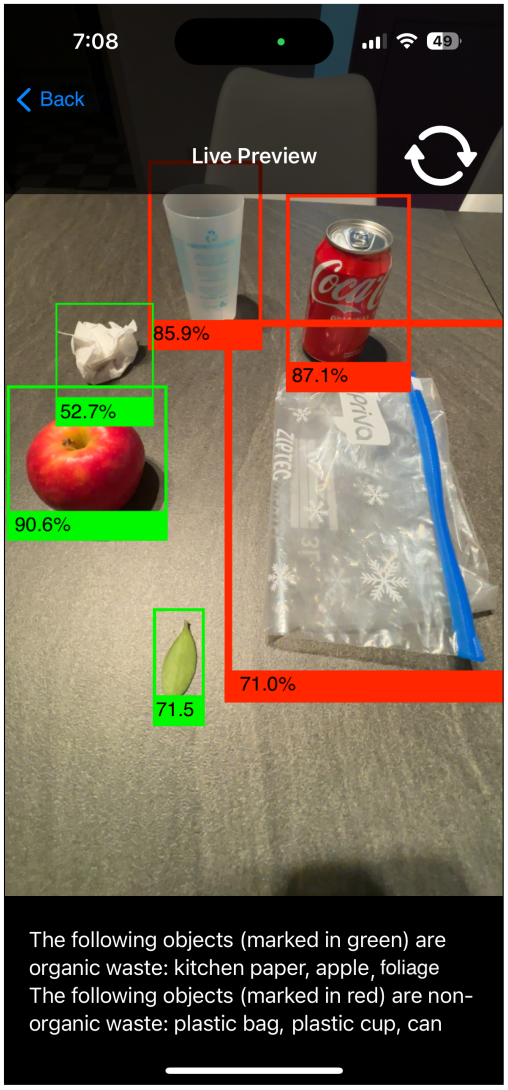
(a) Menu when clicking on the "Check organic waste" button

(b) Menu when clicking on the "Classify objects" button

Figure A.4: Menu for selecting the capture option (System appearance is set to light)



(a) Live view for mode "Check organic waste", only non-organic waste objects are highlighted



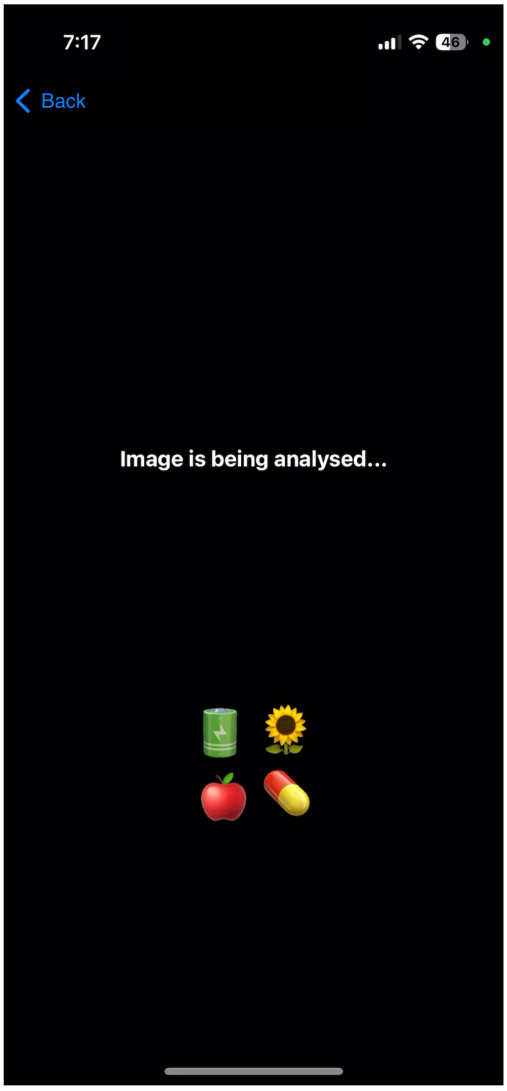
(b) Live view for mode "Classify objects", both organic and non-organic waste objects are highlighted

Figure A.5: Live view for both modes (System appearance is set to dark)

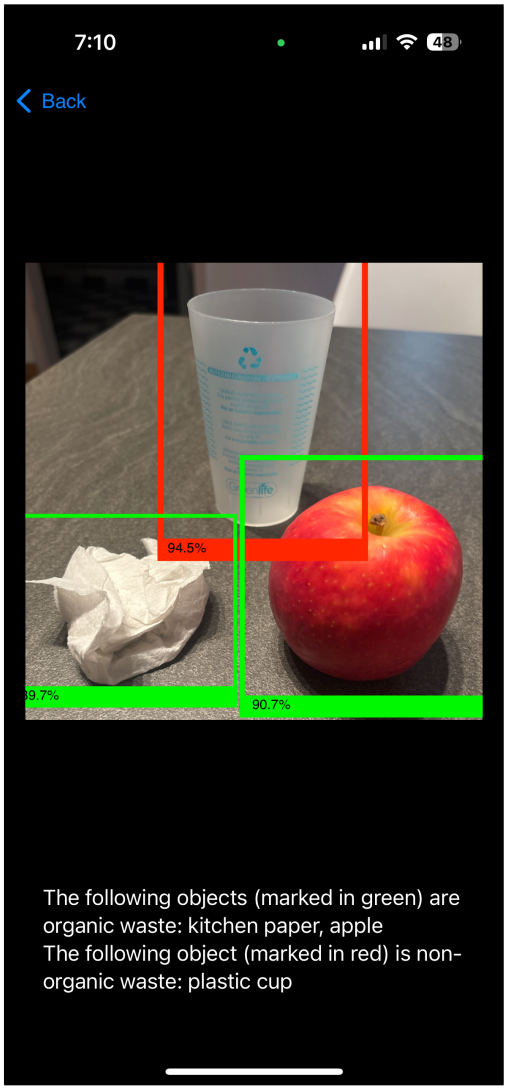
(a) Apple, foliage and kitchen paper are all organic waste. As there are no foreign matter objects in the picture, no bounding boxes are drawn

(b) The plastic cup is the only foreign matter object in the picture and therefore the only object with a bounding box

Figure A.6: Different behaviour of app in "Check organic waste" mode (System appearance is set to dark)



(a) The loading screen while a single image is being analysed. The icons are animated in a GIF & rotate in a circle



(b) The single image view for the mode "Classify objects" after a picture has been analysed

Figure A.7: Procedure when a single image has been selected for analysis. A.7a is shown until all predictions are made (<2 seconds) (System appearance is set to dark)

Appendix B: App Code Snippets

```
private var objectsBiov2: [String] = ["feather", "flower", "egg", "kitchen paper"
    , "apple", "foliage", "soil", "egg carton", "orange", "bone"]
/**
 Processes object detection observations.

 - Parameters:
    - observations: A list of VNObservation objects.
    - viewSize: Size of the display area.
    - useCase: Integer specifying object detection purpose.

 - Returns: A list of detected objects.

 - Note: This method converts object observations to DetectedObject instances,
     optionally skipping organic waste objects for useCase 1.
 */
func processObservation(observations: [VNObservation], viewSize: CGSize, useCase:
     Int) -> [DetectedObject] {
    var processedObservations: [DetectedObject] = []

    for observation in observations where observation is
        VNRecognizedObjectObservation {
        let objectObservation = observation as! VNRecognizedObjectObservation

        // Convert normalized bounding box to image coordinates
        let objectBounds = VNImageRectForNormalizedRect(objectObservation.
            boundingBox, Int(viewSize.width), Int(viewSize.height)

        // Adjust the bounding box to match the display size
        let flippedBox = CGRect(x: objectBounds.minX, y: viewSize.height -
            objectBounds.maxY, width: objectBounds.maxX - objectBounds.minX,
            height: objectBounds.maxY - objectBounds.minY)

        // Get the label and confidence of the detected object
        let label = objectObservation.labels.first!.identifier

        // Create a DetectedObject instance
        let processedOD = DetectedObject(label: label, confidence:
            objectObservation.confidence, rectangle: flippedBox)

        // Skip organic waste objects for useCase 1
        if ((objectsBiov2.contains(label)) && useCase == 1) {
        continue
        } else { processedObservations.append(processedOD)
```

```
        }
    }
    return processedObservations
}
```

Code Snippet B.1: processObservation Method of the Detection class

```
class Labeling {

    private var objectsNonBiov2: [String] = ["cigarette", "pill", "plastic cup",
        "plastic bag", "glass", "face mask", "ceramic", "can", "battery", "stone"
        ]

    /**
     Add detected objects to an image and annotate with colored bounding boxes
         and confidence values.
     */
    func labelImage(image: UIImage, detections: [DetectedObject]) -> UIImage? {
        // Create a new graphics context for annotating the image
        UIGraphicsBeginImageContext(image.size)
        image.draw(at: CGPoint.zero)  // Draw the original image as the
            background
        let context = UIGraphicsGetCurrentContext()!

        for detection in detections {
            var color: CGColor?
            color = self.determineBoundingBoxColor(for: detection.label)  //
                Determine the bounding box color

            let confidenceLabel = String(format: "%.1f", detection.confidence *
                100) + "%"  // Format the confidence as a label
            let boundingBox = detection.rectangle

            self.drawBoundingBox(context: context, bounds: boundingBox, color:
                color!)  // Draw the bounding box
            self.drawConfidenceLabel(context: context, label: confidenceLabel,
                near: boundingBox)  // Annotate with the confidence label
        }

        let annotatedImage = UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext()
        return annotatedImage  // Return the annotated image
    }

    // Drawing Bounding Box
    func drawBoundingBox(context: CGContext, bounds: CGRect, color: CGColor) {
        context.setStrokeColor(color)
        context.setLineWidth(bounds.height * 0.02)
        if bounds.height < 125 {
            context.setLineWidth(10)
        }
        context.addRect(bounds)
        context.drawPath(using: .stroke)
    }
```

```
    // Drawing Confidence Label
    func drawConfidenceLabel(context: CGContext, label: String, near bounds:
        CGRect) {
        context.setFillColor(CGColor.black)
        context.addRect(self.calculateLabelBounds(near: bounds))  // Calculate
            the position and size of the label box
        context.drawPath(using: .fill)

        let labelTextColor = UIColor.black
        let labelFont = UIFont(name: "Helvetica", size: 35)!

        let labelFontAttributes = [
            NSAttributedString.Key.font: labelFont,
            NSAttributedString.Key.foregroundColor: labelTextColor
        ] as [NSAttributedString.Key : Any]

        label.draw(in: self.calculateLabelBounds(near: bounds).offsetBy(dx:
            bounds.width * 0.05, dy: bounds.height * 0.05), withAttributes:
            labelFontAttributes)  // Draw the confidence label
    }

    // Determine Bounding Box Color
    func determineBoundingBoxColor(for label: String) -> CGColor {
        return objectsNonBiov2.contains(label) ? CGColor(red: 255, green: 0, blue
            : 0, alpha: 1) : CGColor(red: 0, green: 255, blue: 0, alpha: 1)
    }

    // Additional functions (grouped in { ... }) - Drawing text boxes and
        calculating text bounds
    {
        // ...
    }
}
```

Code Snippet B.2: Labeling class with labelImage method

```
private var objectsBiov2: [String] = ["feather", "flower", "egg", "kitchen paper"
    , "apple", "foliage", "soil", "egg carton", "orange", "bone"]
private var objectsNonBiov2: [String] = ["cigarette", "pill", "plastic cup", "
    plastic bag", "glass", "face mask", "ceramic", "can", "battery", "stone"]

@Published var observationInformation: String = ""
@Published var useCase: Int?

// Function to adjust observation information based on detected objects
private func adjustObservationsInfo(observations: [DetectedObject]) {
    // Check if only non-organic waste is detected
    if observations.allSatisfy({ objectsNonBiov2.contains($0.label) }) && !
        observations.isEmpty {
        if useCase == 1 {
            let objects = observations.map { $0.label }
            let objectsSet = Array(Set(objects))
            observationInformation = objects.count == 1 ? "Detected object: \(
                objectsSet.joined(separator: ", "))\nPlease remove the marked
                object from organic waste.\n" : "Detected objects: \(objectsSet.
```

```
            joined(separator: ", "))\nPlease remove all marked objects from
            organic waste.\n"
    } else if useCase == 2 {
        let objects = observations.map { $0.label }
        let objectsSet = Array(Set(objects))
        observationInformation = objects.count == 1 ? "Detected object: \(
            objectsSet.joined(separator: ", "))\nThe marked object is non-
            organic waste.\n" : "Detected objects: \(objectsSet.joined(
            separator: ", "))\nAll marked objects are non-organic waste.\n"
    }
}
// Check if only organic waste is detected
else if observations.allSatisfy({ objectsBiov2.contains($0.label) }) && !
    observations.isEmpty {
    if useCase == 2 {
        let objects = observations.map { $0.label }
        let objectsSet = Array(Set(objects))
        observationInformation = objects.count == 1 ?  "Detected object: \(
            objectsSet.joined(separator: ", "))\nThe marked object is organic
             waste.\n" : "Detected objects: \(objectsSet.joined(separator: ",
             "))\nAll marked objects are organic waste.\n"
    }
}
// Check if both organic and non-organic waste are detected
else if observations.contains(where: { objectsBiov2.contains($0.label) }) &&
    observations.contains(where: { objectsNonBiov2.contains($0.label) }) && !
    observations.isEmpty {
    if useCase == 2 {
        var bioObjects: [String] = []
        var nonBioObjects: [String] = []
        observations.forEach { observation in
            if objectsBiov2.contains(observation.label) {
                bioObjects.append(observation.label)
            } else if objectsNonBiov2.contains(observation.label) {
                nonBioObjects.append(observation.label)
            }
        }
        let bioObjectsSet = Array(Set(bioObjects))
        let nonBioObjectsSet = Array(Set(nonBioObjects))
        let bioObjectText = bioObjects.count == 1 ? "The following object (
            marked in green) is organic waste: \(bioObjectsSet.joined(
            separator: ", "))\n" : "The following objects (marked in green)
            are organic waste: \(bioObjectsSet.joined(separator: ", "))\n"
        let nonBioObjectText = nonBioObjects.count == 1 ? "The following
            object (marked in red) is non-organic waste: \(nonBioObjectsSet.
            joined(separator: ", "))\n" : "The following objects (marked in
            red) are non-organic waste. \(nonBioObjectsSet.joined(separator:
            ", "))\n"
        observationInformation = "\(bioObjectText)\(nonBioObjectText)"
    }
}
// Check if no objects are detected
else if observations.isEmpty {
    if useCase == 1 {
        observationInformation = "No foreign matter was detected in the waste
            .\n"
```

```
        } else if useCase == 2 {
            observationInformation = "No objects were detected."
        }
    }
}
```

Code Snippet B.3: adjustObservationsInfo method of the DataModel class

```
struct SingleImageView: View {
    @StateObject private var model = DataModel()
    @Environment(\.dismiss) var dismiss

    @State var useCase: Int
    @State var picture: Data?
    @State var showLoadingScreen: Bool = false

    var body: some View {
        NavigationView {
            if(!showLoadingScreen) {
                // Display loading message when the image is being analysed.
                VStack{
                    Text("Image is being analysed...")
                    GIFImage(name: "LoadingPop")
                }
            } else {
                GeometryReader { geometry in
                    // Once the image is analysed, display the annotated image
                        and observation information.
                    VStack {
                        ImagePreview(image: $model.selectedImage)
                        Text(model.observationInformation)
                    }
                }
            }
        }.task {
            Timer.scheduledTimer(withTimeInterval: 1, repeats: true) { timer in
                // Check if the detection process is ready.
                if(model.detection.ready) {
                    // If the image is provided, handle the photo preview,
                        showing the loading screen while processing.
                    if(picture != nil) {
                        model.handlePhotoPreview(image: CIImage(data: picture!)!,
                             useCase: useCase)
                        showLoadingScreen = true
                        timer.invalidate()
                    }
                }
            }
        }
    }
}
```

Code Snippet B.4: SingleImageView (Code for style adjustment was removed due to irrelevance)

```swift
struct LiveCameraView: View {
    @StateObject private var model = DataModel()

    @State var useCase: Int
    @State private var isImageTaken: Bool = false

    var body: some View {
        NavigationView {
            VStack {
                // Display the live camera view with the detected objects and
                    bounding boxes.
                ImagePreview(image: $model.viewfinderImage)
                    .overlay(alignment: .top) {

                // Display real-time information about detected objects.
                Text(model.observationInformation)
            }
            .toolbar(content: {
                // Button to switch between front and rear cameras.
                Button {
                    model.camera.switchCaptureDevice()
                } label: {
                    Label("Switch Camera", systemImage: "arrow.triangle.2.
                        circlepath")
                }
            })
            .task {
                // Set the selected use case and start the camera feed.
                model.useCase = useCase
                await model.camera.start()
                await model.handleCameraPreviews(useCase: useCase)
            }
            .navigationTitle("Live Preview")
        }
    }
}
```

Code Snippet B.5: LiveCameraView (Code for style adjustment was removed due to irrelevance)

```swift
    let camera = Camera()
    var detection = Detection()
    var labeling = Labeling()

    @Published var selectedImage: Image?
    @Published var viewfinderImage: Image?
    @Published var observationInformation: String = ""
    @Published var useCase: Int?

    // Asynchronously handles camera previews, processes detections, and updates
        UI.
    func handleCameraPreviews(useCase: Int) async {
        let imageStream = camera.previewStream
            .map { $0 }
```

```
        for await image in imageStream {
            Task { @MainActor in

                // If the detection is not ready, show the raw camera preview.
                if !detection.ready {
                    viewfinderImage = image.image
                    return
                }

                camera.isPreviewPaused = true

                // Detect and process objects in the camera preview.
                let detections = self.detection.detectAndProcess(image: image,
                    useCase: useCase)

                // Update the observation information based on detections.
                adjustObservationsInfo(observations: detections)

                // Label the camera preview with bounding boxes and display it.
                let annotatedImage = labeling.labelImage(image: UIImage(ciImage:
                    image), detections: detections)
                viewfinderImage = Image(uiImage: annotatedImage!)

                camera.isPreviewPaused = false
            }
        }
    }

    // Processes a static photo image and updates the UI.
    func handlePhotoPreview(image: CIImage, useCase: Int) {
        // Detect and process objects in the static photo image.
        let detections = self.detection.detectAndProcess(image: image, useCase:
            useCase)

        // Update the observation information based on detections.
        adjustObservationsInfo(observations: detections)

        // Label the photo image with bounding boxes and display it.
        let annotatedImage = labeling.labelImage(image: UIImage(ciImage: image),
            detections: detections)
        selectedImage = Image(uiImage: annotatedImage!)
    }
```

Code Snippet B.6: handleCameraPreviews and handlePhotoPreview methods of DataModel class

Name: Marlene Mika Matrikelnummer: 1042495

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Marlene Mika