**BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA**
**FACULTY OF MATHEMATICS AND COMPUTER SCIENCE**
**SPECIALIZATION COMPUTER SCIENCE IN ENGLISH**

# DIPLOMA THESIS

# NutriNinja: Applying Semantic Segmentation for Efficient Food Recognition and Nutritional Tracking

**Supervisor**
**Conf. Phd. Bocicor Maria-Iuliana**

*Author*
*Oana-Andreea Ilie*

2024

ABSTRACT

This thesis explores the development of an advanced mobile nutrition-tracking application that uses machine learning techniques, specifically food image classification and semantic segmentation, to identify food items from uploaded images. The application aims to provide users with a reliable and precise tool to help monitor their dietary intake and encourage them to make healthier dietary choices. This project intends to combine the traditional manual searches for logging meals with the features offered by the advancements in machine learning to offer a better user experience.

This work addresses the problem of accurately identifying and segmenting food items from images in a mobile application environment. The proposed solution involves developing a model capable of recognising a wide range of food categories and users with reliable nutritional information based on their meals. The research evaluates various configurations of machine learning models to determine the most effective food classification and segmentation approach.

The thesis is organized to present the development process and underlying research systematically. Chapter 2 provides a historical overview of nutritional science and the evolution of nutrition tracking methods. Chapter 3 presents the theoretical foundations supporting this research's approach. Chapter 4 discusses the related work for the proposed solution. The subsequent chapters focus on the design, implementation, and performance evaluation of the food recognition module, followed by a detailed description of the mobile application's system architecture and implementation.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

# Contents

# Chapter 1

# Introduction

## 1.1 Context and motivation

The history of nutrition tracking begins with the history of civilisation and reflects the general interest in health and dietary management. From Ancient civilisations, such as those in Greece, where the link between diet and wellness was recognised[1], to the 19th and 20th centuries, which brought significant advancements with the discovery of essential nutrients [2], nutrition was a highly iterated research subject.

Digital technologies have significantly impacted how people track and manage their dietary habits. The evolution of technology has relieved the process of nutrition tracking, migrating it from paper-based journals to complex mobile applications. These tools offer increased accuracy in dietary logging and facilitate the user experience through features like real-time nutrition analysis.

In recent years, the focus has been on personalising dietary guidelines as much as possible and providing more accurate approaches to dietary goals. The evolution of artificial intelligence and machine learning played a significant role in this domain, making mobile health applications more interactive and personalised. However, at the time of this study, most applications focus only on food classification. A few research papers focus on food segmentation tasks, but the results are not used in a practical application.

This thesis derives its motivation from the need to integrate segmentation tasks, especially semantic segmentation ones, into nutrition-tracking applications. These tasks could significantly improve the application's engagement, allowing the user to precisely see the identified food items, especially in complex meals, with different items on various levels. At the same time, in the case of an erroneous food identification, this application allows the user to easily identify the mistake and manually log the correct food items.

## 1.2 Objectives

This thesis aims to develop an engaging nutrition-tracking application that combines traditional manual search-based calorie trackers with the advancements of machine learning technologies in food image classification and segmentation. It also allows manual adjustments in the identified food items and provides editable default weights for each. This way, the application assures it offers alternatives to users with various preferences while maintaining a dynamic and pleasant user experience.

At the same time, this study will try to offer an accurate classification and segmentation model extended to a large number of food categories. In this process, various configurations will be considered, and their performances will be evaluated to produce a performant model.

Ultimately, this project intends to contribute to mobile health technology by facilitating food logging and encouraging users to track their meals consistently.

## 1.3 Thesis Structure

The thesis is structured to systematically explore and develop a mobile nutrition-tracking application enhanced by machine learning for advanced food image classification and segmentation. Following, we discuss the historical progression of nutritional science and tracking methodologies, presented in Chapter 2. We continue with Chapter 3, which details the theoretical foundations relevant to our approach. Chapter 4 discusses the related work for the proposed solution. Afterwards, we discuss the food recognition module, focusing on its implementation and performance. Finally, we present the mobile application's system design and implementation.

## 1.4 Declaration of Generative AI and AI-assisted technologies during the preparation of this work

The following statement presents a disclosure of the usage of generative AI and AI-assisted technologies.

During the preparation of this thesis, the author employed the generative AI technologies *ChatGPT* and *Grammarly* for spell checking and coherence. Additionally, the author employed the generative AI technologies *ChatGPT* through its image generation feature *DALL·E* for creating the icons for the mobile application. After using these tools, the author reviewed and edited the content as needed and took full responsibility for the content of the thesis.

# Chapter 2

# Evolution of Nutrition Tracking

## 2.1 A Brief History of Nutrition

The history of nutrition dates from the beginning of civilisation. Throughout history, nutrition has significantly evolved along with science and technology. The relationship between nutrition and health has been treated since ancient times when people discovered some treatments for nutrition-related health conditions. Hippocrates, the Greek physician often named the Father of Medicine, underlined the importance of diet for well-being. He proposed diet and exercise to treat some diseases. [1]

The discovery of some nutrients marked a significant evolution in understanding nutrition. Identifying essential nutrients like fats, proteins, and carbohydrates in the 19th century marked the beginning of modern nutrition. The 20th century illustrated notable improvements in nutritional science, leading to a deeper understanding of nutrients and their role in human health. This era began with the first vitamin isolation in 1926 [2], which set the stage for a century of scientific discoveries regarding the impact of diet on health.

The early part of the 20th century is recognised by the identification and synthesis of some essential vitamins and minerals. This process began with the discovery of vitamin A in 1913. This event was followed by the rapid identification of vitamins B1, B2, B3, C, D, and others. Because of that, this era was named the ”era of vitamin discovery”. These discoveries were crucial for identifying and preventing nutritional deficiency-related diseases like nutritional anaemias, scurvy, rickets, beriberi, and pellagra. [2]

The concept of Recommended Dietary Allowances (RDAs) was first introduced in the middle of the 20th century, during World War II and the Great Depression, as the result of the recognition that people required specific amounts of nutrients to avoid deficiencies and maintain health. The first RDAs helped ensure that military personnel and civilians ate adequate nutritional intake during the food rationing periods.

Nutritional debates during the last half of the 20th century were often centred around the roles of fat and sugar in chronic diseases. Due to the prevalence of cardiovascular diseases,

the focus was on dietary fats and their association with heart health. The work of John Yudkin [3] and others established the effects of excess sugar in dental caries, coronary diseases, hypertriglyceridemia, and cancer.

In the context of global hunger and malnutrition issues, the discussion about protein versus calorie intake gained prominence. Scientific research emphasised that the quantity of food and its quality was substantial. Diets should be calorie-sufficient and protein-adequate, addressing energy and essential amino acid requirements.

In the latter part of the 20th century, many countries began developing dietary guidelines as a response to the growing research on diet and chronic disease. These guidelines aimed to provide advice on macronutrient ratios and food quality to improve health outcomes and prevent chronic diseases. These guidelines were adapted together with new scientific discoveries to prevent contemporary health challenges such as obesity, diabetes, and other diet-related conditions. [4]

At the same time, studies on dietary patterns became noticeable. They examined the health outcomes of diets as a whole instead of looking for individual nutrition. Some dietary patterns, like the Mediterranean diet, were associated with numerous health benefits, including longer life expectancy and reduced heart disease risk. [5]

Entering the 21st century, the focus of nutrition science shifted to personalised nutrition, which aims to use individual genetic and metabolic profiles to adapt dietary recommendations. This approach represents a significant evolution from general dietary guidelines and a better approach to chronic disease prevention and management.

## 2.2 Evolution of Nutrition Tracking and Dietary Planning

The concept of nutrition tracking has significantly evolved during the last decades. The interest and awareness regarding nutrition being closely related to health has increased due to numerous studies dedicated to this field. At the beginning of nutrition tracking, people used journals to record all their dietary intakes. The primary purpose of nutrition tracking was to monitor patients' dietary habits to manage various health conditions. This method was subjective and error-prone since it relies on patients' memory and quantity appreciation. Because of that, the records led to inaccurate analyses of nutrition intake and even misledged dietary recommendations.

With the technological evolution and the introduction of computers and the internet, nutrition tracking has become more accessible. The first digital nutrition trackers appeared in the early 2000s as simple websites and basic mobile applications. The main goal of these platforms was to let users log their food intake more accurately and efficiently and provide basic calculations for calories and nutrients. [6]

Concurrently, significant advancements have been made in understanding the nutritional needs of individuals. The development of dietary formulas for computing personalised nutri-

ent requirements marked a pivotal improvement in the field. These formulas consider various factors like age, sex, height, weight, activity levels, and the person's goals (maintain weight, loss, or gain). The establishment of Dietary Reference Intakes (DRIs) covered a set of nutrition recommendations. It expanded the Recommended Dietary Allowances, published by the National Academy of Sciences from 1941 to 1989. At the same time, it expanded the Recommended Nutrient Intakes, published by the Canadian government. These reference values include Recommended Dietary Allowance (RDA), Adequate Intake (AI), Estimated Average Requirement (EAR), and Tolerable Upper Intake Level (UL). [6]

As smartphones and wearable technology became widespread in the late 2000s, nutrition-tracking apps grew more advanced and user-friendly. Popular apps like **MyFitnessPal**[7] and **Fitbit**[8] combine tracking calorie intake with physical activity monitoring and provide a comprehensive overview of health and wellness. These applications use large databases of food nutritional information, barcode scanning, and personalised feedback mechanisms to boost user interaction and increase data accuracy.

Recent advancements have focused on integrating machine learning techniques to improve the functionality of nutrition-tracking applications. These functionalities help simplify the user experience and increase the accuracy of dietary tracking by using features like image recognition to identify food and automatic portion size estimation.

## 2.3 Related Applications for Nutrition Tracking

The field of mobile health applications, with a focus on nutrition-tracking applications, has seen substantial growth and evolution. These applications have considerably altered how people monitor their dietary habits and manage their health. Some estimations from 2021 indicated over 165000 publicly available mobile health apps. At the same time, the adoption rate among users is continually increasing. A study from 2021 about the United States population suggests that 58% of mobile phone users have downloaded an application related to health checks. [9]

From the large variety of mobile health applications, we propose to focus on the domain of nutrition tracking, which has received a rapid expansion, together with the advancements of smartphone technology, wearables technology and artificial intelligence. Early nutrition tracking applications were simple tools for calorie counting and basic dietary logging from the user's input. With all the technological advancements, these applications have started integrating more complex algorithms to provide detailed analyses and personalised dietary recommendations. The way users input their dietary logs has also evolved, especially with the help of artificial intelligence. At present, users can scan the barcode to identify a packaged meal or snap a photo of their plate, and the app will identify the meal and compute the nutritional content.

From all the existing nutrition tracking applications, it is estimated that over 10000 mo-

bile applications focus on diet and weight management, demonstrating users' interest in healthier eating habits and lifestyle changes. These applications are not only designed to track calorie intake and balance macronutrients for weight loss or gain but also offer capable functionalities that cover specific dietary needs and health conditions. These specialised applications provide support for chronic health conditions management, such as diabetes and gastrointestinal disorders. For example, they allow users to monitor their carbohydrate and sugar intake for diabetes management or identify trigger foods in gastrointestinal conditions. [9]

We will focus on analysing the main features of five of the most important mobile nutrition tracking applications, according to Samad et al.[10]: **Foodvisor**[11], **MyFitnessPal**[12], **LoseIt!**[13], **MyNetDiary**[14] and **Lifesum**[15]. They analysed 80 mobile food consumption tracking and recommendation applications by computing ratings for performance, aesthetics, usability, impact, transparency, subjective, and mean.

Table 2.1 represents their scoring for the five applications we proposed to analyse.

| App name | MyFitnessPal | LoseIt! | MyNetDiary | Lifesum | Foodvisor |
|---|---|---|---|---|---|
| Aesthetics | 4.75 | 4.50 | 5.00 | 5.00 | 4.00 |
| General | 4.43 | 4.14 | 3.57 | 3.14 | 2.43 |
| Performance | 5.00 | 5.00 | 5.00 | 5.00 | 4.83 |
| Usability | 4.75 | 4.75 | 4.75 | 4.00 | 5.00 |
| Functionality | 2.50 | 2.67 | 2.33 | 2.50 | 3.67 |
| Transparency | 4.40 | 4.40 | 4.40 | 4.40 | 3.20 |
| Impact | 4.17 | 4.00 | 3.83 | 4.00 | 3.17 |
| Subjective | 4.25 | 3.75 | 3.75 | 3.75 | 4.00 |
| Mean (Std Dev) | 4.29(0.83) | 4.21(0.76) | 4.13(0.97) | 4.01(0.93) | 3.76(0.93) |

Table 2.1: Assessment scores for the applications [10]

These five nutrition-tracking applications offer a comprehensive approach to dietary management and physical activity tracking. Their user-friendly interfaces contain impressive features that attract users to log their nutrition intake and physical activity daily and help them meet their dietary needs and health goals.

**Foodvisor** and **Lifesum** emphasise holistic health and wellness, integrating meal recommendations and dietary tips based on user activity and health data. **MyFitnessPal** and **LoseIt!** are more traditionally focused on weight loss and calorie tracking but are expanding their functionalities to include broader health metrics. **MyNetDiary** focuses on precise dietary tracking and appeals to users with specific health conditions like diabetes, offering detailed blood glucose tracking and insulin management tools.

All five nutrition tracking applications use large, complex databases containing various food information to provide accurate information about nutrition intake. They enable the user to easily log the food intake using a barcode scanner or manual entry of the consumed

food and its quantity.

Advanced nutrition tracking algorithms can estimate macro and micronutrients for each logged meal and compute the daily total. They also use advanced algorithms to compute the user's dietary needs according to personal information, like sex, age, weight, height, activity level, goals, and dietary preferences.

For a more accurate analysis, these applications can sync with other health applications and devices to track physical activity and overall health and provide a more general overview of the user needs to achieve their goals.

Three applications (**Foodvisor**[16], **MyFitnessPal**[7], and **LoseIt!**[17]) have introduced food image recognition to simplify the meal-logging process. They use Artificial Intelligence and Machine Learning algorithms to identify and classify food items in images. They used large datasets to train their models and received high accuracy in identifying food in images. Users can select from the identified items, search the database for a precise log, enter or adjust the essential suggested weight, or manually log them if the identification is inaccurate. However, none of these applications segment the identified items to emphasise the identification's accuracy and increase the applications' engagement.

All of these applications provide a feature for creating grocery lists. Unfortunately, in all cases, the user must manually log the items on the list. Other applications, like **Mealime**[18] and **Paprika**[19], generate a semi-automatic grocery list based on the user's meal plan or user-selected recipes. However, we could not find a well-known nutrition tracking application that automatically suggests grocery lists based on the food logged in the previous days and the recipes for complex logged meals.

# Chapter 3

# Theoretical Foundations

## 3.1 Supervised & Unsupervised Learning

Machine learning systems can be broadly classified based on three main criteria:

1. **Type and Amount of Supervision During Training**:
    - **Supervised Learning**: The model learns from labelled data.
    - **Unsupervised Learning**: The model finds patterns in data without labels.
    - **Semi-Supervised Learning**: Uses a small amount of labelled data with a large amount of unlabelled data.
    - **Self-Supervised Learning**: Uses automatically generated labels for training.
    - **Reinforcement Learning**: Learns by receiving rewards or penalties based on actions.
2. **Learning Method**:
    - **Batch Learning**: The model simultaneously learns from the entire dataset.
    - **Online Learning**: The model learns incrementally from data as it appears.
3. **Generalisation Approach**:
    - **Instance-Based Learning**: The model memorises and uses specific instances from the training data.
    - **Model-Based Learning**: The model learns a general rule or pattern from the data.

These criteria can be combined to suit different tasks and requirements. [20]

Supervised learning is essential for dividing an image into meaningful segments in image segmentation. This approach uses labelled data to train the model. In this case, each pixel in the image is associated with a specific category.

Supervised learning uses annotated datasets where each pixel is assigned a category. Unlike supervised learning, unsupervised learning derives patterns from data without predefined labels.

This method allows models to learn a function, typically represented as *f*, where

$$Y = f(X; \Theta) \tag{3.1}$$

In this context, *X* represents the input image, *Y* represents the output segmented image, and $\Theta$ represents the parameters that are fine-tuned during the training process.

In supervised learning for image segmentation, the function *f* represents a deep neural network such as a Fully Convolutional Network (FCN) or a U-Net. These networks are designed to take an input image and produce an output image where each pixel is labelled with a class indicating the object or region it belongs to.

The goal during training is to adjust the parameters $\Theta$ of the network to minimise the loss function *L*. In image segmentation, cross-entropy loss is the most commonly used loss function. This loss function is suitable for classification tasks, including pixel-wise classification in image segmentation. The function *L* can be described as:

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(p_{i,c}) \tag{3.2}$$

where $N$ represents the total number of pixels, $y_{i,c}$ is a binary indicator (0 or 1) if the true class of pixel $i$ is $c$, and $p_{i,c}$ is the predicted probability that pixel $i$ belongs to class $c$.

In contrast to unsupervised learning, where techniques like clustering are used to group similar pixels based on properties like texture or colour, supervised learning can precisely identify and outline complex objects within an image. For instance, supervised segmentation could differentiate between humans, cars, and roads in a cityscape, which is extremely useful for tasks such as autonomous driving.

On the other hand, unsupervised learning does not rely on labelled data. Instead, it looks for patterns and structures within the data by grouping pixels based on similar characteristics like colour, texture, or intensity. This method is especially useful when an appropriately labelled dataset is not available.

Some popular techniques for image segmentation using unsupervised learning are clustering methods like k-Means, Gaussian mixture models, and hierarchical clustering. These methods group pixels into clusters that share similar features. Another powerful technique is using autoencoders, which are neural networks that learn to create efficient data representations. Autoencoders can segment images by learning to reconstruct them from compressed versions.

## 3.2 Image Segmentation Techniques

In this subchapter, we will detail some of the most common techniques used in image segmentation.

### Thresholding

Thresholding is a simple and commonly used technique for image segmentation. The aim of this technique is to cover a grayscale image into a binary image. Pixels above a particular threshold value are classified as foreground (e.g., objects), and those below are classified as background. This technique is effective for images with distinct intensity differences between objects and backgrounds.

### Clustering-Based Methods

Clustering algorithms, such as k-Means, partition pixels into clusters based on their features, like colour, texture, or intensity. Each cluster represents a different segment of the image. Clustering methods do not require labelled data, making them suitable for unsupervised segmentation.

- **k-Means Clustering**[21]: This algorithm partitions the image into $k$ clusters by minimising the variance within each cluster. Each pixel is assigned to the nearest cluster centre and then the cluster centres are updated.
- **Gaussian Mixture Models (GMM)**[22]: This probabilistic model assumes that the pixel values are generated from a combination of several Gaussian distributions. The aim is to estimate the parameters of these distributions and assign pixels to the component with the highest probability.

### Edge Detection

Edge detection techniques identify boundaries within an image with significant changes in intensity or colour. These edges often correspond to the boundaries of objects within the image.

- **Canny Edge Detector**[23]: This popular edge detection algorithm uses a multi-stage process for detecting edges in images. It includes noise reduction, gradient calculation, edge tracking by hysteresis, and non-maximum suppression.

### Region-Based Methods

Region-based methods are used to segment an image into regions. These regions are similar according to predefined criteria, such as intensity or texture.

- **Region Growing**[24]: This technique starts with a seed point and grows the region by adding neighbouring pixels that have similar properties.
- **Watershed Algorithm**[25]: Inspired by how water flows across a landscape, this algorithm treats the grayscale image like a topographic surface and finds the lines where water would naturally separate, thus defining region boundaries.

### Autoencoders

Autoencoders represent a type of neural network designed to learn efficient data representations. Typically, their purpose is dimensionality reduction. In the context of image segmentation, autoencoders can segment images by learning to reconstruct them from compressed versions. The network is trained to minimise the discrepancies between the input image and its reconstruction.

- **Convolutional Autoencoders**[26]: These autoencoders use convolutional layers to capture spatial hierarchies in images, making them particularly suitable for image segmentation tasks. The encoder compresses the input image into a latent space representation. At the same time, the decoder reconstructs the image from this representation. The decoder can be modified by training on labelled data to produce segmentation maps.

### Deep Learning-Based Methods

With the advancements in deep learning, more complex and accurate image segmentation techniques have been developed. These methods often rely on Convolutional Neural Networks (CNNs) to perform pixel-wise classification.

- **Fully Convolutional Networks (FCN)**[27]: FCNs replace the fully connected layers of traditional CNNs with convolutional layers that allow the network to produce a spatially structured output, ideal for image segmentation. FCNs can take an input image of any size and produce a segmentation map of the same size.
- **U-Net**[28]: U-Net is a type of FCN with a symmetrical encoder-decoder structure. The encoder path captures the context of the image, while the decoder path enables precise localisation. It uses skip connections to combine low-level features with high-level features. It was originally developed for biomedical image segmentation.

Each technique has strengths and weaknesses and is suitable for different types of images and segmentation tasks. For example, simple thresholding might be enough for images with high contrast between objects and background, while more complex methods like U-Net are better suited for medical imaging, where precise boundaries are essential.

## 3.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a fundamental machine learning component, especially in deep learning tasks. ANNs were inspired by the structure and function of the human brain, where numerous neurons communicate with each other to process information.

A biological neuron comprises a cell body, dendrites, an axon, and synapses. It receives input signals from other neurons through its dendrites, processes them, and transmits an output signal through its axon. Similarly, an artificial neuron, often called a perceptron, receives input signals. It applies a set of weights to these inputs, computes their weighted sum, and then passes the result through an activation function to produce an output. Then, this output can be used as input for other neurons, forming a layered structure known as a neural network.

Figure 3.1 visually represents these processes. The automated neuron receives input signals (denoted as $x_0$, $x_1$, and so on), which are transmitted from one neuron to another via synapses. The synaptic strengths, or weights ($w_1$), control the influence of inputs, and the aggregated signal $Z(\sum_i w_i x_i + b)$ is processed through an activation function $f$, producing the output ($y$) that is sent forward to the next neuron.

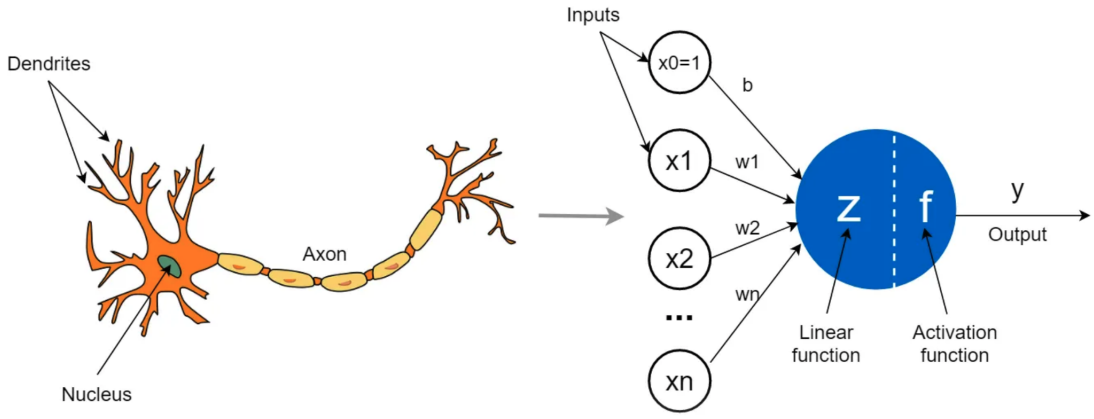



Figure 3.1: Bilogical neuron compared with automated neuron [29]

Artificial Neural Networks are composed of layers of nodes, or neurons, where each layer is fully connected to the next one. The basic structure of an ANN includes:

- **Input Layer**: The initial data is received in this layer. Each neuron in this layer represents a feature or attribute of the data.
- **Hidden Layers**: These layers perform most of the computations required by the network. A network can have one or more hidden layers. In this layer, each neuron receives inputs from all the neurons in the previous layer and sends outputs to all the neurons in the next layer.

- **Output Layer**: This layer produces the final result of the network's computations. Each neuron in this layer represents a possible output or classification.

Figure 3.2 exemplifies the Structure of an Artificial Neural Network.

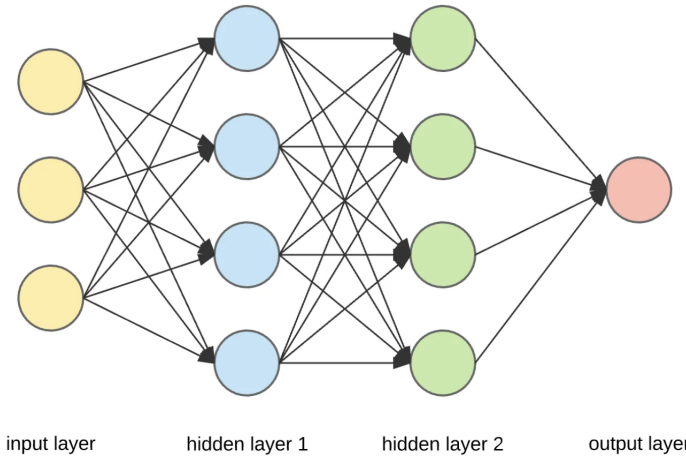



Figure 3.2: ANN structure [30]

Each connection between neurons has an associated weight, adjusted during learning to minimise the error in the network's predictions. The adjustment is typically done using backpropagation, which involves calculating the gradient of the loss function concerning each weight and updating the weights in the opposite direction of the gradient.

The learning process in ANNs involves three main steps:

- **Forward Propagation**: The input data is passed through the network layer by layer to generate an output.
- **Loss Calculation**: The difference between the predicted output and the actual output is calculated using a loss function. Some usual loss functions are mean cross-entropy loss for classification tasks and squared error for regression tasks.
- **Backpropagation**: The network adjusts its weights based on the loss calculated. This step involves propagating the error backwards through the network and updating the weights to minimise the loss.

The network iterates these steps multiple times during training, gradually improving accuracy.

## 3.4 Computer Vision Using Deep Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are deep learning models inspired by the organization of the visual cortex. They are adequate for various computer vision tasks and of-

ten use images as inputs, making them ideal for applications requiring high-level pattern recognition, such as image and video recognition, image classification, object detection, and semantic segmentation. Unlike ordinary neural networks, CNNs are structured to handle high-dimension images efficiently, reducing the need for prior feature extraction.

CNNs consist of several types of layers:

- **Convolutional Layer**
- **Pooling Layer**
- **Fully Connected Layer**

Each layer impacts the network's ability to learn and make predictions.

### 3.4.1 Convolutional Layers

The main building block of a Convolutional Neural Network (CNN) is the convolutional layer. Unlike fully connected layers, which connect every neuron to every other neuron, convolutional layers maintain the spatial structure of the input by connecting each neuron to a small, local region of the input volume. This local region is called the receptive field.

This local connectivity allows CNNs to effectively learn and recognize spatial hierarchies of features in the data. For example, early convolutional layers might detect simple features like edges and textures, while deeper layers can detect more complex features like shapes or even specific objects within the image.

Mathematically, the output of a convolutional layer is computed as follows:

$$z_{i,j,k} = b_k + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_{n'}} x_{i',j',k'} \cdot w_{u,v,k',k}, \text{ with } \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases} \tag{3.3}$$

In this equation, $z_{i,j,k}$ is the output of the neuron located at row $i$, column $j$ in feature map $k$ of the convolutional layer, $b_k$ is the bias term, $x_{i',j',k'}$ is the input from the previous layer, $w_{u,v,k',k}$ are the weights of the convolutional filter, $f_h$ and $f_w$ are the height and width of the filter, respectively, and $s_h$ and $s_w$ are the stride values in the height and width directions, respectively.

In a convolutional layer, multiple filters are used to detect different features. Each filter is a small matrix that slides over the input image, performing the convolution operation described above. The result of applying one filter to the input image is called an activation map or feature map.

Key hyperparameters in a convolutional layer include:

- **Filter Size**: The dimensions of the filters (e.g., 3x3, 5x5). Smaller filters are used to capture fine details, while larger filters can capture broader patterns.

- **Number of Filters**: The number of filters determines the depth of the output feature map. More filters allow the network to detect more features.
- **Stride**: The step size with which the filter moves across the input image. A larger stride results in a smaller output feature map.
- **Padding**: Adding extra pixels around the border of the input image to control the output size. Common padding strategies are 'valid' (no padding) and 'same' (padding such that the output size is the same as the input size)

The dimensions of the output volume can be calculated using the following formulas, where $W_0$ and $H_0$ are the width and height of the input volume, $F$ is the size of the filter, $P$ is the padding, $S$ is the stride, and $D$ is the depth of the filter:

$$W_1 = \frac{W_0 - F + 2P}{S} + 1 \tag{3.4}$$

$$H_1 = \frac{H_0 - F + 2P}{S} + 1 \tag{3.5}$$

The width and height of the output volume are denoted as $W_1$ and $H_1$, respectively. These formulas ensure that all parameters and operations within the convolutional layer are taken into account when determining the size of the output volume.

By adjusting these hyperparameters, CNNs can be adapted to different tasks and datasets, allowing them to learn and extract relevant features from the input data effectively.

### 3.4.2 Pooling Layers

Pooling layers are an essential part of Convolutional Neural Networks (CNNs). They help reduce the size of the feature maps, decreasing the number of parameters and computations in the network. This also makes the network more robust to small translations in the input image.

Max pooling is the most commonly used pooling operation in CNNs. It works by selecting the maximum value from each region of the input feature map. This helps in retaining the most important features while reducing the spatial dimensions.

Max pooling operates in the following way:

1. **Select a Pooling Window**: Choose the size of the window (e.g., $2 \times 2$) and the stride, which is the step size with which the window moves across the feature map.
2. **Slide the Window Across the Feature Map**: Move the window across the input feature map according to the stride.
3. **Compute the Maximum Value**: For each position of the window, find the maximum value within that window.

4. **Output the Result**: The result is a smaller feature map that retains the most significant features from the input.

Figure 3.3 exemplifies the case when we have an input feature map of size $4 \times 4$, and we use a pooling window of size $2 \times 2$ with a stride of 2.

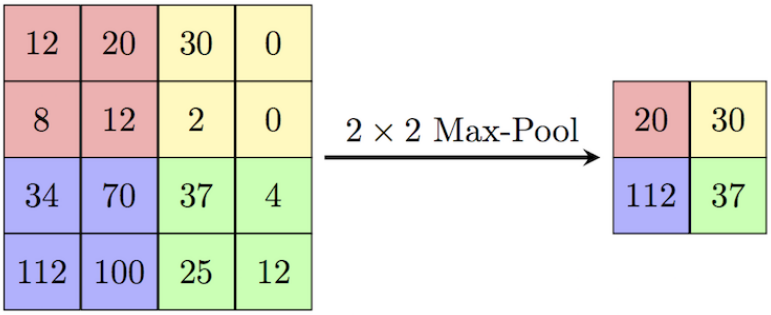



Figure 3.3: Max Pooling Layer [31]

Apart from max pooling, there are other types of pooling:

- **Average Pooling**: Computes the average value within each window.
- **Global Pooling**: Uses a window that covers the entire feature map, reducing it to a single value per feature map.

### 3.4.3 Fully Connected Layers

Fully connected layers are an essential part of Convolutional Neural Networks (CNNs) and many other neural networks. In these layers, each neuron is connected to every neuron in the previous layer, forming a dense network of connections. This setup allows the network to combine all the features it has learned so far to make a final decision.

The key characteristics of Fully Connected Layers are the following:

- **Dense Connections**: Each neuron in a fully connected layer gets input from all the neurons in the previous layer. This means the layer can take into account all the information from the previous layer.
- **Activation Functions**: After gathering inputs, the neuron applies an activation function like ReLU, sigmoid, or tanh. This function adds non-linearity, which helps the network learn more complex patterns.
- **Output**: The output of a fully connected layer is usually a vector. This vector can represent class scores for classification tasks, predicted values for regression tasks, or other outputs depending on the task.

Fully connected layers are usually found at the end of a CNN. They gather all the features identified by the convolutional and pooling layers and use them to make the final prediction. These layers are especially common in image classification networks, where they help map the extracted features to the final class labels.

### 3.4.4 Convolutional Neural Networks Architectures

Typical CNN architectures consist of a series of convolutional and pooling layers, followed by one or more fully connected layers.

Figure 3.4 exemplifies a typical Convolutional Neural Network Architecture.

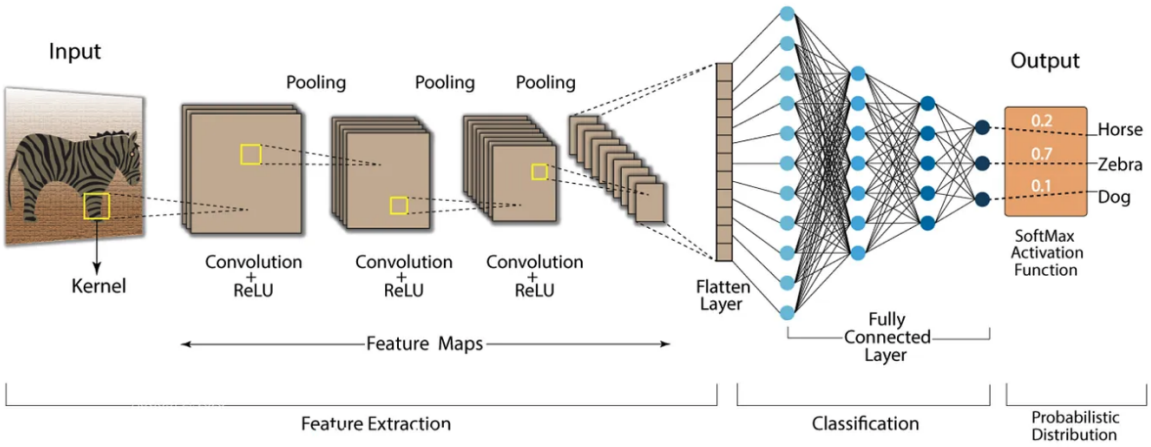



Figure 3.4: CNN Architecture [32]

In this subchapter, we will discuss some key architectures that have been influential in the field of computer vision.

**LeNet-5**, developed by Yann LeCun in 1998 [33], is one of the earliest CNN architectures designed for handwritten digit recognition. It consists of two sets of convolutional and pooling layers, followed by three fully connected layers.

**AlexNet** [34], which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, significantly advanced the field by using deeper networks, ReLU activations, and dropout for regularization. It demonstrated the power of CNNs on large-scale datasets.

**GoogLeNet**[35], the winner of the ILSVRC 2014, introduced the Inception module, which allows for more efficient computation by combining multiple convolutions with different filter sizes into a single layer. This architecture significantly reduced the number of parameters compared to previous models.

**ResNet**, introduced by Kaiming He in 2015 [36], addresses the degradation problem in deep networks by using residual connections. These connections allow the network to learn identity mappings, which help train very deep architectures. ResNet models achieved state-of-the-art results in image classification, detection, and segmentation tasks.

### 3.4.5 Object Detection and Semantic Segmentation

Object detection and semantic segmentation are two essential tasks in computer vision, each serving unique purposes in helping machines understand and interpret images.

**Object Detection** focuses on identifying and localizing objects within an image. This task involves recognizing what objects are present and determining their precise locations, typically through bounding boxes. Object detection is widely used in applications such as autonomous driving, surveillance, and any scenario where understanding the position of objects is essential. It is a combination of image classification and localization, where the model predicts the classes of objects and their coordinates within the image.

Figure 3.5 exemplifies a case of using object detection.

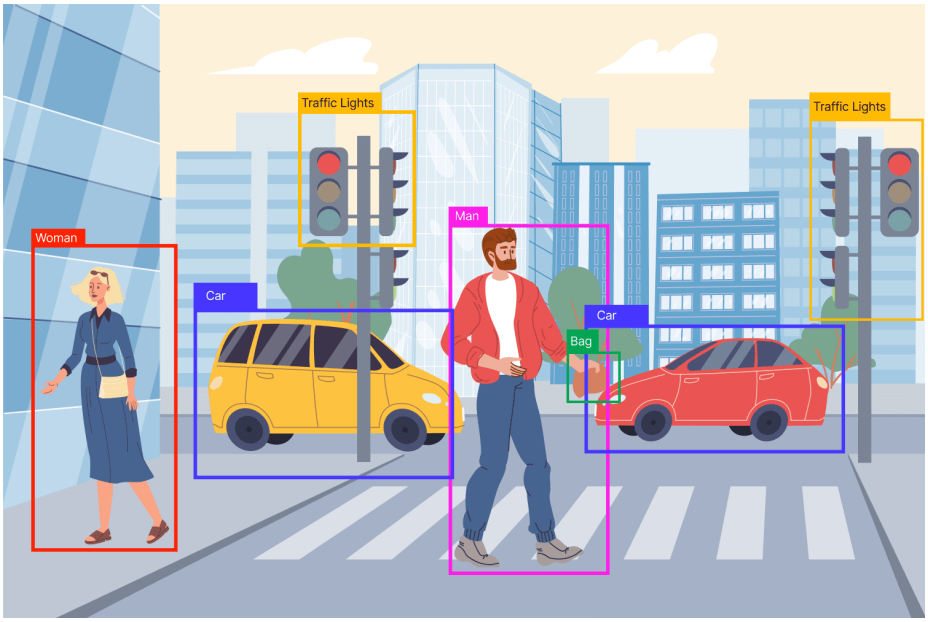



Figure 3.5: Object detection representation [37]

**Semantic Segmentation**, on the other hand, aims to label each pixel in an image with a corresponding class. This process divides the image into meaningful parts and classifies each part into one of the predefined categories. Semantic segmentation is essential for tasks requiring a fine-grained understanding of the scene, such as medical imaging, satellite image analysis, and image editing. Unlike object detection, which provides bounding boxes around objects, semantic segmentation provides a much more detailed and comprehensive understanding by classifying every pixel.

Figure 3.5 exemplifies a case of using semantic segmentation.

While object detection focuses on identifying the presence and location of objects, semantic segmentation provides a pixel-level classification, offering a richer and more nuanced understanding of the image.

The introduction of **Fully Convolutional Networks (FCNs)** represented a real improvement for semantic segmentation. Introduced by Long, Shelhamer, and Darrell in 2015[39],

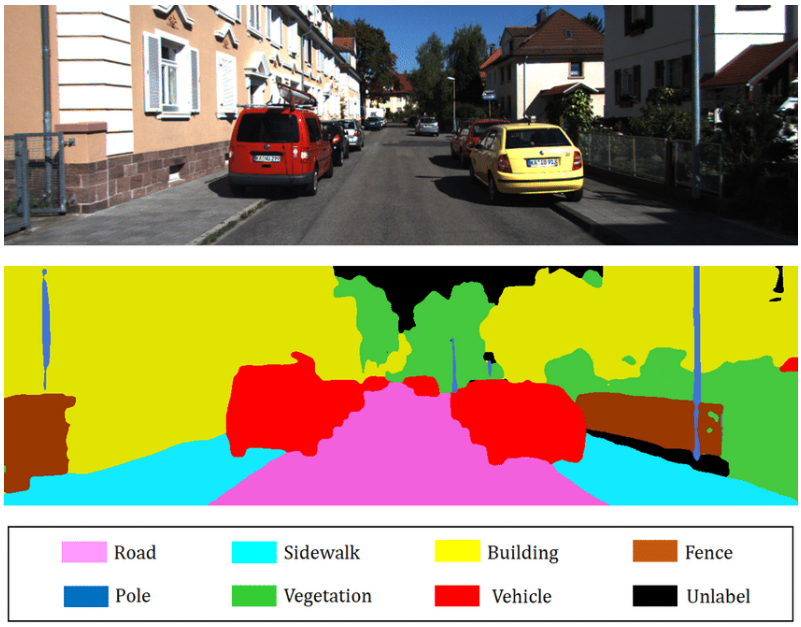



Figure 3.6: Semantic segmentation representation [38]

FCNs replace the fully connected layers found at the end of traditional Convolutional Neural Networks (CNNs) with convolutional layers. This change allows the network to take an input image of any size and produce an output with corresponding spatial dimensions, making pixel-wise predictions. The key innovation is using deconvolution layers (transposed convolutions) to upsample the feature maps back to the original image size, enabling precise segmentation.

**R-CNN (Region-based Convolutional Neural Networks)**[40] initially tackled object detection but set the stage for advances in segmentation. The idea was to generate region proposals and classify each one using CNN. Although effective, R-CNNs were slow because each region proposal had to be processed independently.

**Faster R-CNN**, introduced by Ren et al.[41], improved R-CNN by adding a Region Proposal Network (RPN) that shares convolutional features with the detection network, speeding up the process significantly. The RPN generates region proposals, which are then classified and refined. This end-to-end training approach boosts both speed and accuracy, making it a foundation for more advanced networks.

**Mask R-CNN**, developed by He et al. [42], builds on Faster R-CNN by adding a branch for predicting segmentation masks for each Region of Interest (RoI). This architecture introduces RoIAlign, a technique that aligns the extracted features with the original image more precisely, improving mask predictions. Mask R-CNN is versatile and excels at instance segmentation, where each object instance is identified and segmented separately.

**Cascade R-CNN**[43] enhances object detection by addressing the issue of IoU (Intersection over Union) threshold mismatches during training and inference. This architecture uses multiple detection stages with progressively increasing IoU thresholds, refining the detec-

tions at each stage. The result is improved accuracy and robustness, especially for detecting objects of different sizes and shapes.

**Hybrid Task Cascade (HTC)** takes the idea of cascaded refinement further by integrating object detection and instance segmentation into a single framework. Introduced by Chen et al.[44], HTC performs these tasks in an interleaved manner across multiple stages, progressively enhancing both detections and segmentation masks. This approach is particularly effective for complex scenes with many objects and overlapping instances.

These architectures represent significant advancements in semantic segmentation and object detection, building on the successes of previous models and addressing their limitations.

# Chapter 4

# Related Work

## 4.1 Overview

Food image classification and segmentation represent two challenging and exciting tasks in food computing with various applications, especially in health and nutrition. The recognition and segmentation of food items in images can help estimate the nutrition intake, focusing on tracking dietary habits, especially for individuals with specific dietary requirements or health conditions like diabetes and obesity. [9]

Compared with manual tracking, automated food recognition tools can quickly provide detailed nutrition information from an image and provide more accurate and user-friendly dietary monitoring. It can also facilitate personalising nutrition plans based on more precise dietary tracking. Integrating these automated food recognition tools can improve the user experience through an application and help researchers better analyse the population's dietary patterns and their health impact, identifying trends and making associations between diets and health outcomes.

Food classification represents the ability to precisely identify the type of food in images, while segmentation outlines each food item and even the containing ingredients. Starting from identifying singular food items in an image, we can now discuss multiple food item identification with the evolution of deep learning.

However, the complexity of these tasks comes from various challenges. Food can appear in various ways due to the cooking methods, presentation, or ingredients used. At the same time, in many dishes, ingredients overlap or are mixed, making it difficult for segmentation models to distinguish between food items accurately. [45]

The quality of the datasets can also impact the accuracy of the models. In this case, high-quality, large-scale datasets are essential. The diversity and representation of real-world conditions to avoid biases are also important. [46]

The context of the images taken can also impact the models' results. Images taken under varying lighting conditions, different camera angles, and varying resolutions add complexity

to the classification and segmentation tasks. [46]

Finally, distinguishing between similar food items, such as different types of bread or pasta, can impede accurate food identification. This task requires fine-grained classification capabilities to allow the models to capture subtle differences in appearance and texture. [47]

This section outlines the recent advancements in food recognition, focusing on the progress of image classification and segmentation techniques and providing a detailed review of some relevant works.

## 4.2 Existing Datasets

As mentioned above, developing accurate food image classification and segmentation models depends on using robust, high-quality datasets. Large, real-world, and non-biased datasets are essential for these tasks. This section reviews notable datasets for food recognition and their advantages and disadvantages.

The introduction of novel datasets like UEC-FOOD100 [48], UEC-FOOD256 [49], UEC-FoodPix [50], and UEC-FoodPixComplete [51] established a notable advancement in the use of machine learning for food image recognition.

The UEC-FOOD100 dataset contains 100 categories of Japanese food and 14361 images. UEC-FOOD256 expands this to 256 categories and 25088 images.[47] The primary focus of both datasets is the food classification task, assuming that each image contains a single food item and aiming to predict the food class from the image. They include bounding box annotations, facilitating object detection. [46]

The ETH Food-101 dataset consists of 101000 images divided into 101 food categories, each containing 1000 images. It supports research in food recognition and is particularly useful for tasks like dish classification. [52]

Food2k [53] consists of over a million images across 2000 food categories. It represents one of the largest collections available for food image analysis. Despite its size, the dataset may face challenges related to image consistency and quality, which may impact the model training.[47]

Recipe 1M [54] is an extensive dataset containing over 900000 images and 1 million recipes. It is used in multi-modal learning between images and recipes. A larger dataset, Recipe1M+[55], was constructed based on Recipe1M. It contains more than 13 million food images.

The UEC-FoodPix and UEC-FoodPixComplete datasets introduce the Food Image Segmentation task. They consist of 102 food categories, 10000 images with multiple food items per image, and an associated pixel-wise segmentation mask. [45]

Mohanty et al.[46] introduced the MyFoodRepo-273 dataset, which contains 273 food categories, 24119 images, and 39325 segmented polygons, with at least 35 annotations per class. The data available through the benchmark was collected through the MyFoodRepo

application[56] from numerous users who tracked their nutrition intake between July 2018 and June 2020. The users were able to anonymise the private pictures and make them public for research purposes. This dataset was used during the Food Recognition Challenge, founded by AIcrowd.[57] At the time of this study, the dataset has evolved, containing 498 food categories and a total of 43962 images with 95009 labelled objects. [58]

Developed by Wu et al.[45], FoodSeg103 includes 7118 images with 103 ingredient classes, while FoodSeg154 extends to 9490 images and 154 ingredient classes, including diverse and challenging Asian food images. These datasets are designed to facilitate fine-grained segmentation tasks with detailed pixel-wise annotations for various ingredients.

## 4.3 Technologies Used For Food Recognition and Their Results

In recent years, advancements in food recognition have been directly influenced by the rapid evolution of deep learning techniques. Various researchers have explored different architectures and datasets to improve the accuracy of food recognition systems. This section details several key experiments and their results.

Aguilar et al.[59] introduced deep learning in food image recognition by applying YOLOv2 and Dark-Net19 to the UNIMIB2016 dataset. Their approach aimed to enhance the precision of food detection and recognition in numerous images. The use of YOLOv2, known for its ability to detect objects in images in real-time, combined with DarkNet-19, allowed them to achieve a precision of 0.841. This result represented a significant improvement in food recognition and showcased the potential of deep learning models to identify and localise food items in images accurately.

Ye et al.[60] conducted Mask R-CNN experiments with MobileNet and ResNet backbones for food segmentation. They selected ten food categories from the MS COCO dataset to perform a comparative analysis of different deep-learning architectures. Their study revealed that Mask R-CNN with ResNet significantly outperformed Multi-SVM in food segmentation tasks. This experiment enhanced the ability of deep learning models, especially Mask R-CNN, to handle complex segmentation tasks by leveraging ResNet's robust feature extraction capabilities.

Freitas et al.[61] experimented with multiple deep learning architectures, including Mask R-CNN, DeepLab v3, SegNet, and ENet, on a proprietary dataset of Brazilian food items. Their results demonstrated that Mask R-CNN outperformed the other approaches, achieving a mean average precision (mAP) of 0.87. In contrast, the other methods scored below 0.79. This study underscored the effectiveness of Mask R-CNN in accurately segmenting and recognising food items, particularly in a dataset with diverse and complex food types.

Mohanty et al.[46] conducted extensive research to improve food recognition using deep

learning. They focused on creating and utilising a comprehensive benchmark dataset as part of the Food Recognition Challenge, maintained by AICrowd [57]. We will detail their experiments and results below.

The dataset used was MyFoodRepo-273, which contains images taken by real users, providing a diverse and unbiased sample.

The benchmark was divided into four rounds, each increasing in complexity:

- Round 1 focused on 40 food categories with 5545 training images. The best model achieved a mean average precision (mAP) of 0.573 and a mean average recall (mAR) of 0.831.
- Round 2 expanded to 61 categories with 7949 training images. The top model reached a mAP of 0.633 and a mAR of 0.886.
- Round 3 covered 273 categories with 24119 training images. The best submission had a mAP of 0.551 and a mAR of 0.884.
- Round 4 maintained the 273 categories but introduced a new test set. The best model achieved a mAP of 0.568 and a mAR of 0.767.

As the number of food categories increased, the task became more challenging, reflected in the mAP scores.

Various deep-learning models for segmenting and classifying food items were tested during the challenge. Some notable results were obtained by using Mask R-CNN and Hybrid Task Cascade (HTC) models.

**Mask R-CNN**: They used different versions of this model with backbones like ResNet50, ResNet101, and ResNeXt101. The best results achieved by combining techniques such as multi-scale training, weighted loss, and data augmentation are presented in the table 4.1:

| **Model** | **mAP** | **mAR** |
|---|---|---|
| ResNet50 | 0.506 | 0.809 |
| ResNet101 | 0.523 | 0.817 |
| ResNeXt101 | 0.535 | 0.825 |

Table 4.1: Performance of Mask R-CNN models on food recognition tasks.

**Hybrid Task Cascade (HTC)**: This model performed even better, with the results presented in the table 4.2:

| **Model** | **mAP** | **mAR** |
|---|---|---|
| ResNet50 | 0.525 | 0.861 |
| ResNet101 | 0.539 | 0.867 |

Table 4.2: Performance of Hybrid Task Cascade models on food recognition tasks.

Aggressive data augmentation techniques were used to make the models more robust and prevent overfitting. These included resizing images, applying random rotations, flips, shifts, scaling, and adding visual effects like blurring and noise. Colour changes were kept to a minimum to preserve the subtle differences that distinguish similar food items.

The benchmark utilized mAP and mAR metrics to evaluate model performance. mAP was calculated based on the intersection over union (IoU) of segmentation masks, aligning with the standards set by the PASCAL VOC Challenge. The best-performing models from the benchmark were deployed via an API to assist human annotators in the MyFoodRepo app, significantly improving annotation efficiency and accuracy.

Wu et al.[45] have significantly contributed to the field of food image segmentation by developing comprehensive datasets and leveraging multi-modality knowledge to enhance model performance. Their work involved creating the FoodSeg103 and FoodSeg154 datasets, developing a novel pre-training approach called ReLeM(Recipe Learning Module), and conducting extensive experiments to validate their methods.

They constructed two datasets: **FoodSeg103** and its extension **FoodSeg154**. FoodSeg103 includes 7118 images annotated with 103 ingredient classes, while FoodSeg154 adds 2372 images of Asian food, bringing the total to 9490 images with 154 ingredient classes. Each image in these datasets is annotated with pixel-wise segmentation masks, providing a fine-grained understanding of the food items.

They introduced a novel pre-training method called **ReLeM** (Recipe Learning Module), which integrates recipe information into the visual representation of food images. This approach uses multi-modality knowledge transfer, where visual and textual data (from recipes) are combined to improve the model's ability to recognise ingredients despite variations in appearance due to different cooking methods.

**ReLeM** leverages two types of encoders:

- **Text Encoder**: Processes ingredient labels and cooking instructions using LSTM or transformer-based models.
- **Vision Encoder**: Extracts visual features from food images, using models like ResNet-50 or ViT-16/B.

The goal of ReLeM is to align visual representations of the same ingredient across different dishes by incorporating textual information from recipes.

They validated ReLeM by incorporating it into three semantic segmentation models: **CCNet**, **FPN**, and **SeTR**. They tested each model with and without ReLeM, using both LSTM-based and transformer-based text encoders.

**Baseline Models**:

1. **CCNet**: A dilation-based model using criss-cross attention to reduce computation costs.

2. **FPN**: Integrates feature maps from different layers to enhance shallow-layer representations.

3. **SeTR**: Uses transformer-based attention mechanisms to capture contextual information.

**ReLeM-Variants**:

- ReLeM-CCNet
- ReLeM-FPN
- ReLeM-SeTR

The experiments on FoodSeg103 demonstrated significant improvements in segmentation performance when incorporating ReLeM. The results showed that ReLeM enhanced convolutional and transformer-based models, with LSTM-based ReLeM generally performing better than transformer-based ReLeM.

| **Model** | **mIoU** | **mAcc** |
|---|---|---|
| CCNet | 35.5 | 45.3 |
| ReLeM-CCNet (LSTM) | 36.8 | 47.4 |
| ReLeM-CCNet (Transformer) | 36.0 | 46.5 |
| FPN | 27.8 | 38.2 |
| ReLeM-FPN (LSTM) | 29.1 | 39.8 |
| ReLeM-FPN (Transformer) | 28.9 | 39.7 |
| SeTR | 41.3 | 52.7 |
| ReLeM-SeTR (LSTM) | 43.9 | 57.0 |
| ReLeM-SeTR (Transformer) | 43.2 | 55.7 |

Table 4.3: Performance of different models on FoodSeg103.

The team also tested their models on the Asian food subset of FoodSeg154 to evaluate cross-domain adaptability. They fine-tuned the models on the new subset and observed that ReLeM consistently outperformed baseline models, demonstrating its effectiveness in handling diverse food appearances.

These studies collectively highlight the advancements and challenges in food recognition. The experiments conducted by Aguilar et al., Ye et al., Freitas et al., Mohanty et al., and Wu et al. demonstrate the effectiveness of various deep learning architectures in improving the accuracy of food image analysis systems.

# Chapter 5

# Food Recognition Modelling

This section of the thesis discusses the modelling process for the food recognition component of the system, focusing on the data preparation and enhancement techniques and the model performance and optimisation.

## 5.1 Dataset

For this research, we used the 'Food Recognition 2022' dataset, released by AIcrowd [57], in 2022 and available on Kaggle [58]. The data available through the benchmark was collected through the MyFoodRepo application [56] from numerous users who tracked their nutrition intake. This way, the dataset contains a large variety of food images taken in real-world conditions. This represents an advantage compared with the large variety of food images taken in laboratory conditions, under specific light and angles, or the existing social media ones containing perfect-looking, unrealistic meals.

The 'Food Recognition 2022' dataset is particularly well-suited for tasks involving food identification and segmentation since it includes a wide spectrum of food items commonly consumed worldwide. The dataset is diverse, ranging from basic items like 'Water' and 'Bread' to more complex dishes and ingredients such as 'Pizza, ' Margherita', and 'Chicken curry'.

The dataset comprises 498 distinct food categories, containing a training set of 39962 images of food items, with 76491 annotations, each category represented by a varying number of images. At the same time, the validation set contains 498 food categories, 1000 images and 1830 annotations, and the test set contains only 3000 images. In the case of the training and validation datasets, each image is accompanied by annotations in the COCO format detailing the food items, representing the necessary labels for tasks like classification and segmentation. The dataset images and annotations distribution is detailed in Figure 5.1

The distribution of images across these categories is uneven, with some categories being significantly more represented than others. For instance, the category 'Water' contains 2928

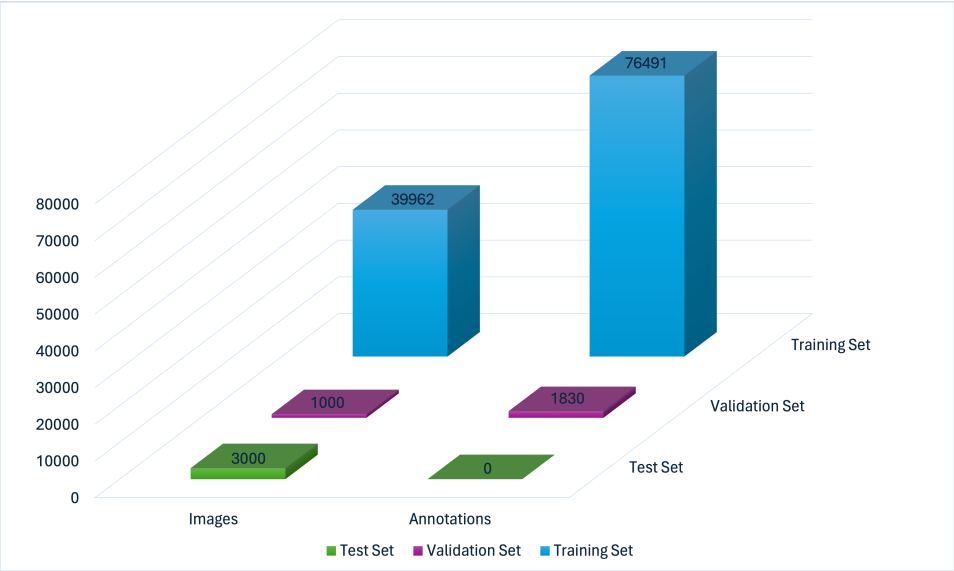



Figure 5.1: Images and annotations per set distribution

images, making it the most represented category in the dataset. Other highly represented categories include 'Salad, leaf / salad, green' with 2002 images and 'Bread, white' with 1891 images. Due to their large number of images, these categories present more chances to be well-learned by the model, leading to higher accuracy in recognising them.

On the other hand, there are categories with significantly fewer images, such as 'Pecan nut' with 27 images and 'Blackberry' with 19 images. The underrepresentation of these categories presents a challenge for model training, since the model may not have sufficient data to learn the features associated with these foods.

Figures 5.2, 5.3, 5.4 provide a summary of the number of images available for some of the categories within the dataset:

## 5.2 Data Preparation and Enhancement

Data preparation is essential in developing a machine learning model, particularly in food image recognition and segmentation tasks. The dataset needs to be processed and enhanced to achieve reasonable accuracy. This subchapter discusses the strategies and techniques employed to prepare, augment, and enhance the dataset, ensuring the accuracy of the trained model.

### Data Augmentation

Data augmentation plays a significant role in increasing the diversity of the training dataset without the need for collecting new data. It involves applying various transformations to the existing images, thus artificially expanding the dataset and helping the model generalize better to unseen data. However, given the nature of food images, certain considerations must

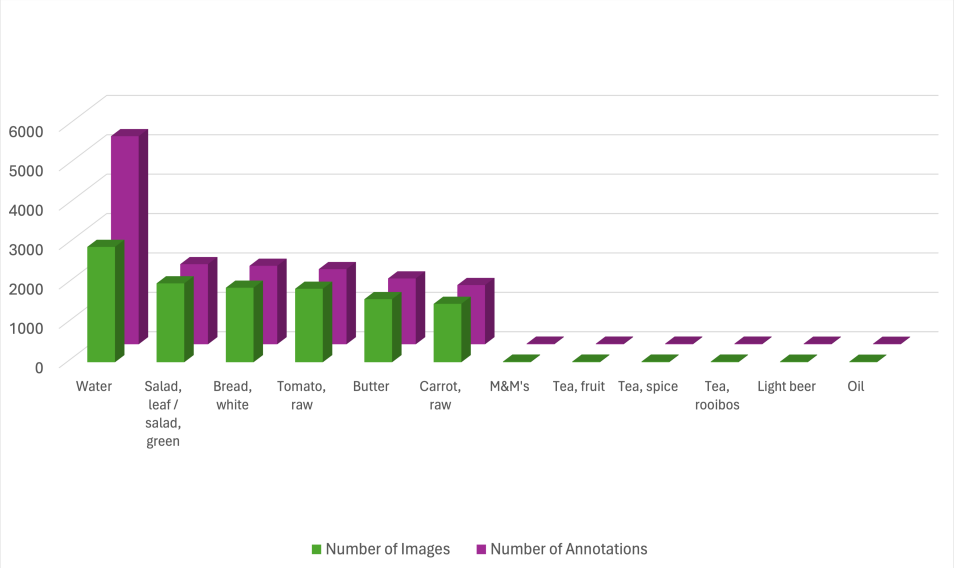



Figure 5.2: Images and annotations distribution on the top 6 categories and the lowest 6 categories sorted by the number of images

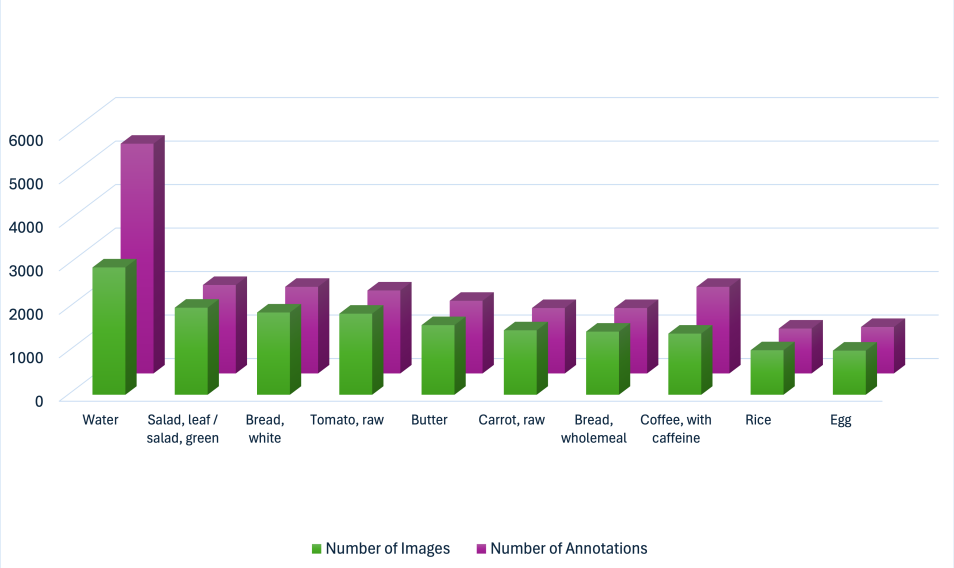



Figure 5.3: Images and annotations distribution on the top 10 categories sorted by the number of images

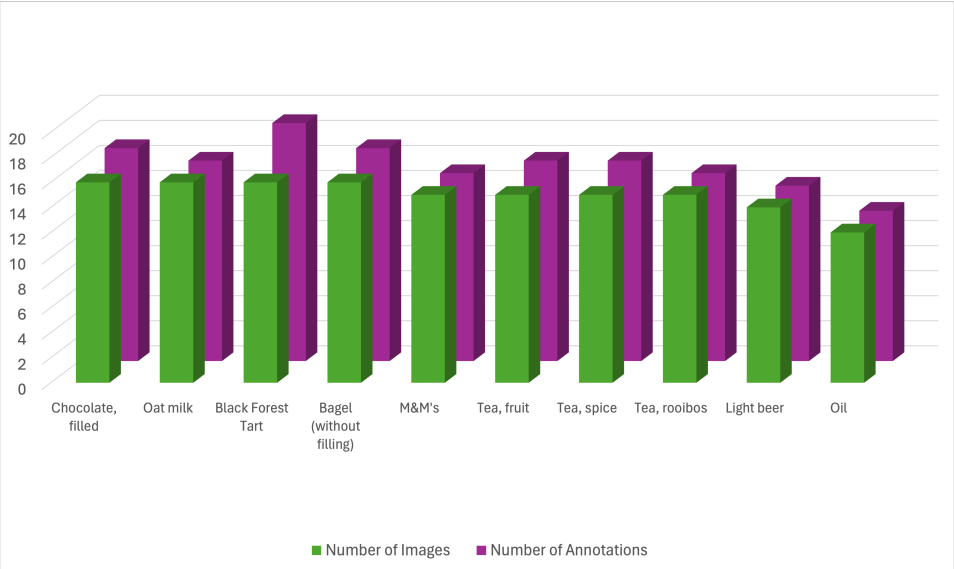



Figure 5.4: Images and annotations distribution on the lowest 10 categories sorted by the number of images

be taken into account during augmentation to maintain the integrity of the data.

**Colour Augmentation**: Colour augmentations were applied with caution since the colour of the food items directly correlates with the food type and its preparation state. Therefore, minimal hue, saturation, and brightness adjustments were employed to avoid distorting the food's natural appearance.

**Rotation and Spatial Augmentation**: The rotation of images could simulate different viewpoints. However, excessive rotation could result in unnatural representations of food items, especially those with clear orientation, like beverages in cups. Therefore, the rotation angle was restricted to a maximum of 5 to 10 degrees. Additionally, the corresponding segmentation masks were rotated simultaneously with the images to preserve the accuracy of the annotations.

**Flipping**: Horizontal flipping was applied as an augmentation technique, which is particularly useful since most food items do not have a specific left or right orientation. However, vertical flipping was avoided as it could result in unnatural representations of many food items, such as beverages in cups.

### Data Splitting

The 'Food Recognition 2022' dataset comprises three primary subsets: the training set, the validation set, and the test set. The splitting of data into these subsets was performed with careful consideration of the dataset's distribution.

The test set, containing 3000 images, included no annotations. Understandably, this test set was provided without annotations as part of an AI competition. However, in the context of this research, it represented an issue, as the test dataset could not be used to properly

evaluate the model and compute the evaluation scores. Consequently, the test set was kept separate from the model and used only for visual checks during demos.

The validation set contained only 1000 images with 1830 annotations, which needed to be improved for model tuning and hyperparameter optimisation. This way, a portion of the training data was reallocated proportionally to the validation set, ensuring that the diversity and representation of categories in the training were mirrored in the validation set.

Figure 5.5 presents the train and validation set distribution after splitting the data.


Figure 5.5: Images and annotations per set distribution

## 5.3 Model Selection and Design

In this research, the model selection process involved exploring various architectures suitable for instance segmentation. Since the model needs to accomplish the classification and segmentation tasks, a convolutional neural network (CNN) architecture was chosen because it was designed to handle image data.

Among the available architectures, the Mask R-CNN architecture was selected for its proven efficacy in tasks involving object detection and instance segmentation. The configuration of the model was carried out using the Detectron2[62] library, developed by Facebook AI Research, designed specifically for computer vision tasks. The Mask R-CNN model with a ResNet-50 backbone and a Feature Pyramid Network (FPN) was selected for its balance between computational efficiency and performance. The ResNet-50 architecture offers a deep convolutional network that can extract rich feature representations from images, while the FPN can be useful in detecting objects at different scales.

## 5.4 Training and Validation

The training process for the Mask R-CNN model was conducted on the prepared dataset, which included 37960 images with 72663 annotations across 498 food categories. The Detectron2[62] library was employed for managing the training proces.

The training was carried out over 10000 iterations, with checkpoints saved every 1000 steps. This checkpointing strategy ensured that the model's progress could be monitored, and training could be resumed from a saved state in case of interruptions. The loss function used during training combined the losses from classification, mask prediction, and bounding box regression, reflecting the multi-task nature of the problem.

After the training was completed, the model's final weights were saved, and its performance was evaluated on the validation set. The validation dataset contained 3002 images and 5658 annotations. This evaluation involved generating predictions for the images in the validation set and comparing them with the ground truth annotations. The model's predictions were visualised using the Detectron2 visualiser.

### 5.4.1 Performance Evaluation

To evaluate the performance of the machine learning model in the semantic segmentation task, we use a set of metrics that can capture the model's accuracy and efficiency, like mean average precision (mAP).

To define the mean average precision (mAP), we need to detail the intersection over union (IoU), precision, and recall concepts.

Intersection over Union (IoU), represented in Figure 5.6, is a metric used to evaluate the accuracy of the model's segmentation capabilities. IoU compares the overlap between the predicted segmentation mask and the ground truth mask. This is done by dividing the area of intersection by the area of the union. This metric directly reflects how well the model can delineate the boundaries of food items within an image. The IoU score is directly proportional to the segmentation performance, with the model closely matching the ground truth annotations.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

Precision 5.1 represents a metric that reflects the ability of the model to identify food items without erroneously classifying non-food items or incorrect categories as food. It computes the ratio between the number of true positive (TP) identifications made by the model relative to the total number of positive identifications, both true positives (TP) and false positives (FP). Positive refers to the class on which we compute the precision score, so the true positive means, in general, a correct prediction, and the false positive represents a prediction of that class when the class is not actually present. In the case of image segmen-

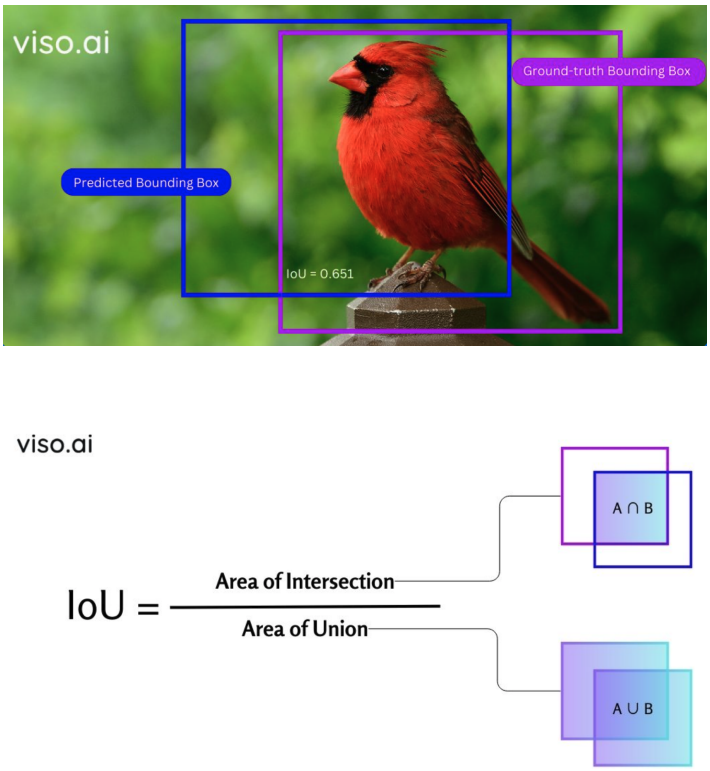



Figure 5.6: IoU definition [63]

tation, a true positive depends on the IoU score. High precision indicates the accuracy of the model, minimising the occurrence of false positives. [63]

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

Recall 5.2 is another metric that determines the ratio between the true positive identifications made by the model and the total number of actual positives (i.e., all instances of the item that should have been identified). The recall formula is similar to the precision one. Still, it represents the ratio between the true positive (TP) and the sum of the true positive (TP) and the false negative (FN). A false negative prediction (FN) means the model does not predict a class when it should. A model with high recall successfully identifies most items present, minimising the number of false negatives. [63]

The mean Average Precision (mAP) is a widely recognised performance metric in object detection and segmentation tasks. It offers a comprehensive evaluation by considering both the precision and recall across various thresholds of Intersection over Union (IoU). First, mAP is computed by calculating the Average Precision (AP) for each class. This involves determining the precision-recall curve for that class. This curve is generated by varying the confidence threshold for what constitutes a positive detection. At each threshold, precision and recall are recalculated, and the corresponding points are plotted on a graph, which contains the recall on the horizontal axis and precision on the vertical axis. Average precision is computed by calculating the area under the curve (AUC). The mean average precision rep-

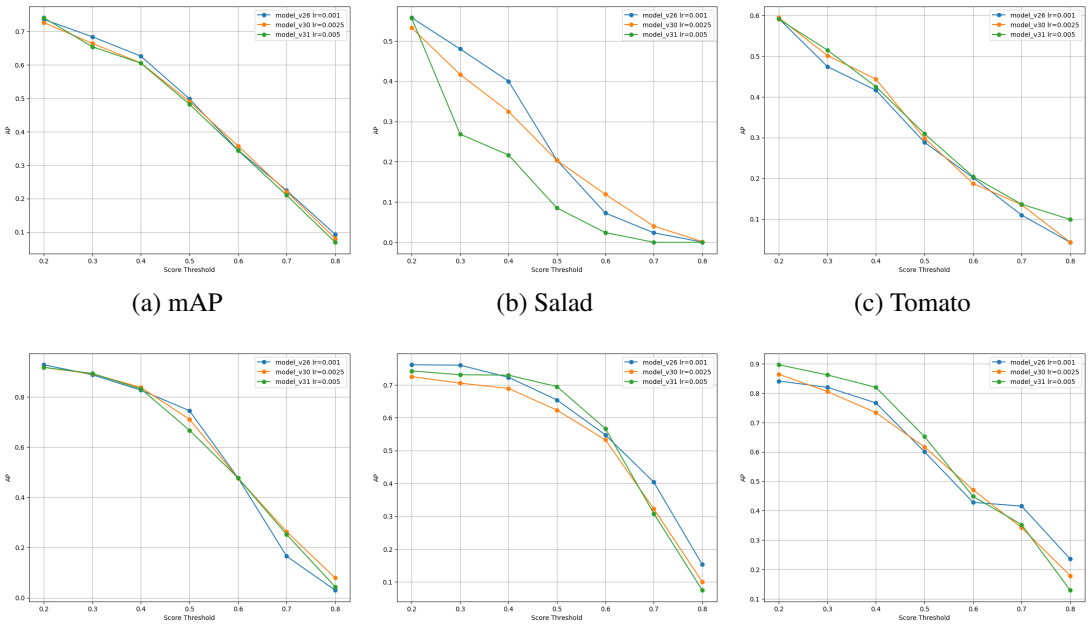
resents the average of AP values computed for each class. [63] We use the COCO evaluator within the Detectron2 framework to compute mAP.

### 5.4.2 Hyperparameter Tuning

Hyperparameter tuning is the process of selecting a set of hyperparameters for the model to optimise its performance. Hyperparameters are set before training and control various aspects of the training process and the architecture of the model. Hyperparameter tuning can significantly impact the accuracy of the model and the convergence speed.

Several hyperparameters were tuned to increase the performance of the food recognition model trained using the Detectron2 framework. We will analyse some of them and their impact on the model's results. Since the dataset consists of 498 classes and is unbalanced, as mentioned previously, the mean average precision is highly influenced by the low scores obtained for the classes with few images and annotations. Because of that, we will focus on analysing the average precision for the top five classes by the number of corresponding images in the dataset. We will analyse the results for the bounding boxes and the segmentation masks.

First, we consider the learning rate. Changing the model parameters in response to the estimated error is determined by the learning rate each time the model weights are updated. We experimented with learning rates of 0.001, 0.0025, and 0.005. The results are represented in Figures 5.7 and 5.8.

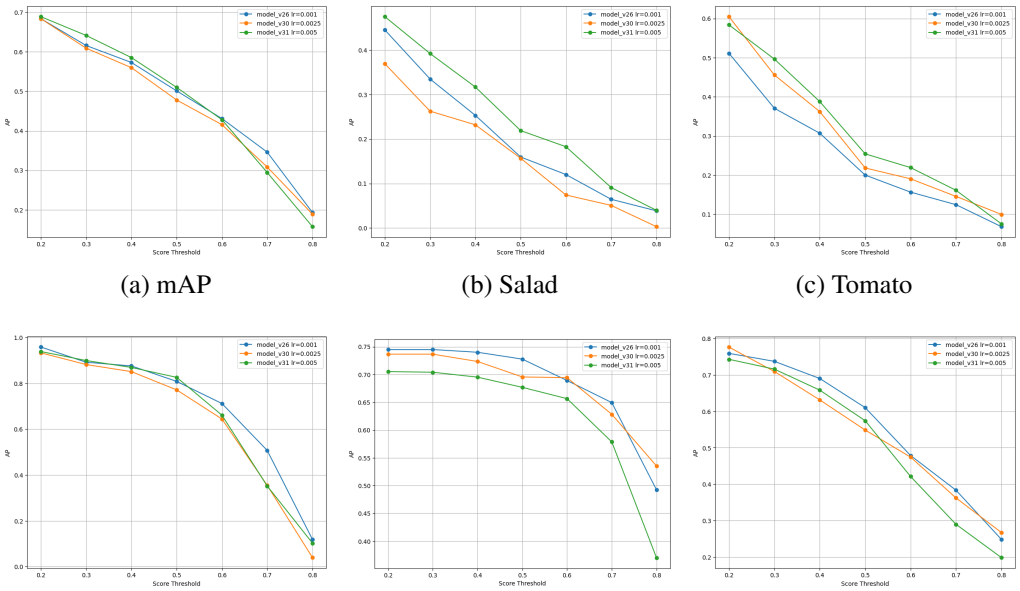



(a) mAP (b) Salad (c) Tomato

(d) Bread (e) Butter (f) Water

Figure 5.7: mAP and AP variations for different learning rates on bounding boxes

Then, we analysed the Batch Size, which represents the number of images processed




(a) mAP (b) Salad (c) Tomato

(d) Bread (e) Butter (f) Water

Figure 5.8: mAP and AP variations for different learning rates on segmentation masks

before the model parameters are updated. Considering the GPU's memory limitations, we tested batch sizes 4, 8, and 16 as presented in Figures 5.9 and 5.10.

We also adjusted the number of iterations. It represents the frequency of the model when processing the entire training dataset. We experimented with 1000, 5000, and 10000 iterations, as shown in the Figures 5.11 and 5.12

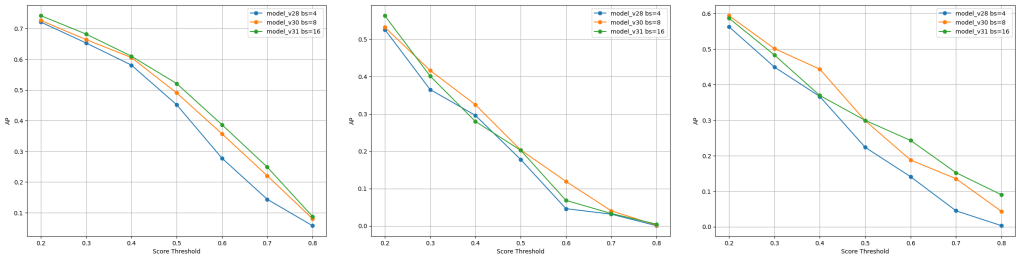
### 5.4.3 Final Results

After tuning the hyperparameters, the model was evaluated on the validation set. The model used a Mask R-CNN model with a ResNet-50-FPN backbone and achieved a Mean Average Precision (mAP) of 0.274 on segmentation masks, indicating a moderate level of accuracy in identifying and segmenting food items, considering the biased dataset.

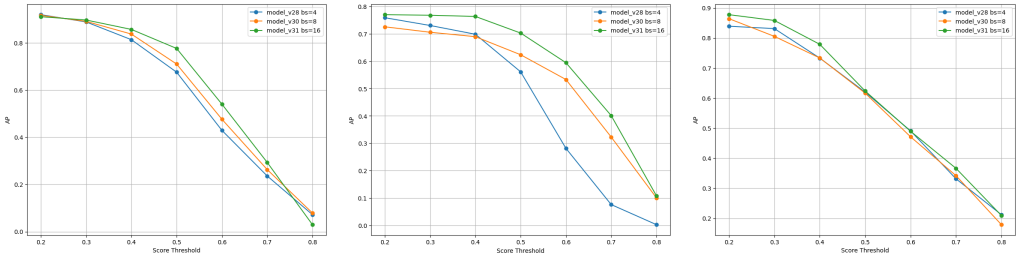
The table 5.1 presents the first three hyperparameter tuning configurations that scored the best results. The chosen configuration (Learning Rate: 0.001, Batch Size: 16, Iterations: 5000) provided the best mean average precision. The Average Precision results for the first five food categories sorted by the number of images in the training dataset for the same configuration are presented in Table 5.2

Table 5.1: Model Configurations

| Learning Rate | Batch Size | Iterations | Overall mAP |
|---|---|---|---|
| 0.0001 | 16 | 5000 | 0.274 |
| 0.0001 | 16 | 10000 | 0.268 |
| 0.0001 | 16 | 1000 | 0.243 |

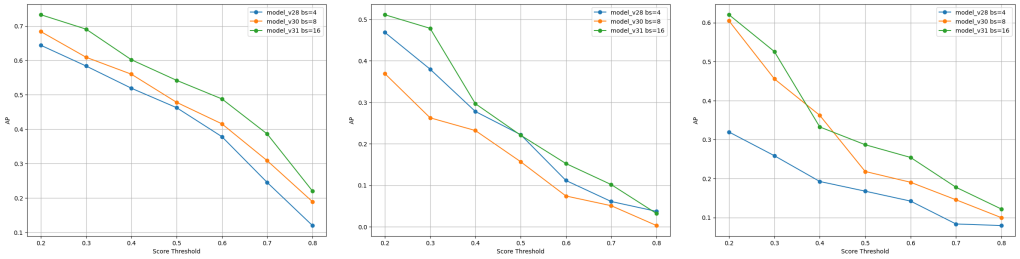



(a) mAP (b) Salad (c) Tomato

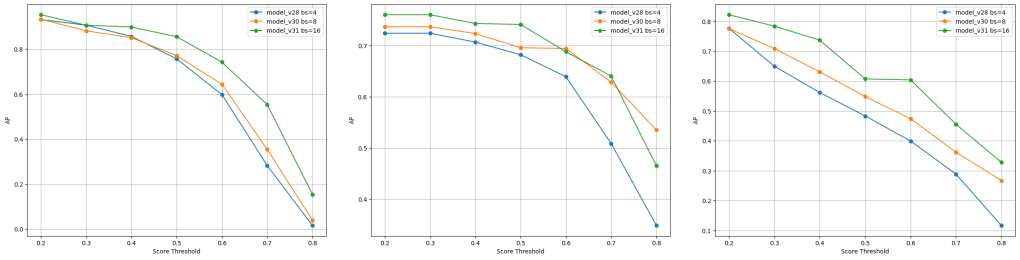



(d) Bread (e) Butter (f) Water

Figure 5.9: mAP and AP variations for different batch sizes on bounding boxes

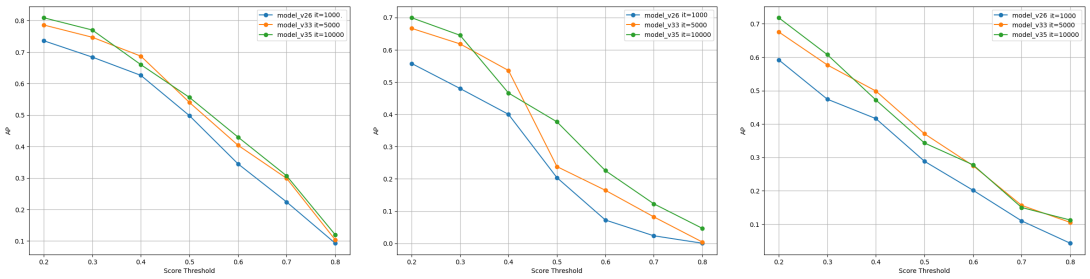



(a) mAP (b) Salad (c) Tomato

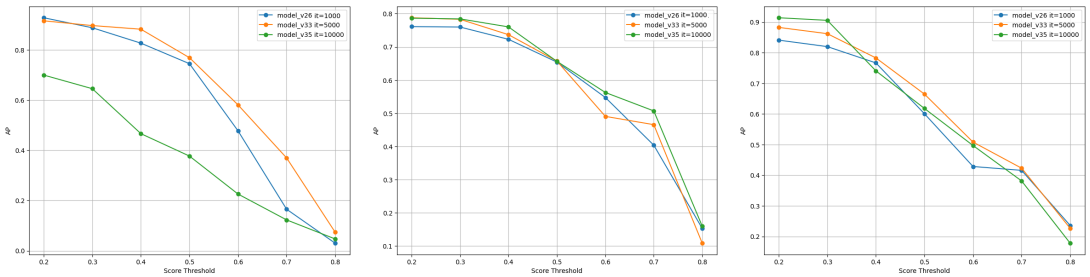



(d) Bread (e) Butter (f) Water

Figure 5.10: mAP and AP variations for different batch sizes on segmentation masks

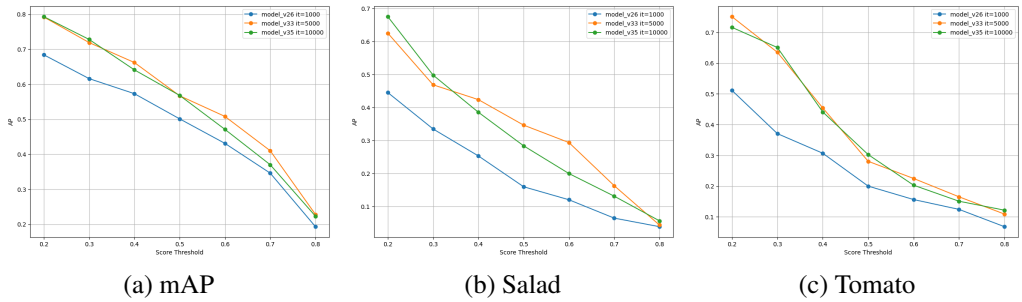



(a) mAP (b) Salad (c) Tomato

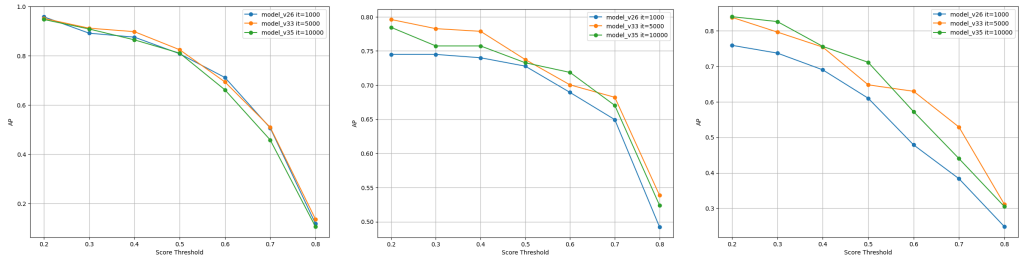



(d) Bread (e) Butter (f) Water

Figure 5.11: mAP and AP variations for different max iterations on bounding boxes




(a) mAP (b) Salad (c) Tomato




(d) Bread (e) Butter (f) Water

Figure 5.12: mAP and AP variations for different max iterations on segmentation masks

Table 5.2: AP results

| Salad | Tomato | Bread | Butter | Water |
|---|---|---|---|---|
| 24.22 | 27.45 | 81,68 | 59.62 | 59.87 |

At the moment of this study, we are not aware of a released study based on the dataset used, containing 498 food classes). Because of that, we will compare the obtained results with the ones presented in the Mohanity et al. [46] study which was conducted on the 273 classes version of this dataset. Containing only 273 classes, that dataset was less biased compared with the 498 classes version, thus the implicit higher accuracy. In Table 5.3, the results of a similar architecture and the best-resulting architecture from this study are compared with our final results.

Table 5.3: Results comparison

| Dataset | Architecture | Backbone | mAP |
|---|---|---|---|
| FoodRecognition-273[64] [46] | Hybrid Task Cascade (HTC) | ResNet101 | 0.539 |
| FoodRecognition-273 | Mask R-CNN | ResNet50 | 0.473 |
| FoodRecognition-498 [58] | Mask R-CNN | ResNet50 | 0.274 |

We visualised predictions on sample images from the validation set to additionally analyse the performance of the model. Figure 5.13 demonstrates the model's ability to detect and segment food items with bounding boxes and segmentation masks.


Figure 5.13: Bounding box and segmentation mask detection

# Chapter 6

# Mobile Application

This chapter outlines the proposed solution for the nutrition counting application, focusing on integrating semantic segmentation for food recognition into a mobile application. The system is designed to recognise, analyse, and enhance food items from images to provide nutritional information, which can improve the user's experience with dietary tracking and management.

## 6.1 System Overview

The proposed system is structured to optimise user interaction and backend processing to deliver accurate food recognition and nutritional data. The system architecture is divided into several components, each serving a specific function:

**Mobile Application:** It represents the front-end user interface where users interact with the system. This application allows users to perform onboarding, directly search for their meals, or upload images of the meals and let the model determine them. It processes the received information and displays the analysed data, such as calorie counts and nutritional content. It aims to provide accurate information and be intuitive and user-friendly.

**Image Processing Module:** After an image is uploaded through the mobile application, it is sent to the backend server. The image processing module then assumes control, handling tasks such as image pre-processing (e.g., resizing, normalisation) and extracting relevant features required for subsequent recognition tasks.

**Machine Learning Model:** At the core of the system is the machine learning model responsible for recognising, classifying and segmenting the food items in the images.

**User and Meal Information API:** The User and Meal Information API handles user registration, authentication, and meal logging. It interacts with the PostgreSQL database to store and retrieve user data and meal logs.

**Nutritional Information API:** The system integrates external APIs to fetch additional nutritional information, ensuring that the application provides comprehensive and up-to-date nutritional data.

The system architecture is presented in Figure 6.1

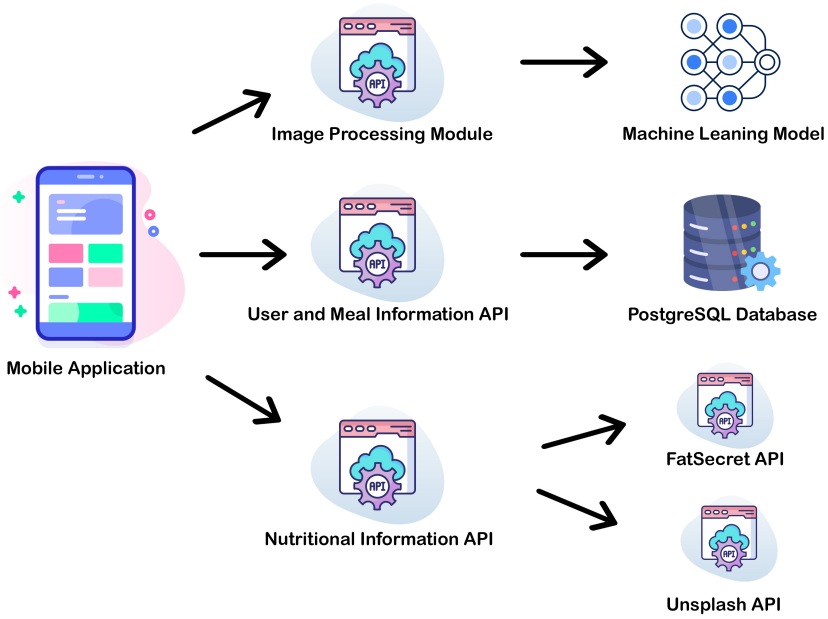



Figure 6.1: System Architecture Diagram

Integrating these components creates a clean architecture that provides a seamless and efficient user experience while providing high accuracy in food recognition and nutritional analysis.

This section describes the practical part of the solution, which includes the food recognition and segmentation tasks through a complex nutrition tracking application. Each subsection will outline the steps from the initial requirements gathering to the final testing phases by discussing the methodologies and strategies for creating this tool and providing a detailed view of the application's lifecycle.

## 6.2 Requirements Elicitation

This thesis focuses on developing a mobile application that integrates semantic segmentation to enhance nutrition tracking and dietary management. The application allows users to manually search or upload images of their meals, which are then analysed using image recognition and semantic segmentation to identify food items. Users can adjust the identified items and quantities as needed. Additionally, the application features an automated system that generates grocery lists based on the user's meal history, supporting health maintenance and dietary planning. This project aims to facilitate the user experience and increase interest

in these applications by integrating machine learning technologies and providing a seamless and interactive experience for monitoring and improving nutritional habits.

### Onboarding

The nutrition-tracking application's user registration and onboarding functionality begins when a new user launches the application for the first time. The app prompts users to register by entering essential personal details such as age, gender, height, weight, dietary goals, and activity level. This information is essential for personalising the dietary recommendations and tracking progress. The onboarding process ensures that the application is adapted to the user's specific health needs and objectives.

### Tracker Screen

The main screen functionality offers users a complete overview of their daily nutritional intake. It shows the total calories and macronutrients (proteins, carbohydrates, and fats) consumed for the selected day. The screen is organized by meal types (breakfast, lunch, dinner, and snacks). Each meal type can be expanded to display the specific food items logged, providing a clear and detailed view of the user's dietary habits.

### Meal Addition via Search

The meal addition via search functionality allows users to manually log their meals by searching for food items in the application's API. Users can input the food item and specify the amount consumed. The app calculates the nutritional information corresponding to the meal based on the entered data. This feature ensures that users can accurately track their dietary intake even if they prefer manual logging.

### Meal Addition via Image Recognition

The meal addition feature via image recognition utilises machine learning to simplify the process of logging meals. Users can upload an existing image of their meal. The application's image recognition model identifies and segments the food items, improving their visual representation by enhancing their contour. The names of the identified items and default quantities are displayed as prompts, which users can modify if needed. This feature streamlines the meal-logging process and improves accuracy by minimising manual input.

### Automated Grocery List Generation

The automated grocery list generation functionality creates a grocery list based on the user's logged meals from the previous week. The application analyses the types and quantities of food items consumed and generates a list. Users can review, check, uncheck, edit, delete, or

manually add items to the list. This feature simplifies meal planning and ensures that users have all the necessary ingredients for their planned meals.

All the functionalities are represented in figure 6.2. Central to this diagram is the 'User', the primary actor who engages with various app functionalities. The 'User' interacts with the system by onboarding, logging meals via image recognition or search, viewing daily intake, and managing a dynamically generated grocery list.

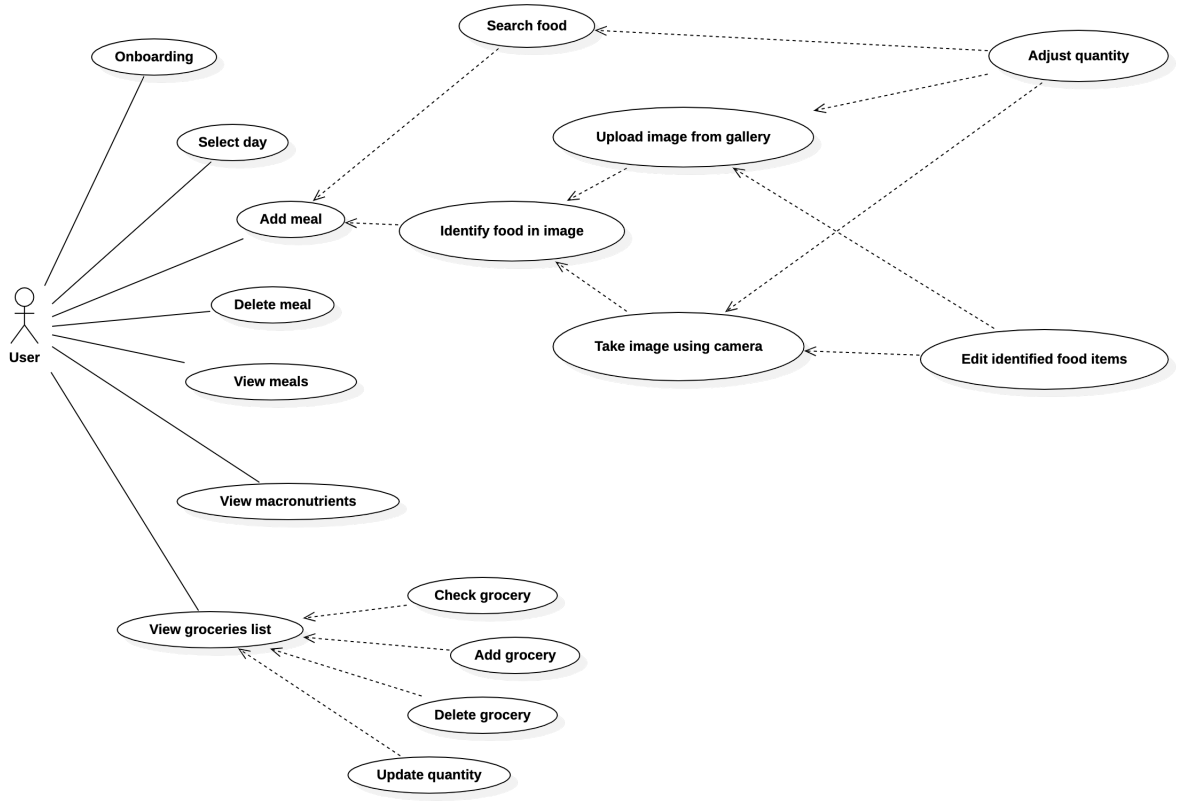



Figure 6.2: Use Case Diagram

## 6.3 Requirements Analysis

Each use case identified in the elicitation phase is analysed to detail the interaction flows between the system and the actors, particularly focusing on the sequence of actions and decision points that could affect functionality. This detailed analysis helps understand the requirements at a granular level, which is essential for the subsequent design and implementation phases.

### Onboarding

In the User Onboarding flow for the nutrition-tracking app, the process begins when a new user launches the application for the first time. The user is greeted with a welcome screen that

invites them to register or log in. The registration process involves entering personal details such as gender, age, height, weight, activity level, goal, and macronutrient percentages. This information is essential as it helps the application adapt dietary recommendations and track progress according to individual health goals and physical attributes. Once registration is complete, the user navigates to the application's main screen, the nutrition tracker overview, as shown in Figure 6.3

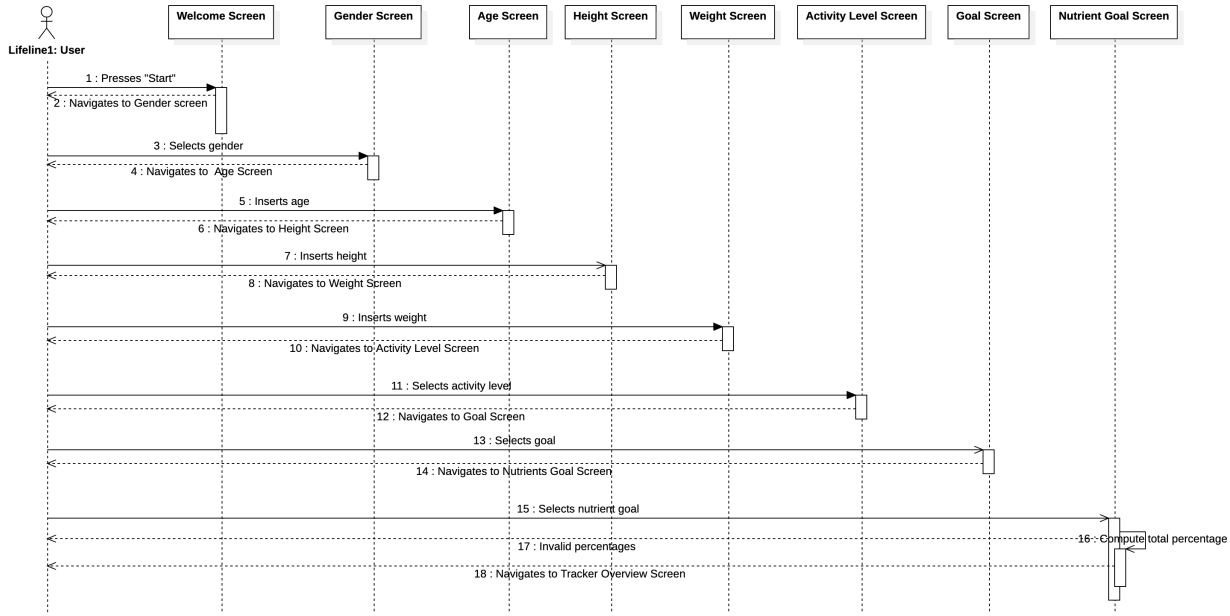



Figure 6.3: Onboarding Sequence Diagram

### Tracker Screen

The tracking screen flow for the nutrition-tracking application begins once the user successfully logs in or completes the onboarding process. The system navigates the user to the main screen and offers a comprehensive overview of their daily nutritional intake. This screen displays the total calories and macronutrients (proteins, carbohydrates, and fats) consumed for the selected day.

The user can see the meals they have logged, organised by meal types such as breakfast, lunch, dinner, and snacks. Each meal type section is expandable, allowing the user to view detailed entries of the food items logged under each meal. The screen is thoughtfully designed to provide a clear and user-friendly interface, ensuring users can easily track their nutritional intake, be aware of their dietary choices, and manage their dietary goals.

If the user wishes to add a new meal, they can select the appropriate meal-type and navigate to the Search Screen. This interactive and dynamic screen ensures users have an accurate and convenient way to monitor and manage their nutrition intake, as shown in Figure 6.4

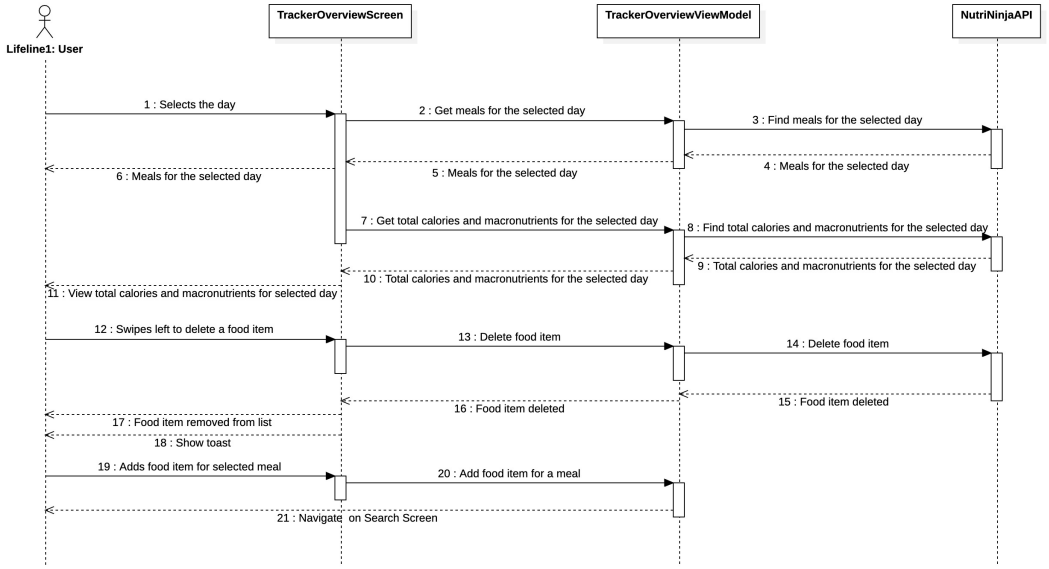



Figure 6.4: Tracker Overview Diagram

### Meal Addition via Search

In the Meal Addition via Search flow for the nutrition-tracking application, the process starts when the user decides to log a new meal. From the Tracker Screen, the user selects the option to add a meal according to the meal type (breakfast, lunch, dinner, or snack). The system navigates to the search screen, where the user can manually search for food items or use the image recognition feature.

The user is presented with a search bar where they can enter the name of the food item. As the user types, the system dynamically suggests matching items from a comprehensive food API. Once the user selects the desired food item, they are asked to specify the quantity consumed. After the user enters the quantity, the system calculates the nutritional content based on the selected food item and the specified amount. The new meal is added to the Tracker Screen, updating the daily totals for calories and macronutrients.

This flow ensures that users can accurately log their meals even when they choose not to use the image recognition feature, offering flexibility and convenience in tracking their nutrition intake.

### Meal Addition via Image Recognition

If the user opts for image recognition, they can upload an existing image from their gallery. Once the image is captured or selected, the app uses a machine learning model trained for food recognition and segmentation to analyse the image. The model identifies and segments different food items present in the meal.

The identified food items are then highlighted in the image with contours around each. For each segmented food item, a prompt appears with the name of the identified food and

a default quantity. The system allows the user to review these suggestions and make any necessary adjustments. The user can edit the identified food items, correct misidentifications, and adjust the quantities as needed.

After confirming the details, the user saves the entry, and the new meal is added to the Tracker Screen. Based on the logged meal, the app updates the daily totals for calories and macronutrients. This flow utilises advanced image recognition technology to simplify the meal logging process, making it quick and accurate for users to track their nutrition intake with minimal manual input.

The search flow is presented in the Figure 6.5

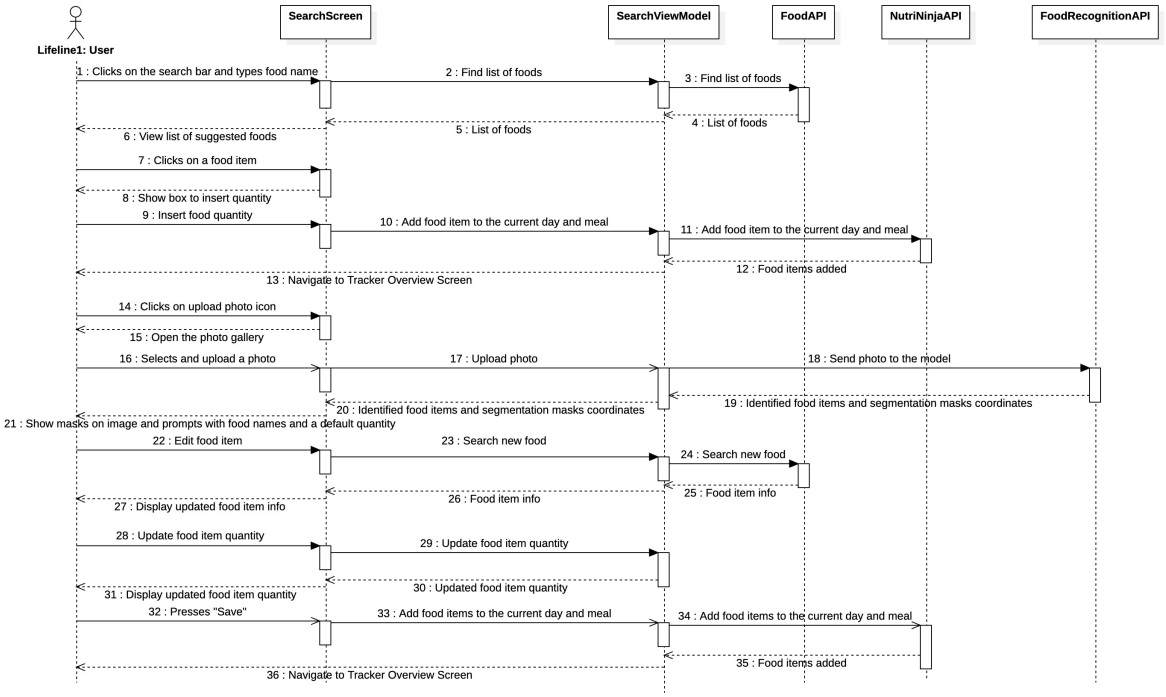



Figure 6.5: Search Diagram

### Automated Grocery List Generation

In the Automated Grocery List Generation flow for the nutrition-tracking application, the process begins with the user navigating to the grocery list feature from the bottom navigation bar. This feature optimises grocery shopping by automatically generating a list based on the user's logged meals over the past week.

When the user accesses the grocery list, the application reviews the food items logged in the user's meal history from last week. It analyses the types of meals, the frequency of each food item, and the quantities consumed. The application compiles a list of groceries needed to prepare similar meals for the upcoming week using this information.

The generated grocery list includes a suggested quantity for each food item based on the user's past consumption patterns.

The user can then review the automatically generated grocery list. They can check or uncheck items, indicating whether they need to purchase them. The application allows the user to edit quantities, delete items they do not need, and manually add new items that might not have been included in the automated list.

This automated process enhances the user's shopping experience by saving time and ensuring they have all the necessary ingredients to meet their nutritional goals and prepare their meals for the week.

## 6.4 System Design

The system design of the nutrition-tracking application is structured to ensure an accurate and user-friendly experience. It includes the architectural layout and the detailed design of the Android application and the backend services.

The application architecture consists of three primary layers: the Client Layer, the Backend Layer, and the Database Layer. The Client Layer, represented by the Android application, captures user inputs, displays data, and communicates with backend services. The Backend Layer consists of multiple APIs responsible for handling user data, meal information, and machine learning operations, ensuring efficient data processing, storage, and retrieval. The Database Layer uses PostgreSQL to store user information, meal logs, and other relevant data. These layers work together, with the client layer communicating with the backend through RESTful APIs and the backend interacting with the database to manage data.

#### Client Layer (Android Application)

The Android application is developed using Kotlin and Jetpack Compose and follows a multi-module Model-View-ViewModel (MVVM) clean architecture. This architecture enhances code organization, maintainability, and testability, making the application more scalable. The application is divided into modules, each responsible for specific features or functionalities, improving code reusability and organisation.

In the MVVM architecture, the View handles the user interface and user interactions. The ViewModel manages UI-related data and business logic, interacting with repositories to fetch and update data. The Model represents the data layer, including repositories and data sources. Repositories act as intermediaries between the ViewModel and data sources, handling data fetching from APIs and databases and providing a clean API for the ViewModel. LiveData is used to observe data changes and update the UI accordingly. Additionally, the Room database, an abstraction layer over SQLite, is used for local storage, storing user data and meal logs.

Figure 6.6 presents the class diagram of the Tracker Overview submodule.

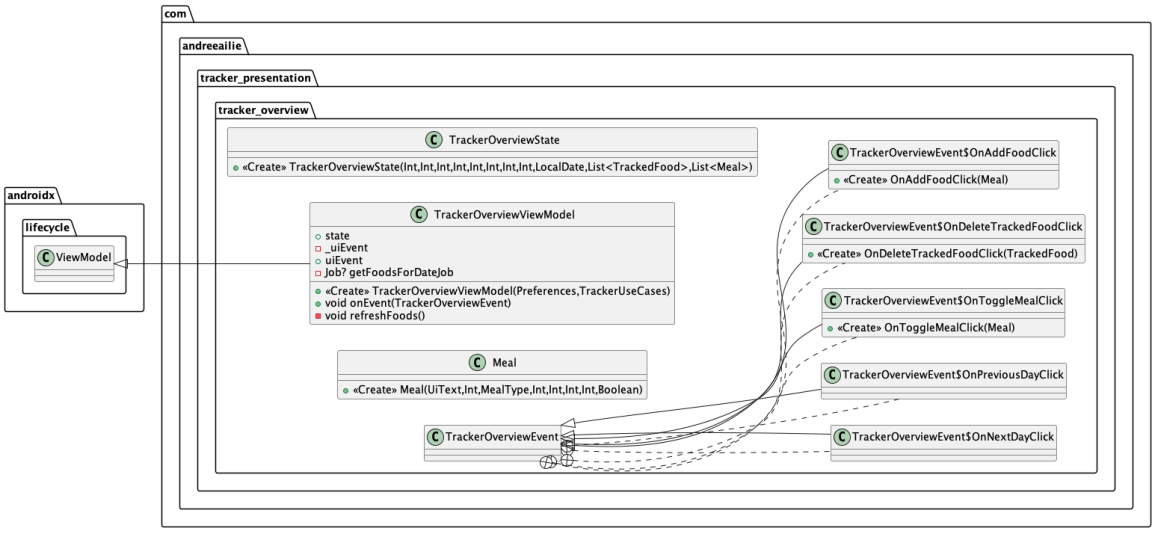



Figure 6.6: Tracker Overview submodule class diagram

Figure 6.7 presents the class diagram of the Search submodule.

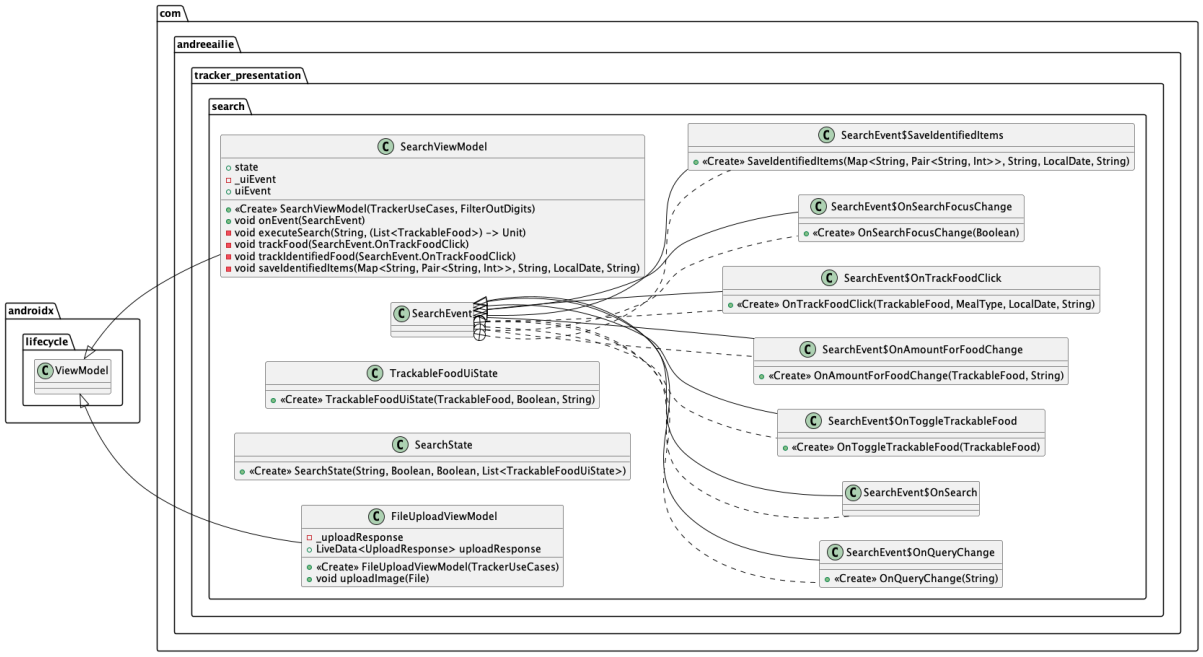



Figure 6.7: Search submodule class diagram

### Backend Layer (APIs)

The backend consists of three main APIs developed using Golang and Flask. The User and Meal Information API, developed using Golang and the Gin framework, handles user registration, authentication, and meal logging. It interacts with the PostgreSQL database to store and retrieve user data and meal logs. This API provides endpoints for registering new users, authenticating users, retrieving meals for a specific user, and logging new meals.

The Nutrition Information API, developed using Flask, encapsulates the FatSecret API and Unsplash API. It lists meals with names, images, unit measurements, and macronutrients based on search queries. This API allows users to search for food items and receive nutritional information.

The Food Recognition API, also developed using Flask, handles image recognition and segmentation. It uses machine learning models to identify food items in an image and provide segmentation masks and confidence scores. This API receives images from the client layer, processes them to identify and segment food items, and returns the results to the client.

#### Database Layer

The database layer uses PostgreSQL to store user data, meal logs, and nutritional information. The database includes tables for users, meals, and food items. The User Table stores user details such as user ID, username, password (hashed), age, gender, height, weight, activity level, and dietary goals. The Food Item Table contains detailed food items within each meal, including food item ID, meal type, name, quantity, and macronutrients.

## 6.5 Implementation

The implementation of the nutrition-tracking mobile application involves multiple components, each indispensable in delivering a flawless and efficient user experience. This subchapter details the implementation of the Android application, the backend APIs, and the integration of various services.

#### Android Application

The Android application is developed using Kotlin and targets Android API level 34. Kotlin was chosen for its modern features, interoperability with Java, and strong support from the Android development community. The application architecture follows the Model-View-ViewModel (MVVM) pattern. This architecture allows a clear separation of concerns and easier maintenance and testing.

The goal of the user interface (UI) is to be intuitive and user-friendly, providing easy access to all functionalities. The main screen displays the user's daily nutritional summary, with expandable sections for each meal type (breakfast, lunch, dinner, and snacks). The UI components are built using Jetpack Compose, which allows for a declarative approach to designing UI, enhancing readability and reducing boilerplate code.

The backend infrastructure is composed of multiple APIs, each serving a specific purpose:

### User and Meal Information API

The primary backend API is built using Golang, using the Gin framework for routing and handling HTTP requests. This API manages user data and meal logs, interfacing with a PostgreSQL database to store and retrieve information efficiently. Golang and Gin were selected for their performance, scalability, and minimalistic approach to web development.

### FatSecret and Unsplash API Encapsulation

A Flask API is implemented to encapsulate the FatSecret and Unsplash APIs. This intermediary API handles user queries by fetching relevant meal data and images, standardising the responses to include the meal name, image, unit measure, and macronutrients.

### Food Recognition API

The Food Recognition API processes images uploaded by users to identify and segment food items. This API is built using Python, Flask and Detectron2 integrating the Food Recognition model. The model processes the image, identifies food items, and returns segmentation masks and confidence scores.

### Integration

The integration of these components ensures a smooth and efficient workflow. The Android application communicates with the backend APIs using Retrofit, a type-safe HTTP client for Android.

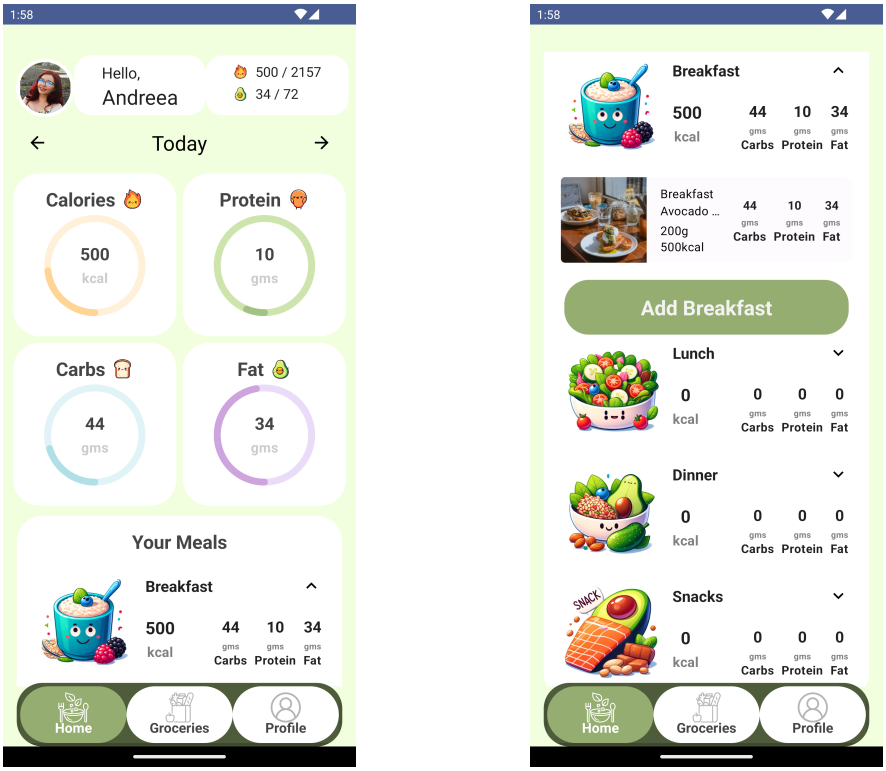
## 6.6 Interface

The application contains four main screens on which the user can navigate easily. If the user is not logged in when opening the application, there will be another screen, including a welcome message, the login/register screen and multiple onboarding screens.

The Tracker Screen of the application displays the total calories and macronutrients (proteins, carbohydrates, and fats) consumed for the selected day. The user can see their logged meals, organised by meal types, such as breakfast, lunch, dinner, and snacks. Each meal type section is expandable, allowing the user to view and edit detailed entries of the food items logged under each meal.

Figure 6.8 contains two screenshots of the Android application, illustrating the Tracker Screen.

The Search Screen allows the user to manually log their meal or use the food image recognition module.




(a) Main Screen (b) Main Screen

Figure 6.8: Android application Tracker Screen

For the first option, the user can use a search bar to enter the food item's name. The system dynamically suggests matching items from a comprehensive food database as the user types. Once the user selects the desired food item, they are asked to specify the quantity consumed.

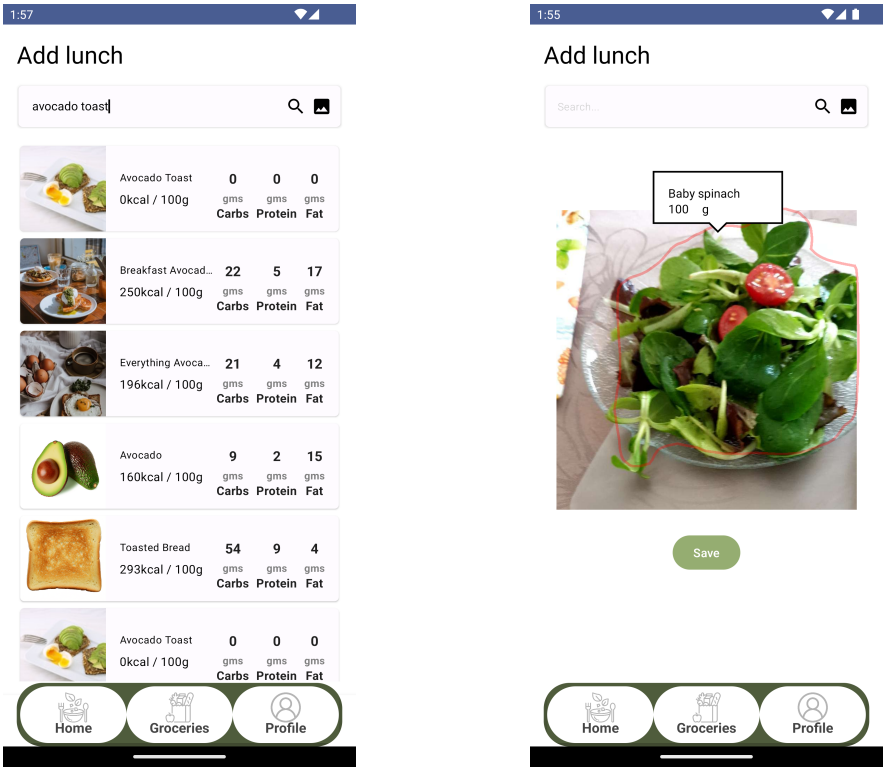
The user can use the food recognition module by pressing the image upload icon on the search bar. This action will allow them to open the photo gallery and select and upload an image. A processed image containing the segmentation masks, a prompt for each identified food item, and a default quantity will appear. The user can edit the identified food items and their quantities and save the entries.

Figure 6.9 contains two screenshots of the Android application, illustrating the search screen and meal logging using manual search and food image recognition feature.

Each main screen of the application contains a navigation bar on the bottom of the screen from where the user can navigate to the Tracker Screen, Grocery List Screen and Profile Screen.

The Grocery List Screen presents an automated generated list based on the user's logged meals over the past week. Each item includes a suggested quantity based on the user's past consumption patterns. The user can review the generated list, check or uncheck items, delete them or add new ones.

The Profile Screen presents the user information such as name, email, profile picture,




(a) Search Screen with manual food entry

(b) Search Screen with image recognition on uploaded image

Figure 6.9: Android application Search Screen

height, weight, and nutrient goals that can be edited. The user can also change the account password, log out or delete the account.

## 6.7 Testing

We conducted a series of tests for this nutrition-tracking application, covering backend and client-side components.

Unit tests were created for the User and Meal Information API to validate the main functionalities. These tests primarily focused on the handlers, services, and routes.

We used unit tests, data layer tests with mock web servers, and end-to-end (E2E) tests for the mobile application. The unit tests focused on the core use cases of the application, such as tracking meals and interacting with the data layer. Using a mock web server, we could simulate API responses and test how the application handles different scenarios, such as unexpected data formats. End-to-end testing was conducted to validate the entire user journey, from food logging to generating grocery lists. These tests ensure all application components work seamlessly together, including the user interface, data processing, and backend integration. For example, the food tracking feature was tested by simulating user interactions such as adding a meal, verifying that the correct nutritional information is displayed, and

checking that the meal is saved correctly.

# Chapter 7

# Conclsuions and Future Work

The thesis presented in this document discussed the development of a mobile nutrition-tracking application using machine learning techniques, particularly semantic segmentation, to enhance food image recognition. The application aims to provide users with a reliable and precise tool to help monitor their dietary intake and encourage them to make healthier dietary choices.

The goal of this research was to create an engaging mobile application capable of recognising and segmenting various food items from images uploaded by the user. Semantic segmentation has shown promising results in identifying food categories and has helped achieve this goal. The results obtained from the experiments were encouraging for some food categories, showing that the model could identify and segment food items with a reasonable degree of accuracy. However, the model performance was highly influenced by the quality of the training dataset, which presented biases in representing some of the food categories.

The application successfully achieved its primary objectives in terms of usability. Users could log meals through manual search or image uploads, receive accurate nutritional information, and manage their dietary intake with minimal effort. The automated grocery list generation feature increased user convenience by suggesting grocery items based on past consumption patterns.

While the current implementation of the application demonstrates a solid foundation, several avenues for future work could significantly improve its functionality and user experience.

One improvement would be allowing users to take pictures directly from the application rather than uploading them from their gallery. This feature would streamline the meal-logging process and make the application more convenient. Additionally, incorporating a barcode scanning feature would provide users with an alternative method for logging food items, especially for packaged goods, thereby increasing the application's versatility.

Machine learning algorithms could enhance the grocery list feature. The application could generate personalised grocery lists by analysing users' dietary habits, preferences, and nutritional needs. Such a feature would reduce the time spent on these actions and encourage

healthier purchasing decisions.

Gamification is another potential enhancement that could increase user engagement. The application could increase the user's motivation by allowing them to connect with friends, share their progress, and participate in challenges. Gamification has been shown to be effective in other health-related applications, and its integration into a nutrition-tracking application could lead to better adherence to dietary goals.

Machine learning techniques, particularly semantic segmentation, remain at the core of the functionality of the application. However, the accuracy of the model could be improved in future work through better training datasets. Addressing the biases in the current dataset and expanding it to include a more diverse range of food items would likely improve the model's generalisation capabilities. Further refinement of the architecture of the model and hyperparameters could enhance its performance, leading to more precise food recognition.

Another area for future research is the development of an approach for precise weight approximation using machine learning algorithms. By using additional data, such as image dimensions and user feedback, the application could estimate the weight of food items, providing users with more reliable nutritional information.

# Bibliography


[1] Haseeb Ahsan. A brief overview and history of human nutrition and health. *Current Biochemistry*, 9:98–103, 12 2022.

[2] Dariush Mozaffarian, Irwin Rosenberg, and Ricardo Uauy. History of modern nutrition science—implications for current research, dietary guidelines, and food policy. *BMJ*, 361:k2392, 2018.

[3] John Yudkin. *Pure, White and Deadly*. Viking Press, 1972.

[4] Lisa Jahns, Wendy Davis-Shaw, Alice H Lichtenstein, Suzanne P Murphy, Zach Conrad, and Forrest Nielsen. The history and future of dietary guidance in america. *Advances in Nutrition*, 9(2):136–147, 2018.

[5] Harvard T.H. Chan School of Public Health. Diet review: Mediterranean diet, 2023. Accessed: 2024-08-23.

[6] Institute of Medicine. *Dietary Reference Intakes for Energy, Carbohydrate, Fiber, Fat, Fatty Acids, Cholesterol, Protein, and Amino Acids*. The National Academies Press, Washington, DC, 2005.

[7] MyFitnessPal. Meal Scan FAQ. `https://support.myfitnesspal.com/hc/en-us/articles/360045761612-Meal-Scan-FAQ`. Online; accessed 30 April 2024.

[8] Fitbit. The Fitbit App. `https://www.fitbit.com/global/us/technology/fitbit-app`. Online; accessed 23 August 2024.

[9] B. N. Limketkai, K. Mauldin, N. Manitius, L. Jalilian, and B. R. Salonen. The age of artificial intelligence: Use of digital technology in clinical nutrition. *Current Surgery Reports*, 9(7):20, 2021.

[10] Sabiha Samad, Fahmida Ahmed, Samsun Naher, Muhammad Ashad Kabir, Anik Das, Sumaiya Amin, and Sheikh Mohammed Shariful Islam. Smartphone apps for tracking food consumption and recommendations: Evaluating artificial intelligence-based functionalities, features and quality of current apps. *Intelligent Systems with Applications*, 15:200103, 2022.

[11] Foodvisor. Foodvisor App. `https://www.foodvisor.io/en/`. Online; accessed 23 August 2024.

[12] MyFitnessPal.

[13] LoseIt! LoseIt! App. `https://www.loseit.com/`. Online; accessed 23 August 2024.

[14] My Net Diary. My Net Diary App. `https://www.mynetdiary.com/`. Online; accessed 23 August 2024.

[15] Lifesum. Lifesum App. `https://lifesum.com/`. Online; accessed 23 August 2024.

[16] Foodvisor. Food Image Recognition Explained. `https://www.foodvisor.io/en/guides/article/food-image-recognition-explained/`. Online; accessed 30 April 2024.

[17] Lose It! Snap It. `https://www.loseit.com/snapit/`. Online; accessed 30 April 2024.

[18] How the grocery list works. `https://support.mealime.com/article/75-how-the-grocery-list-works`, 2024. Online; accessed 30 April 2024.

[19] Paprika app help - groceries. `https://www.paprikaapp.com/help/ios/#groceries`, 2024. Online; accessed 30 April 2024.

[20] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, 2017.

[21] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image segmentation using k -means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015.

[22] Haim Permuter, Joseph Francos, and Ian Jermyn. A study of gaussian mixture models of color and texture features for image classification and segmentation. *Pattern Recognition*, 39(4):695–706, 2006. Graph-based Representations.

[23] Alexander Ze Hwan Ooi, Zunaina Embong, Aini Ismafairus Abd Hamid, Rafidah Zainon, Shir Li Wang, Theam Foo Ng, Rostam Affendi Hamzah, Soo Siang Teoh, and Haidi Ibrahim. Interactive blood vessel segmentation from retinal fundus image based on canny edge detector. *Sensors*, 21(19), 2021.

[24] Regina Pohle and Klaus D. Toennies. Segmentation of medical images using adaptive region growing. In Milan Sonka and Kenneth M. Hanson, editors, *Medical Imaging*

*2001: Image Processing*, volume 4322, pages 1337 – 1346. International Society for Optics and Photonics, SPIE, 2001.

[25] Xiaoyan Zhang, Yong Shan, Wei Wei, and Zijian Zhu. An image segmentation method based on improved watershed algorithm. In *2010 International Conference on Computational and Information Sciences*, pages 258–261, 2010.

[26] Min Chen, Xiaobo Shi, Yin Zhang, Di Wu, and Mohsen Guizani. Deep feature learning for medical image analysis with convolutional autoencoder neural network. *IEEE Transactions on Big Data*, 7(4):750–758, 2021.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[29] Vishnu Subramanian. The concept of artificial neurons perceptrons in neural networks. `https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fa` 2019. Online; accessed: 20 April 2024.

[30] Marc Chia. Applied deep learning - part 1: Artificial neural networks. 2018. Online, accessed: 7 May 2024.

[31] Papers with Code. Max pooling. `https://paperswithcode.com/method/max-pooling`, 2021. Online, accessed: 7 May 2024.

[32] Nafiz Shahriar. What is convolutional neural network (cnn)? deep learning. `https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5` 2020. Online, accessed: 7 May 2024.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1106–1114, 2012.

[35] Minghuang Ma, Haoqi Fan, and Kris M. Kitani. Going deeper into first-person activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[37] DeepLobe. Exploring object detection: Applications and benefits. `https://deeplobe.ai/exploring-object-detection-applications-and-benefits/`, 2023. Online, accessed: 12 May 2024.

[38] Towards AI. Machine learning 7. `https://towardsai.net/p/l/machine-learning-7`, 2023. Online, accessed: 12 May 2024.

[39] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[40] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[42] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. pages 2980–2988, 10 2017.

[43] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6154–6162, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[44] Kai Chen, Wanli Ouyang, Chen Change Loy, Dahua Lin, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, and Jianping Shi. Hybrid task cascade for instance segmentation. pages 4969–4978, 06 2019.

[45] Xiongwei Wu, Xin Fu, Ying Liu, Ee-Peng Lim, Steven C. H. Hoi, and Qianru Sun. A large-scale benchmark for food image segmentation, 2021.

[46] Sharada Prasanna Mohanty, Gaurav Singhal, Eric Antoine Scuccimarra, Djilani Kebaili, Harris Héritier, Victor Boulanger, and Marcel Salathé. The food recognition

benchmark: Using deep learning to recognize food in images. *Frontiers in Nutrition*, 9:875143, 2022.

[47] Chi-Sheng Chen, Guan-Ying Chen, Dong Zhou, Di Jiang, and Dai-Shi Chen. Res-vmamba: Fine-grained food category visual classification using selective state space models with deep residual learning, 2024.

[48] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of multiple-food images by detecting candidate regions. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, Melbourne, VIC, 2012. IEEE.

[49] Y. Kawano and K. Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *Proceedings of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, Zurich, 2014. ECCV.

[50] T. Ege and K. Yanai. A new large-scale food image segmentation dataset and its application to food calorie estimation based on grains of rice. In *Proceedings of ACMMM Workshop on Multimedia Assisted Dietary Management (MADiMa)*, Nice, 2019. ACMMM.

[51] K. Okamoto and K. Yanai. Uec-foodpix complete: a large-scale food image segmentation dataset. In *Proceedings of ICPR Workshop on Multimedia Assisted Dietary Management (MADiMa)*, Milan, 2021. ICPR.

[52] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, pages 446–461. Springer, 2014.

[53] Weiqing Min, Zhiling Wang, Yuxin Liu, Mengjiang Luo, Liping Kang, Xiaoming Wei, Xiaolin Wei, and Shuqiang Jiang. Large scale visual food recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9932–9949, 2023.

[54] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3020–3028, 2017.

[55] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *TPAMI*, pages 187–203, 2019.

[56] My food repo application. `https://www.myfoodrepo.org/`. Online; accessed 2 May 2024.

[57] Food recognition challenge. https://www.aicrowd.com/challenges/food-recognition-challenge. Online; accessed 2 May 2024.

[58] Food recognition dataset - 498 categories. https://www.kaggle.com/datasets/sainikhileshreddy/food-recognition-2022. Online; accessed 2 May 2024.

[59] Eduardo Aguilar, Beatriz Remeseiro, Marc Bolaños, and Petia Radeva. Grab, pay, and eat: semantic food detection for smart restaurants. *IEEE Transactions on Multimedia*, 20:3266–3275, 2018.

[60] He Ye and Qingnan Zou. Food recognition and dietary assessment for healthcare system at mobile device end using mask r-cnn. In Hui Gao, Kang Li, Xiao Yang, and Yunjie Yin, editors, *Testbeds and Research Infrastructures for the Development of Networks and Communications*, pages 18–35. Springer International Publishing, Cham, 2020.

[61] Charles N. C. Freitas, Filipe R. Cordeiro, and Valmir Macario. Myfood: a food segmentation and classification system to aid nutritional monitoring. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 234–239, Recife; Porto de Galinhas, 2020.

[62] Facebook AI Research. Detectron 2. https://ai.meta.com/tools/detectron2/. Online; accessed 8 August 2024.

[63] viso.ai. What is intersection over union (iou)? https://viso.ai/computer-vision/intersection-over-union-iou/. Online; accessed 16 August 2024.

[64] Food recognition dataset - 273 categories. https://www.aicrowd.com/challenges/food-recognition-challenge/dataset_files. Online; accessed 26 August 2024.