FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Carbohydrate estimation through image recognition

Cristiana Maria Monteiro Ribeiro

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Luís Filipe Teixeira

Second Supervisor: João Gonçalves

February 17, 2020

Carbohydrate estimation through image recognition

Cristiana Maria Monteiro Ribeiro

Mestrado Integrado em Engenharia Informática e Computação

February 17, 2020

Abstract

Type 1 diabetes (T1D) diabetes is a chronic illness in which the pancreas produces little or no insulin. Insulin is a hormone used by our body to allow glucose to be absorbed by cells in order to produce energy. This disease has no cure, therefore, counting the meal's carbohydrates (CHO) is a key factor in determining the exact insulin dose and maintaining normal blood glucose levels. The estimation of the prandial dose is a complex and time-consuming task. Although an inaccuracy of $\pm 10g$ in CHO counting does not impair blood glucose levels, a fluctuation of $\pm 20g$ seriously impairs glycemic control.

The high number of patients with diabetes around the world and the proven inability to accurately count CHO has raised the need for automated tools that support these patients in the counting task, in order to know the doses of insulin they should inject. The widespread use of high-tech smartphones, coupled with recent advances in computer vision and deep learning, can help to build applications that automatically recognize the food and estimate its amount of CHO.

The goal of this project is to find a method that is able to estimate the carbohydrates amount present in the meal, through a photograph taken with a smartphone of a plate of food.

This complex problem can be divided into two stages, the first one consists of two tasks: image classification and image segmentation. The second stage refers to the volume estimation of the food objects detected in the input image. To address this problem, several implementations have been tested. To segment and classify food objects in an image, three different networks were used: Mask R-CNN, RetinaNet and Deeplabv3+. The best results were obtained through the use of Mask R-CNN network, which allowed us to train a model with the value of 0.502 for mAP evaluation on the test subset.

For the second stage, the estimation of the food object volume detected by the segmentation network, two different approaches were followed. The solution was achieved by generating the depth map of the input image, and then, use an object in the image with known dimensions to work as the calibration object and in this case a plate was considered. This approach requires just one input image.

Resumo

A diabetes tipo 1 (DM1) ou diabetes insulinodependente é uma doença crónica na qual o pâncreas produz pouca ou nenhuma insulina. A insulina é uma hormona e tem como função metabolizar a glicose (açucar no sangue) necessária para a produção de energia. Esta doença não tem cura, portanto, a contagem dos hidratos de carbono da refeição é um fator chave na determinação da dose exata de insulina e na manutenção dos níveis normais de glicose no sangue. A estimativa da dose prandial é uma tarefa complexa e demorada. Apesar de uma imprecisão de $\pm 10g$ na contagem CHO não prejudicar os níveis de glicose no sangue, uma variação de $\pm 20g$ prejudica seriamente o controlo glicémico.

O elevado número de pacientes com diabetes em todo o mundo e a comprovada incapacidade de contabilizar com precisão a quantidade de hidratos de carbono originaram a necessidade de desenvolver ferramentas que façam a contagem de forma automática, a fim dos pacientes saberem as doses de insulina que devem injetar. A ubiquidade tecnológica dos smartphones aliada aos recentes avanços na visão computacional e *deep learning*, pode ajudar a criar aplicações que reconheçam automaticamente os alimentos e estimem a sua quantidade de hidratos de carbono.

O objetivo é encontrar um método que seja capaz de estimar a quantidade de hidratos de carbono presentes numa refeição através de uma fotografia tirada com um *smartphone* a um prato de comida.

Este problema complexo pode ser dividido em duas fases, a primeira fase é composta por duas tarefas: classificação e segmentação da imagem. A segunda fase é referente à estimativa do volume dos alimentos detetados na imagem. Para resolver este problema, várias implementações foram testadas. Para classificar e segmentar os alimentos numa imagem, três redes neuronais foram usadas: Mask R-CNN, RetinaNet e Deeplabv3+. Os melhores resultados foram obtidos com a implementação da Mask R-CNN que nos permitiu treinar um modelo com o valor de 0.502 na avaliação da mAP no *dataset* de teste.

Para a segunda fase, estimar o volume dos alimentos detetados pela rede de segmentação, duas abordagens foram seguidas. A solução foi conseguida através da geração dos mapas de profundidade da imagem e depois, um objeto com as dimensões conhecidas é usado como objeto de calibração. Esta abordagem requer apenas uma imagem como entrada.

Acknowledgements

First of all, I want to thank to Fraunhofer AICOS for embrace and support the dissertation theme that I chose. If training a model through a GPU can take ten days, I can not imagine the weeks that I would have to wait until a model was ready on my laptop, which does not have a GPU.

I am especially grateful to my supervisor at Fraunhofer AICOS, João Gonçalves, for all the clarified doubts.

The same heartfelt thanks goes to Professor Luís Teixeira that accepted to guide my work.

On a more personal note, I can not forget to thank to my family who always did everything for me and who despite all the sacrifices never gave up on me and made this possible.

To Pedro Moura, for being the person I can always count on.

Last but not least, I would like to thank to the friends that the course gave me, especially to *Família Real* who always gave me strength and allowed that the academic pathway was done in a more funny way.

Cristiana Maria Monteiro Ribeiro

"I have always believed, and I still believe, that whatever good or bad fortune may come our way we can always give it meaning and transform it into something of value. "

Hermann Hesse

Contents

List of Figures

List of Tables

Abbreviations

3D	Three-Dimensional
ANN	Artificial Neural Networks
AP	Average Precision
BRISK	Binary Robust Invariant Scalable Keypoints
BoF	Bag of Features
CE	Cross Entropy
CF	Correction Factor
CHO	Carbohydrate
CNN	Convolutional Neural Network
DCD	Dominant Color Descriptor
EFD	Entropy-Based Categorization and Fractal Dimension Estimation
FAIR	Facebook Artificial Intelligence Research
FPN	Feature Pyramid Network
FCN	Fully Convolution Network
FREAK	Fast Retina Key
GFD	Gabor-Based Image Decomposition and Fractal Dimension Estimation
ICR	Insulin to Carbohydrate Ratio
IoU	Intersection over Union
KNN	K-Nearest Neighbors
Mask R-CNN	Masked Region based Convolution Neural Network
mAP	Mean Average Precision
NMS	Non-maximum Suppression
OID	Open Images Dataset
ORB	Oriented fast and Rotated Brief
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RF	Random Forest
ROI	Regions of Interest
RPN	Region Proposal Network
SCD	Scalable Color Descriptor
SIFT	Scale-Invariant Feature Transform
SSD	Single Shot MultiBox Detector
SURF	Speeded Up Robust Features
STF	Semantic Texton Forests
SVM	Support Vector Machine
SWOT	Strengths Weaknesses Opportunities Threats
T1D	Type 1 Diabetes
YOLO	You Only Look Once
FSANZ	Food Standards Australia New Zealand

Chapter 1

Introduction

This Chapter contextualizes the scope of this project in Section 1.1 and describes the motivations and objectives of this project in Section 1.2. Then, Section 1.3 briefly explains what is the impact and contribution of this project on society. Finally, section 1.4 explains how the rest of this document is structured, and what content should be expected in it.

1.1 Context

Diabetes mellitus is a group of metabolic diseases, where normally, blood glucose levels are very high over a long period of time, due to the incapacity of the body to produce and/or use insulin (the hormone that controls blood sugar levels, or glucose). The number of diabetics has been increasing worldwide.

According to World Health Organization [Org16], while in 1980 there were 108 million diagnosed patients, in 2014, approximately 422 million adults were living with diabetes. In 2012, diabetes killed 1.5 million people and blood glucose levels higher than recommended have caused the death to 2.2 million people due to the increased risk of cardiovascular and other diseases. A particular concern is that 43% of these 3.7 million deaths have occurred before the age of 70 years.

On the other hand, if diabetes is unsuccessfully managed, acute complications develop that threaten health and endanger life. Over time, raised blood sugar damages the heart and blood vessels, eyes, kidneys, nerves and teeth. More than that, people with diabetes also have a higher risk of developing infections and suffering from cardiovascular disease, blindness, kidney failure, and lower limb amputation.

Type 1 diabetes (T1D) is a chronic illness that occurs when the immune system, the body's system for fighting infection, attacks and destroys the insulin-producing beta cells of the pancreas. There are still no proven theories that explain the onset of type 1 diabetes, however, genes and environmental factors may be related.

The disease has no cure and therefore, type 1 diabetics need to inject daily insulin doses to control their blood sugar levels. Insulin injection is in a class of medications called hormones and is used to take the place of insulin that is normally produced by the body. It allows the blood sugar to be absorbed into other tissues of the body where it is used for energy. It also prevents the liver from producing more sugar. All of the types of insulin that are available in the market work in this way, they differ only in how quickly they begin to work and how long they continue to control blood sugar [oM19].

First, to calculate the dose of insulin to inject there are some things to take into account:

- Approximately 40-50% of the total daily insulin dose is to replace insulin overnight, when the patient is fasting and between meals. This is called background or basal insulin replacement. The basal insulin dose usually is constant from day to day, for every patients.
- The other 50-60% of the total daily insulin intake is for carbohydrate (CHO [1]) coverage and high correction of blood sugar. This is called bolus insulin replacement and the type of insulin to be injected is called rapid-acting insulin.

As the basal insulin dose is usually constant and is prescribed by the doctor accompanying the patient, the diabetic just has to calculate the insulin bolus dose. The calculation of the bolus dose depends on the number of carbohydrates ingested (Meal Bolus), blood glucose levels (Correction Bolus), or both [SN14].

A Bolus Dose for Carbohydrate or Meal Bolus is calculated using the Insulin to Carbohydrate Ratio (ICR). ICR is the amount of rapid-acting insulin that a diabetic need for a specific amount of carbohydrate in food. This is the number of grams of carbohydrates that 1 unit of rapid-acting insulin will cover. For example, if a meal has 60 grams of carbohydrates and ICR is 1:10. 60 (grams of carbohydrates) divided by ($\div$) 10 (carbohydrate ratio) = 6 (carbohydrate bolus), so the diabetic should inject 6 units rapid-acting insulin for the carbohydrate bolus.

To determine the Correction Bolus, the Correction Factor (CF) also called sensivity is used. CF is how much 1 unit of rapid-acting insulin will reduce the blood glucose number. The target number is the blood glucose number defined by the diabetic. For example, if the blood glucose is 220, the CF is 50 and the target set is 120; the amount to correct is given by subtracting from 220 (actual blood glucose) the 120 (value of target blood glucose). This results in a value of 100 which after being divided by 50 (Correction Factor) returns 2 as correction bolus, so the diabetic feel the necessity for 2 units of rapid-acting insulin to bring its blood glucose back into target range.

If rapid-acting insulin is given with a meal, the correction dose is added to meal dose [Hos19].

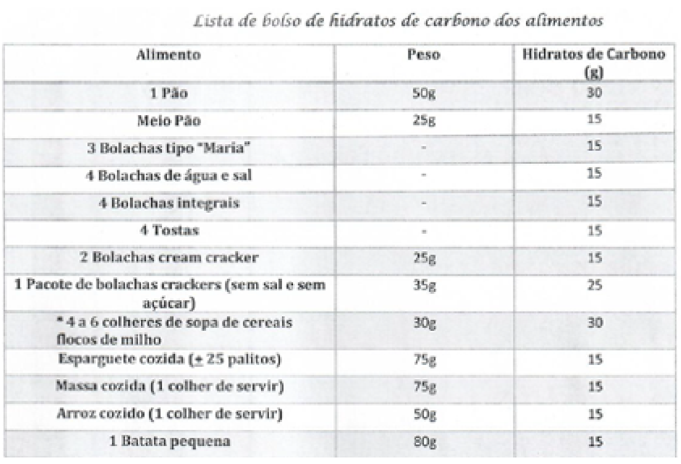
1.2 Motivation and goals

The Portuguese National Health Service is a structure through which the Portuguese Republic assures the right to health (promotion, prevention and surveillance) to all citizens of Portugal.

[1] referencing the three constituent elements, carbon (C), Hydrogen (H) and Oxygen (O)

When a patient is diagnosed as having type 1 diabetes, the approach followed by National Health Service is still very rudimentary, in the sense that only long lists are given to the diabetic. These lists indicate that a certain portion of food x has y grams of carbohydrates.

For a diabetic patient it is practically impossible to memorize all the nutritional tables, like the one in Figure 1.1. Even if the patient is able to do it, he must know what is the exact amount of food he eats in order to determine the amount of carbohydrates that is present in that food.



Lista de bolso de hidratos de carbono dos alimentos

Alimento	Peso	Hidratos de Carbono (g)
1 Pão	50g	30
Meio Pão	25g	15
3 Bolachas tipo "Maria"	-	15
4 Bolachas de água e sal	-	15
4 Bolachas integrais	-	15
4 Tostas	-	15
2 Bolachas cream cracker	25g	15
1 Pacote de bolachas crackers (sem sal e sem açúcar)	35g	25
* 4 a 6 colheres de sopa de cereais flocos de milho	30g	30
Esparguete cozida (± 25 palitos)	75g	15
Massa cozida (1 colher de servir)	75g	15
Arroz cozido (1 colher de servir)	50g	15
1 Batata pequena	80g	15

Figure 1.1: Excerpt from a table provided to diabetic patients by National Health Service

The estimation of the prandial dose is a complex and time-consuming task. Although an inaccuracy of $\pm 10g$ in CHO counting does not impair blood glucose levels [SRE+09], a fluctuation of $\pm 20g$ seriously impairs glycemic control [SKMC12]. Most T1D patients can not count CHO efficiently as suggested by the authors of [RGGEJC00, BMS+09, SRE+10, FLD+16].

According to WHO, around 10% of all people with diabetes have type 1 diabetes [Org16]. The high number of patients with diabetes around the world, and the proven inability to accurately count CHO, has raised the need for automated tools that support these patients in the counting task, in order to know the doses of insulin they should inject. The widespread use of high-tech smartphones coupled with recent advances in computer vision and machine learning can help to build applications that automatically recognize the food and estimate its amount of CHO.

Therefore, the goal of this project is to find a method that makes a viable estimation of the number of carbohydrates present in the meal, through a photograph taken with a smartphone to a plate of food.

1.3 Contributions

At the end, it is expected that the work and techniques used by other authors to address similar problems are understood, in order to put together the best of the different approaches to achieve a more accurate result.

The proposed method can be implemented as a mobile application and the task of counting carbohydrates becomes easier and allows diabetics to take a step forward in the treatment of their health.

1.4 Document structure

Throughout the Chapter 2, the main theoretical concepts to understand the problem are discussed, as well as the works and approaches proposed by other authors related to each topic. This chapter is divided into two parts: traditional machine learning approaches 2.1 and Deep Learning Approach 2.2. The first part includes Food Image Segmentation, Feature Extraction and Algorithms for Image Classification. In the second part, deep learning approaches are explored and the Convolutional Neural Networks are extensively explained. The section 2.3 is called Food Volume Estimation and reviews the main methods and approaches to perform the food volume estimation. Finally, the section 2.4 describes the approaches of the authors who also developed applications capable of doing the image recognition and the estimation of the volume.

In Chapter 3, the literature review of the used networks is done and the theoretical concepts that ground the used implementations are described there. Mask R-CNN 3.3, RetinaNet 3.4, Deeplab 3.5, ResNet 3.1, and the work used to estimate the depth images 3.6 are all described along the chapter.

In Chapter 4, the proposed solution is described in 4.1. The datasets to be used are mentioned in 4.1.1 and the conversion from volume to weight is explained in 4.1.2.

Throughout the Chapter 5, the results obtained are presented and discussed and finally in Chapter 6 the main conlusions extracted from the present work are discussed as well as missing tasks are referenced to future work.

Chapter 2

Background and state of the art

This chapter extensively details the state of the art of this project. It refers works related to the one defined in 1.2 and explains the main theoretical fundamentals needed to understand the problem.

In order to achieve the proposed goals, an automatic system that recognizes and identifies the food present in a meal is necessary. Image classification is a problem of pattern recognition whose aim is to label images into one of a number of predefined classes or categories, which has been previous learned by the model in the training phase [SAM19]. This form of learning is commonly referred to as **Supervised Learning**, because the training phase includes pre-labelled inputs.

This machine learning technique can be complex, depending on the data distribution in the training dataset [Mor19] and to train the model, food images and the corresponding labels are used as input. An issue that may affect the overall accuracy of the model is the visual variability that exists in digital images due to different factors when capturing the photograph, including the angle at which the picture was taken, the lighting conditions, the background of the image, or the color distribution. Besides that, classifying foods in an image is a very challenging task because foods are non-rigid objects that can have multiple forms, and as a consequence there is also a large intra-class diversity, for example, the appearance of boiled eggs and scrambled eggs is very distinct. In addition, the quantity and quality of the images used as well as the selected features are factors that influence the accuracy of the classifier.

According to the survey of Subhi et al., the process by which image feature extraction is conceived guarantees the existence of two strategies to develop a classifier: traditional classifiers with hand-engineered features and deep learning approaches [SAM19].

2.1 Traditional machine learning approaches

The traditional approaches of image classification require the application of a sequence of three methods: segmentation, feature extraction and classification.

2.1.1 Food image segmentation

The principle of **segmentation** states that the pixels in the image that share certain visual characteristics perceptibly distinguishable to the human eye are grouped. Although segmentation is an arduous task, it allows to identify food items in the image and exclude other elements such as food containers and background, which are not relevant for the analysis. Segmentation, when done well, plays a crucial role in the overall performance of the system, mainly if multiple food items have to be identified within a single image or if volume content has to be extracted [SAM19].

The main approaches to perform image segmentation [GH95] are designated thresholding, edge-based methods and region-based methods.

- **Thresholding** - in many applications of image processing, the grey levels of pixels belonging to an object are substantially different from the grey levels of the pixels belonging to the background. Therefore, thresholding becomes a simple but effective way to separate objects from the background. Sometimes, the threshold, t, is manually chosen by the scientist, which tries a range of values and verifies which one is best suited to identify the objects of interest. In the input image, the values of the pixels range from 0 to t, or from $(t + 1)$ to 255 and according to the selected threshold, the output of applying this method is a binary image.

- **Edge-based segmentation** - this method requires the application of an edge filter to the image using different operators and depending on its output, pixels are classifed as *edge* or *non-edge*. Edge-based segmentation treats an image as a weighted, undirected graph and formulates segmentation as a graph partitioning problem. Each pixel is a node in the graph with an edge connecting every pair of pixels. The weight of an edge measures the similarity between the pixels. The image is then divided into disjoint sets (segments) by removing the edges that link some of the segments. The lower the weight of the removed edges, the better the partitioning of the graph [ZBS+11].

- **Region-based segmentation** - region-based segmentation proposals are iterative algorithms where the pixels that correspond to an object are grouped together and detached from the remaining pixels.

In Table 2.1 a brief description of each technique of image segmentation is made, as well as the enumeration of its main advantages and disadvantages. Some of its content can be consulted in [TKP16].

Zhu et al. [ZBK+15], executed multiple segmentation hypotheses by assigning a class label to each pixel in an image. The first step was to detect regions of interest in the image, since, as they had found in an older paper, it is more efficient to first detect regions where potential food items are located instead of segmenting the entire image [ZBS+11]. Then, they used the classifier results as feedback to the segmentation, and assigned confidence scores to each segment. They have shown that their method outperforms the normalized cut framework [JM00] without classifier feedback.

Table 2.1: Advantages and disadvantages of image segmentation techniques

Segmentation Technique	Method Description	Advantages	Disadvantages
Thresholding	Depends on the histogram of an image.	• No need for contextual information. • Reduced data complexity. • Can be used in real time applications.	• It does not work well, if an image is composed of many edges or if they do not fit for flat valleys. • Fail to retrieve spatial information of an image, cannot guarantee that the segmented regions are contiguous. • Hypersensitive to noise.
Edge-based segmentation	Based on the variation of the intensity of the pixels or in its discontinuity.	• Easy edge detection. • It performs well in images that have a good contrast between regions.	• Usually, the boundaries determined are discontinuous.
Region-based segmentation	It groups the pixels that have similar properties and form the region.	• Provides better result in comparison with other segmentation methods. • Flexible when choosing between interactive and automatic technique for image segmentation.	• This technique consists of dual segmentation which takes time and memory. • It is hard to define the stopping criterion for segmentation.

2.1.2 Feature extraction

With the traditional approach, **feature extraction** is performed manually by observing the visual features of the images. In the case of food classification, items vary in color, shape and texture, so the extracted features must relate to these three characteristics. Looking back to the example of boiled eggs and scrambled eggs, its method of food preparation and composition are different, so we must pay attention to different distinguishing features.

Features are categorized into two main groups, local or global features. Local image features (also known as interest points or key points) can be defined as a specific pattern or distinct structure of an image, such as a point, edge, or small image patch (that differs from its immediate neighborhood by intensity, color, or texture) [SQ17]. On the other side, global features have the ability to generalize an entire object with a single vector.

The term extractor refers to the algorithm or technique that detects, or extracts these features and arrange them to be passed to another processing stage that describe their contents, for example, a feature descriptor algorithm. Descriptors encode a local pixel neighborhood into a compact vector representation. This information is invariant under image transformation, such as scaling or orientation changing, therefore allows the comparison between neighborhoods [Mat16].

Descriptors, such as SIFT or SURF, rely on local gradient computations. Binary descriptors, such as BRISK, ORB or FREAK, rely on pairs of local intensity differences, which are then encoded into a binary vector [Mat16].

Bosch et al. show that combining global and local features allows to capture more discriminate visual information for each food item and improve the accuracy of the recognition system [BZK+11]. This theory is confirmed in a later experiment [ZBK+15], where they use color, texture and SIFT features.

He et al. [HXK+14] also describe a food image by combining local and global features. Their experiment also used color and texture features. For the color features, the authors applied the descriptors Scalable Color Descriptor (SCD) and Dominant Color Descriptor (DCD). To extract the texture-based features, they used the descriptors Entropy-Based Categorization and Fractal Dimension Estimation (EFD) and Gabor-Based Image Decomposition and Fractal Dimension Estimation (GFD). They also investigated local region features, by applying the SIFT descriptor.

2.1.2.1 Circle Hough Transform

The Hough Transform was used to implement an algorithm to find a plate in an image and it is a feature extraction technique used in image processing.

The original Hough Transform was designed to identify lines in the image [DH72], but its use was extended to identify positions of arbitrary shapes such as circles or ellipses [Bal81].

The application of the Hough transform to detect circles in an image requires an array of three-dimensional accumulators. This array is indexed by three parameters that specify the location and size of a circle with the center $a = (a_1, a_2)$ and radius r. Considering that $p(x)$ is an image resulting from the detection of a contour and $x = (x_1, x_2)$ is the pair of coordinates of a pixel. For each x

there is a set of circles C_x, which pass through x. Considering also that X_p is the set of points $\{x|p(x) \neq 0\}$.

The algorithm suggested by Duda et. al [DH72] finds the center $a(x)$ and the radius $r(x)$ for each member of C_x, such that $x \in X_p$ through the geometric representation 2.1.

$$(x-A)^2 + (y-B)^2 = C^2 \tag{2.1}$$

The resulting set of center-radius pairs is denoted by $\{(a,r)|x \in X_p\}$. For each member of this set, an accumulator in $(a(x), r(x))$ in the space ab is incremented by one.

After all X_p members have been processed, the accumulator in $(a(x), r(x))$ contains the number of X_p elements that are over the circle of radius $r(x)$ with center in $a(x)$.

2.1.3 Algorithms for image classification

In the literature plenty of classification algorithms are proposed that follow different principles. According to these principles, the algorithms can be categorized as: distance-based algorithms, probability-based algorithms, search-based algorithms, and optimization-based algorithms [Mor19].

- **Distance-based algorithms** - they predict the class of a new object by seeing how similar this object is to others, previous labeled, objects.
- **Probability-based algorithms** - a probabilistic classifier is a classifier that model the statistical relation between the predictive attributes and the target attribute of a data set.
- **Search-based algorithms** - are algorithms that use other algorithms themselves to perform iterative local searches with the purpose of inducing the best predictive model, a local optimum.
- **Optimization-based algorithms** - are used to find the best available alternative under the given constraints and are used to solve optimization problems.

Among the algorithms most used in image classification, K-NN and SVM are the most outstanding.

K-Nearest Neighbor (K-NN) is a distance-based learning algorithm and is one of the simplest classification algorithms. Since K-NN does not have an explicit learning phase and memorizes the training objects, it is based on lazy learning. When the algorithm has to predict a class of a new object, it does so through a local-learning approach, by identifying the class of the k objects more identical to the new object.

Support Vector Machine (SVM) was originally designed for regression and binary classification tasks and belongs to the category of optimization algorithms. An SVM model is a representation of the objects as points in space, mapped so that the objects of the separate categories are divided by a clear border that is as wide as possible. New objects are then mapped into that same space and predicted to belong to a category based on which side of the border they fit.

In the study of He et al. [HXK+14], the color and texture features were classified using the K-NN algorithm, while the local region features were classified through the use of a vocabulary tree classifier.

After applying the multiple segmentation techniques mentioned above, Zhu et al. [ZBK+15], performed vocabulary construction and signature formation through the bag of features model. To classify the images, the authors used K-NN and SVM and noticed that the first one achieved better accuracy, 70%, in contrast with the SVM classifier that only achieved an accuracy of 57%.

2.2 Deep learning approaches

Although calculating the correct features as well as representing them in the best way are very challenging tasks, they can become easier if good features are learned automatically. This is the main advantage offered by deep learning techniques [HMC+16]. According to LeCun et al. [LBH15], deep learning “allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction". In consequence, the process of feature extraction is adopted automatically through a series of connected layers followed by a fully connected layer which is responsible for the final image classification.

The emergence of these methods has brought advantages over classic learning algorithms and improved the state-of-the-art of image recognition. The main advantage is that learning distributed representations enable generalization to new combinations of the features learned at the training phase. In fact, in [CBD+90] it has been proven that image recognition can be done without the need of a very complex preprocessing stage and that a learning network can be fed directly with images instead of feature vectors.

Bossard et al. [BGVG14] compared their model based on Random Forests [1] against several state-of-the-art methods on food recognition and realized that it outperforms all, except for the CNN (whose network architecture was proposed in [KSEH12]).

Convolutional Neural Networks require a large dataset to build a classification model and have been widely used in the field of food recognition and classification. In 2017, Ciocca et al. merged several datasets publicly available: Food50, Food-101, UECFOOD-256, and VIREO [JY09, BGVG14, KY15, CN16] resulting in the Food524DB dataset composed of 524 different food classes. The authors build a network architecture based on the residual network ResNet-50 to obtain the CNN-based features from the Food524DB dataset. To evaluate the designed network, they used the UNICT-FD1200 dataset [FAM+16] and the learned features “greatly outperformed the handcrafted” ones [CNS17b] .

In 2019, Zhang et. al [ZZG+15] compared the application of a CNN, with and without data expansion techniques, with a conventional Bag of Features (BoF) model combined with linear SVM, for food image recognition, in a ten-class small-scale food image dataset collected from

[1] Random forests use decision trees as the base learner and each decision tree is created using a different bootstrap sample.

ImageNet [2]. The first experiment, BoF combined with SVM, resulted in the accuracies [3] of 68% for training and 56% for testing images. However, these results could be improved if the extraction of SIFT descriptors had been done in colored images and not only in grayscale images, as was the case. In the second experiment, a CNN was implemented without data expansion techniques, resulting in an accuracy of 74% for testing images and a training accuracy 95%. Finally, building a CNN with data expansion techniques led to the reduction of overfitting and the increase of the overall test accuracy.

2.2.1 Convolutional neural networks

In 1968, Hubel and Wiesel's [HW62] work inspired the Convolutional Neural Networks architectural design which is based on primate's visual cortex structure. CNNs were introduced by LeCun et al. [CBD+90] and its emergence allowed a boost in the field of computer vision. In terms of image processing, they have been applied to the segmentation, detection and recognition of objects and regions in images. Nowadays, most of the frontrunners of image processing competitions, like Google, Microsoft, and Facebook are employing deep CNN based models and exploring new architectures of CNN [DY14].

The architecture of CNN is composed by a combination of multiple learning stages, such as convolutional layers, subsampling layers, and non-linear processing units [JKRL09]. Each layer performs multiple operations using a bank of convolutional filters (kernels). The convolutional layer extracts correlated local features by dividing the image into smaller parts. The kernel's output is assigned to non-linear processing units that ensures non-linearity in the feature space and helps in learning abstraction. Besides that, this non-linearity generates dissimilar patterns of activations for different responses and facilitates the learning of semantics differences in images. After the phase of the non-linear function follows the subsampling phase, which also guarantees the invariance of the input to geometrical distortions [KSZS19].

CNN has been widely used due to its hierarchical feature extraction ability: it extracts low, mid, and high-level features. The high-level features (more abstract) are a combination of the previous ones. This ability makes the CNN learn a good internal representation from raw pixels with reduced processing.

Summing up, a typical CNN architecture is usually an alternation between convolutional and pooling layers, followed by one or more fully connected layers at the end. Sometimes, the fully connected layer is replaced with global average pooling layer. In addition to the learning phases, some different regulatory units such as batch normalization and dropout are appended to improve the performance of the Convolutional Neural Network.

[2] http://imagenet.org/
[3] ratio between the number of images and the total number of images

2.2.1.1 Convolutional layer

A set of convolutional kernels (each neuron act as a kernel) compose a convolutional layer. The convolutional layer divides the image into small regions known as receptive fields and convolves them with a specific set of weights. The division helps in extracting locally correlated pixel values and this information locally aggregated is also known as feature maps.

As shown in Figure 2.1, the filter moves all over the image and its task is to multiply its values by the original pixel values. All the multiplications are summed up to one number in the end. The equation 2.2 explains the convolution operation: $I_{x,y}$ represents the input image, x,y shows spatial locality where K_l^k represents l^{th} convolution kernel of the k^{th} layer.

$$F_l^k = (I_{x,y} * K_l^k) \tag{2.2}$$

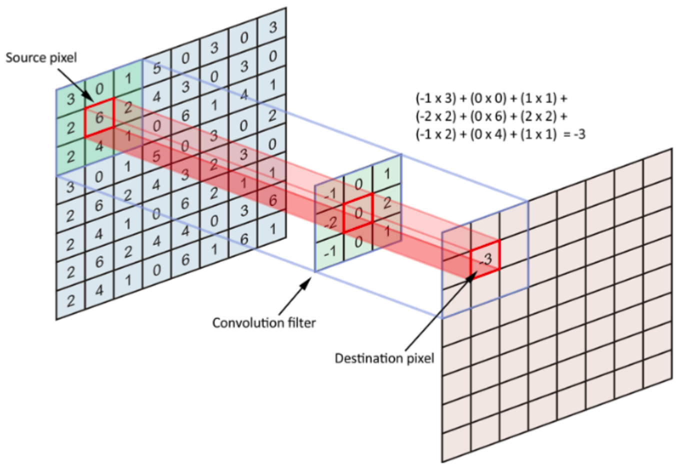

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel

Figure 2.1: Convolution operation[4]

The objective of the convolution operation is to extract features from the input image.

This operation may be further characterized into different types, taking into account the type and size of the filters, type of padding, and the direction of convolution [LBH15].

2.2.1.2 Pooling layer

The output of convolution operation, feature maps, can occur at different locations in the image. After the features are extracted, its exact location becomes less important as long as its approximate position relative to others is preserved.

The pooling layer performs downsampling along the spatial dimensionality of the input, reducing the number of parameters within that activation and the computational effort required in the learning phase.

$$Z_l = f_p(F_{x,y}^l) \tag{2.3}$$

[4] https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

The equation 2.3 defines the pooling operation. Z_l represents l^{th} output feature map, $F^l_{x,y}$ shows l^{th} feature map, where f_p is the pooling function. This operation helps to control the overfitting and decreases the feature map size, while at the same time keeps the important information.

There are several types of pooling formulations for extracting translational invariant features, such as max, average, L2, and overlapping pooling. In Figure 2.2, the max pooling was used, thus the extracted values are the largest ones.

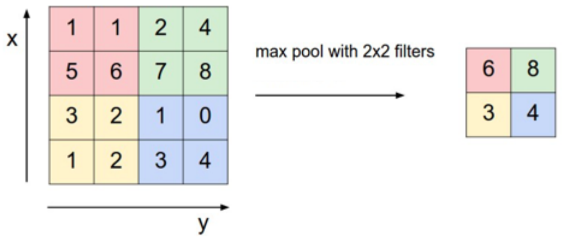


Figure 2.2: Pooling operation[5]

2.2.1.3 Activation function

"Activation function serves as a decision function and helps in learning a complex pattern" [KSZS19]. The learning process can become faster if a good activation function is achieved. In equation 2.4, an activation function for a convolved feature map is defined.

$$T^k_l = f_A(F^k_l) \tag{2.4}$$

F^k_l is an output of a convolution operation that is assigned to the activation function. $f_A(.)$ is the function that adds non-linearity and returns a transformed output T^k_l for k^{th} layer. There are different activation functions in the literature, such as sigmoid or tanh, but ReLU (Rectified Linear Unit) is the most used.

2.2.1.4 Batch normalization

Batch normalization is used to normalize the inputs of each layer, in order to address the internal covariance shift problem within feature maps. The internal covariance shift is a change in the distribution of hidden units' values, which slow down the convergence (by forcing learning rate to small value) and requires attentive initialization of parameters.

$$N^k_l = \frac{T^k_l}{\sigma^2 + \sum T^k_i} \tag{2.5}$$

Batch normalization for transformed feature map T^k_l is shown in equation 2.5. N^k_l is the normalized feature map, where T^k_l is the input feature map and σ illustrate variation in the feature map.

[5] http://cs231n.github.io/convolutional-networks/

2.2.1.5 Dropout

Dropout is a technique where randomly selected neurons are ignored during training. As result, the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data.

2.2.1.6 Fully connected layer

The fully connected layer is used at the end of the network for classification purpose. It takes the input from the previous layer and analyses the output of all previous layers globally. It makes a non-linear combination of selected features, which are used for the classification of data. Unlike pooling and convolution, it is a global operation [RW17].

Finally, image 2.3 is a representation of a complete Convolution Neural Network, where the convolution and pooling layers act as feature extractors from the input image and the fully connected layer acts as a classifier.

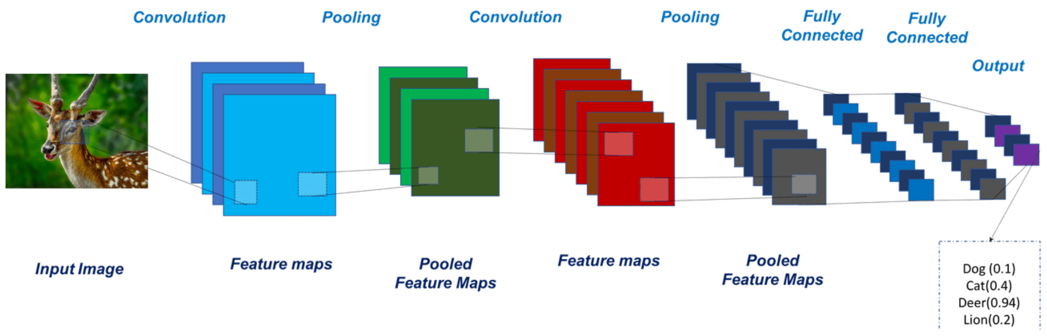


Figure 2.3: Convolution neural network to classify an input image into one of the predefined categories [6]

2.3 Food volume estimation

Estimating the volume of the food present in an image is a crucial task, since without it, it is not possible to estimate the nutrient content, like carbohydrates, sugars or calories.

The computer vision-based food volume estimation techniques can be divided into several types of approach: model-based approach, stereo-based approach, depth-based approach, and deep learning approach [LSQL18]. Some approaches that are model-based are listed next:

- Sun et. al [SBB+15] based their method on a virtual reality approach that uses a 3D wire mesh of known shape and volume for later being adapted to the real food item by rotating, translating and scaling it.

- Xu et. al [XHK+13] reconstruct the 3D model of the food from a video sequence or 15-20 images by using a variation of the method called *Shape from Silhouettes* [KS00]. However,

[6] https://vinodsblog.com/

in their study, only 5 categories of food with regular shapes are considered: banana, bagel, orange, orange juice and rice krispy treat.

- Zhu et. al [ZBW+10] proposed a spherical and a prismatic approximation models to estimate the food item volume. When dealing with spherical food items, the feature points are projected onto the plane surface and the volume is estimated. When the food has irregular shapes, the prismatic approximation model is used. Once again, almost all of the classes used in this experiment have regular shapes: grapefruit, apple, nectarine, orange, plum, and brownie.

On the one hand, the model based approaches have high performance in determining the volume for the classes in which have been trained. In the other hand, this technique is unable to get good volume estimates in irregular food shapes, it requires a lot of manual refinement, and each model is trained on few classes.

In the works presented below, all of them follow a stereo-based approach:

- Puri et. al [PZY+09] use three single 2D images taken at different positions above the plate. To determine the relative camera positions, they perform extrinsic calibration by using the RANSAC algorithm. After that, dense stereo reconstruction is performed and a checkerboard is used to extract the scale. The main drawback of this approach is the long time that the 3D model reconstruction takes.

- Dehais et. al [ADS+15] followed a similar approach, but they tried to use only two images to speed up the process and use a card as reference. A modified RANSAC algorithm that includes local optimization was applied and the volume estimation time is roughly 5 seconds.

- Gao et. al [GLL18] make their experiments by applying a wearable camera and used the convex hull algorithm to form the 3D mesh object. The volume is then estimated through the mesh object, obtained from 23 pictures of different angles.

The main advantages of applying stereo-based techniques include: the ability to handle irregular food items and no need for prior knowledge and pre-built 3D models. The limitations common to this technique are the need of using multiple images captured from different angles, the volume estimation becomes too slow due to feature detection and matching and a fiducial marker is always required to extract the scale.

Deep learning approaches consists of training models from depth images, captured with depth-cameras, to predict the depth in RGB images. This approach is followed by Myers et. al [MJR+15] and Fang et. al [FZJ+16].

In cases where a 2D image is the only input of the system, estimating the volume of the food in the image can be a very difficult task, since there is no other additional information of the real world as the scale or the depth of the objects in the scene. Model and stereo-based techniques both require more than one input image to estimate the volume and collecting the input images involve much levels of human intervention. Therefore, designing a system to estimate the food volume from only one image can be done by using image depth map estimation methods.

2.4 Smartphone applications

As previously seen, there is a large number of researchers trying to find solutions to the problem of recognizing food images and estimating their volume. The number of mobile applications that support the problem has also increased and this section will describe how they were developed.

In 2014, Pouladzadeh et al. proposed a personal instrument to measure the intake of calories and nutrients through a smartphone. The developed system uses image processing and segmentation to identify portions of food (i.e., isolating portions such as chicken or rice from the overall food image), estimate the volume of each portion and finally to calculate their nutritional values. For each detected food portion, a feature extraction process was performed and various food features including texture, color, shape, and size were extracted. At this point, a Gabor filter-bank proposed in [JF91] was applied for texture segmentation of the food images. Then, the extracted features were used in an SVM to perform the classification task. “For increasing the accuracy, after the SVM module has determined each food portion type, the system can optionally interact with the user to verify the kind of food portions". To estimate the volume of each portion of food, the thumb that is present in the image is used as a reference. For the evaluation, the method developed was applied to three different food categories: single, nonmixed, and mixed foods, and the accuracy of the SVMs is approximately 92.21%, 85% and 35-65%, respectively [PSA14].

Recently, in 2015, Nagapavan et al. proved the feasibility of a BoF-based system for the food recognition problem. They performed their experiments with one set of 5000 colored food images, distributed in 11 dissimilar food classes. In order to select the key points of the image, the authors used descriptors based on SIFT. As a result, a 128 dimensional feature vector was calculated to classify RGB values. In the first experiment, the key points were extracted using three methods: SVM, ANN and RF. In the second experiment, the goal was to evaluate if the difference in size of the descriptors would have an impact on the final performance. The best result was achieved by the descriptor combination with sizes of 16, 24 and 32. The developed system, in which the target was diabetes patients, reached an accuracy of 78% [NMLJ15]. The system developed by these authors only classifies images with one type of food and if a data set were used in which an image was composed of several foods, the results obtained would be even more promising after applying a segmentation method.

In 2016, He et al., presented an automatic food classification method - DietCam, whose main components are ingredient detection and classification of ingredients combination. They conceived a novel food ingredient detector with a part-based model of locations and textures. Part-based models consider the shape of each part of the object and their geometric relations to detect and classify rigid-shaped objects. However, these models can not be directly applied to the detection of ingredients due to the variety of textures and shapes present in the food. The authors integrated texture filters into part-based detectors, where textures were classified in texture filters.

As a technique of segmentation and classification, Semantic Texton Forests (STF) was applied to detect the food texture. In addition to developing a new model that detects the ingredients, the authors also developed multikernel SVM to classify various combinations of food ingredients under occlusion. By employing a hierarchy of kernel functions, a classifier that detects and classifies food of different scales and points of view was accomplished. Based on the experiment developed with around 15000 food images, the precision of DietCam was around 90% for general food items, 85% for food that has a dominant plus small ingredients and foods composed by small ingredients many times, 60% for a food image composed by a single dominant ingredient [HKT16]. As the composition of food became more complex the acuracy dropped to 30%.

Anthimopoulos et al. [ADS$^+$15] developed a smartphone application called GoCARB to help type 1 diabetics count the carbohydrates present in the meals they eat. The users need to input two photographs to the system taken from the food dish at specific locations predefined by the application. Part of this work was promoted by the Bern University Hospital "Inselspital", that yielded to the authors the possibility of acquiring images of food in a controlled environment, with a card positioned near the plate. Their first step was to detect the plate and to this end, they used an ellipse detector based on the Canny edge detector. After the plate has been detected, the image is cropped to the plate area and converted to the CIELAB color space (that approximates to the human vision). It is relevant to point out that the food items in the image do not need to be physically separate, but they cannot overlap in order to not affect the subsequent volume estimate. The segmentation step was performed though pyramidal mean-shift filtering [CM02] and achieved an accuracy of 88.5%. In the recognition phase, a hierarchical version of the k-means algorithm was applied to cluster the color space created by the training set of food images. After the combination of color and texture features, a vector of 1280 dimensions was created and fed to a nonlinear SVM with a radial basis function kernel. The algorithm chosen for the image classification was the Support Vector Machine, because it had better performance than Artificial Neural Networks (ANN) and Random Forests (RF) as the authors verified in [AGS$^+$14]. The used classifier had an overall accuracy in the range of 87-90% for different datasets and 9 food classes. The last step was to estimate the food's volume, and for that its 3D shape was reconstructed using passive stereo vision. The volume estimation method achieved a mean absolute percentage error of 9.4%. In order to check if the integrated system was feasible, 24 different food dishes were analyzed and the overall mean absolute error of the system in CHO counting was 6 grams, and in fact, the obtained result is excellent.

In 2016, Ciocca et al. [CNS17a] collected new food images and built UNIMIB2016 Food Database. The process starts with the tray segmentation divided into two parts (a saturation-based one and a color texture one), to detect the regions of interest. They concluded that the location of food regions is well achieved using the saturation channel because food items have saturation values higher than the plate regions, the tray, and the cutlery. Some regions, other than food, have similar saturation levels, so to differentiate them, the JSEG segmentation algorithm [DM01] (that performs well on both color and texture features) was used. The authors intended to evaluate how the segmentation process affects the classification process, hence several visual descriptors were compared. The final result show that “the CNN-based visual descriptors achieve better results than others in all the classification strategy" and “the SVM classifier performs slightly better than K-NN with an accuracy of 78.9%".

In 2017, Tendee et al. [TU17] created a Thai food dataset with images collected from the Internet distributed by 40 food classes. The authors decided to use a CNN to build a system that performs image recognition, since CNNs can be applied to small datasets by using a pretrained model for a large image dataset and retraining it in a smaller dataset (transfer learning). There are some learning parameters, such as learning steps, random brightness, random scale and random crop, however, Tendee et al concluded that only the first two parameters were important.

Table 2.2 compares the detailed mobile applications in 2.4 in relation to: the used dataset, the approach used for the image segmentation, the extracted features, the approach used in the image classification, the approach chosen for the volume estimation of the foods present in the image and the performance of the application developed.

Table 2.2: Comparison between mobile applications that recognize food and estimate its volume in images

Application	**Dataset**	**Segmentation Approach**	**Features**	**Classification Approach**	**Volume Estimation Approach**	**Performance**
Measuring Calorie and Nutrition From Food Image [AVPS12], [PSA14], [VAPS12]	approximately 3000 images	Graph cut segmentation	Color Shape Size Texture	• Support Vector Machine	Usage of the thumb of the user present in the photograph to calculate the area and volume of the different items of food present in the image.	Overall segmentation accuracy of 95% Classification accuracy: • 92.21% for single food • 85% for nonmixed food • 35-65% for mixed food
Large Scale Learning for Food Image Classification [NMLJ15]	approximately 5000 images belonging to 11 classes	Descriptors based on SIFT	Color Texture	• Support Vector Machine • Artificial Neural Network • Random Forest	Without food volume estimation.	Overall classification accuracy of 78%
Dietcam: Multiview food recognition using a multi-kernel svm [HKT16]	15 262 food images belonging to 55 classes	Development of a food ingredient detector with a part-based model of textures and locations	Texture - STF Shape Color	• multiview multikernel-based SVM	Without food volume estimation.	According to the complexity of food composition, the accuracy obtained has dropped from 90% in the simplest items to 30% for more complex items.
Computer vision-based carbohydrate estimation for type 1 patients with diabetes using smartphones [ADS+15]	• 3800 food images downloaded from the Internet • 1620 images of food prepared at the Inselspital's restaurants 9 classes	Ellipse detector based on the Canny edge detector	Color Texture	• nonlinear Support Vector Machine	The input of the system, was a pair of images from different viewing angles, both including a reference card. 3D shape reconstruction using passive vision stereo. A process called polar rectification is used to match the points between the 2 images.	Segmentation accuracy of 88.5% Classification accuracy of 87% - 90%
Food recognition: A new dataset, experiments, and results [CNS17a]	UNIMIB2016 (1027 food images belonging to 73 classes)	A combination of color, saturation, and JSEG. Several visual descriptors have been compared.	Color Texture	• Support Vector Machine • K-Nearest Neighbors	Without food volume estimation.	The SVM classifier performs better than k-NN with a tray accuracy of 78.9%

Most of the approaches followed by the authors of the works that were presented throughout this Chapter follow the traditional machine learning approach to classify food images. However, currently, there are vast deep learning approaches with excellent results in the task of image recognition and there is a lack of exploring them with regard to the recognition of food images.

A great contribution of this work is to explore image recognition networks that have been developed recently and that have boosted the computer vision field of image processing: Mask R-CNN, RetinaNet, and Deeplab.

Chapter 3

Deep learning models for semantic segmentation and depth estimation

In this Chapter, the theoretical concepts that underlie the design of the neural networks used in the implementation of the project are described as well as the most important constituents of each architecture. Therefore, the description of Mask R-CNN is in Section 3.3, the description of RetinaNet is in Section 3.4, and the description of Deeplab is in 3.5. In turn, these networks use as a backbone different networks: or ResNet or Xception. A little reference is made to them in Section 3.1 and Section3.2, respectively. Finally, some aspects of the implementation that allowed us to generate depth images from RGB images are described in 3.6.

3.1 ResNet

Deep learning is fundamental in the feature extraction process because of the need to learn low, mid, and high-level features. When dealing with image processing, this is analogous to learn, for example, edges, shapes and objects. Theoretically, if more layers are used, the features extracted will be more valuable.

However, He et. al, in their paper called Deep Residual Learning for Image Recognition [HZRS15] show that simply stacking more convolutional layers do not improve the network's learning.

With the increase of number of layers, or depth increasing, the accuracy of the network gets saturated and then downgrade quickly. Yet, this degradation problem is not caused by over-fitting. As can be seen in Figure 3.1, two plain networks with different number of layers (20 and 56) are being compared. On the right side of the figure, in the test error graphic, the 56 layer network is performing worse than the 20 layer network, which means that a deeper network was not able to perform better. However, when looking to the training error on the right, the 56 layer network also perform worse than the 20 layer network and for this reason, it can be concluded that the

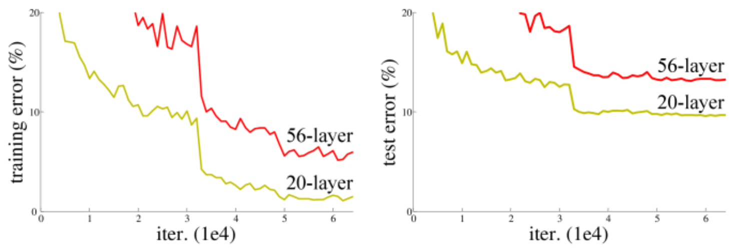


Figure 3.1: Training error (left) and test error (right) with 20-layer and 56-layer "plain" networks [HZRS15]

degradation problem is not due to over-fitting, otherwise the training error on the 56 layer network would be much lower.

He et. al considered that this is actually an optimization problem, that is, deeper models are harder to optimize than simple shallow networks. Considering a shallower architecture and its deeper counterpart that adds more layers onto it, the deeper model should be able to perform at least as well as the shallower model. The deeper model could be constructed by copying the learned layers from the shallower model and setting additional layers to identity mapping. To achieve this purpose, they introduced a deep residual learning framework: instead of hoping that each stacked layers directly fit a desired underlying mapping, they made a block of layers fit a residual mapping.

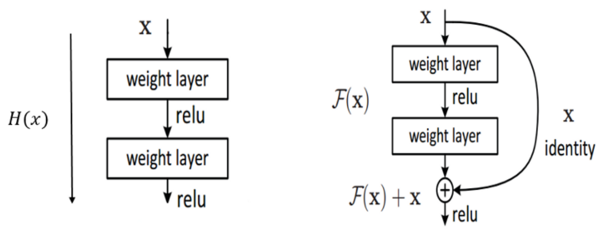


Figure 3.2: Direct mapping and residual mapping [HZRS15]

Looking at Figure 3.2, the desired underlying mapping was denoted by $H(x)$ and the stacked layers on the right side of the figure fit a residual mapping denoted by $F(x) = H(x) - x$ instead of $H(x)$ directly. The desired underlying mapping can be rewritten as $H(x) = F(x) + x$.

In the best scenario, if an identity mapping was optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers, which means that would be easier to optimize the residual mapping function $F(x)$ than the original one $H(x)$.

3.1.1 Architecture

A ResNet network consists of an input stem, four subsequent stages and a final output layer, which is illustrated in Figure 3.3. The input has a 7x7 convolution with an output channel of 64 and a stride [1] of 2, followed by a 3x3 max pooling layer also with a stride of 2.

[1] Stride is the amount by which the kernel is moved by as the kernel is passed over the image.

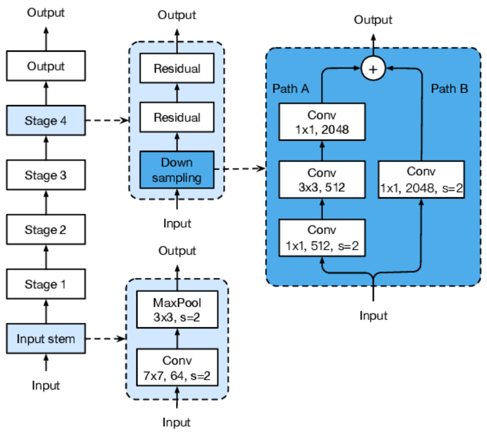


Figure 3.3: Architecture of ResNet-50 [HZZ+18].

The input width and height are reduced by 4 times and the channel size is increased to 64 by the input stem. From the second stage, each stage begins with a down-sampling block, followed by several residual blocks. In the down-sampling block, there are two paths, A and B. The first one has three convolutions, whose kernel sizes are 1x1, 3x3 and 1x1, respectively. The first convolution has a stride of 2 to halve the input width and height, and the last convolution's output channel is 4 times larger than the previous two, which is called the bottleneck structure. Path B uses a 1x1 convolution with a stride of 2 to transform the input shape to be the output shape of path A, so we can add the outputs of both paths to obtain the output of the down-sampling block.

A residual block is similar to a down-sampling block except for only using convolutions with a stride of 1.

By varying the number of residual blocks in each stage, different ResNet models are obtained, such as ResNet-50 and ResNet-101, where the number presents the number of convolutional layers belonging to the network.

To get more detailed information about ResNet architectures, please refer to [HZRS15].

3.2 Xception

Xception was born in 2016 by Google and it is the backbone network of our implementation of the Deeplab network 5.4. Xception stands for *Extreme Inception* and as the name implies, it takes the principles of Inception network to an extreme. The assumption behind Inception is that "cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly". In the traditional CNNs, the convolution layers seek out correlations across both space and depth (represented by the RGB channels), but it may be not necessary to consider the image region and the channels at the same time.

Thus, Inception uses a $1x1$ convolution to project the original input into several separate smaller spaces and then, all the correlations of the smaller spaces are mapped through $3x3$ or $5x5$ convolutions. In its turn, instead of partitioning input data into several compressed chunks, Xception maps the spatial correlations for each output channel separately, and then performs a $1x1$ convolution to capture cross-channel correlation [Xu17].

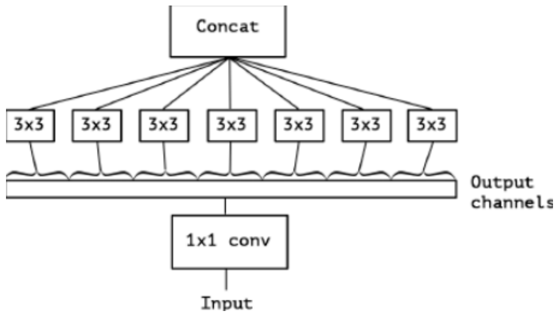


Figure 3.4: Extreme Inception module with one spatial convolution per output channel of the $1x1$ convolution [Cho16]

Figure 3.4 shows the Extreme Inception module that is very similar to a depthwise separable convolution which is composed by a depthwise convolution (a spatial convolution applied to each channel, separately) followed by a pointwise convolution (a $1x1$ convolution across channels).

3.3 Mask R-CNN

Masked Region based Convolution Neural Network (Mask R-CNN) was released in 2017 by Facebook Artificial Intelligence Research (FAIR). Mask R-CNN works towards the problem of **instance segmentation** - the process of detecting and delineating each distinct object of interest in an image. In turn, instance segmentation is a combination of two sub-problems: object detection and semantic segmentation. Object detection is the problem of finding and classifying a variable number of objects in an image - variable because the number of detected objects in an image can vary from image to image. The second sub-problem, semantic segmentation, consists in the understanding of an image at the pixel level, in other words, it assigns an object class to each pixel of the image.

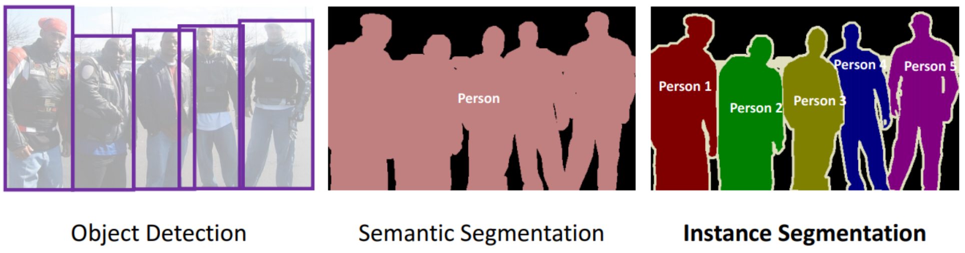


Figure 3.5: Combining Object Detection with Semantic Segmentation results in Instance Segmentation [2]

[2] https://medium.com/@yanfengliux/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef

In Figure 3.5, it can be seen that the bounding boxes provided by object detection together with the shaded masks generated by semantic segmentation result in object instance segmentation, allowing marking two objects of the same class separately.

Since instance segmentation is performed in two phases, Mask R-CNN architecture is also based on two frameworks: Fast/Faster R-CNN and Fully Convolution Network (FCN) for object detection and semantic segmentation, respectively.

According to He et al. [HGDG17], Faster R-CNN is a two-stage detector. The first stage is called Region Proposal Network (RPN) and it generates a set of bounding boxes (also called proposals) which have the higher probability of an object being present. Those proposals are then passed through a detection network (which is in essence Fast R-CNN) that extracts features from each proposal and performs classification and bounding box regression.

Mask R-CNN adopts the same two-stage procedure, however it differs from Faster R-CNN because in the second stage, parallel to object classification and bounding box regression, it also outputs a binary mask for each RoI (Region of Interest). The existence of different branches running in parallel, one for object classification and bounding box regression, and the other to generate the masks makes possible that object's classification does not depend on the prediction of its mask. Both stages are linked to the backbone structure.

3.3.1 Backbone

The backbone contains a convolution neural network that is responsible for extracting features from the whole input image. Using this network as initial stage to extract features from the input image saves time as otherwise, the features would have to be extracted from each proposed bounding box. In [HGDG17], the authors used a ResNet architecture 3.1 as backbone, in combination with a Feature Pyramid Network [LDG$^+$17]. They experimented several ResNet approaches, but the better results were obtained by using a ResNet-101.

3.3.2 Feature pyramid network

The Feature Pyramid Network, that is used in Mask R-CNN, creates a top-down path-way connected by lateral connections. FPN obtains feature maps at different scales from a single-scale input. As we go down along the top-down pathway, the features extracted are lower-level and the spatial resolution increases. The top layers of this this top-down pathway have lower resolution which allows the network to perform well when detecting small objects and to obtain strong semantics in all pyramid levels.

The top-down pathway is done by up-sampling [3] the feature maps coming from higher levels and the technique used to upsample is called nearest neighbour interpolation. At every layer, the feature map size is reduced by half and the number of feature maps is doubled.

[3] Up-sampling is one image operation that increases the spatial resolution, but keeps the same two-dimensional representation [You07]

[4] https://medium.com/@fractaldle/mask-r-cnn-unmasked-c029aa2f1296

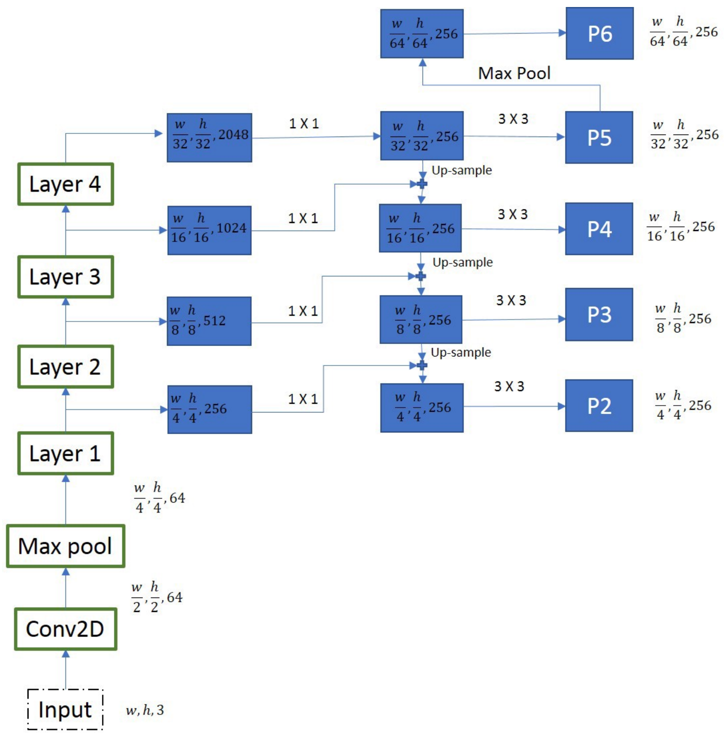


Figure 3.6: FPN feature extraction process [4]

In the example of Figure 3.6, features are being extracted from 4 feature maps (layer-1, layer-2, layer-3, and layer-4 outputs). The feature map of layer-2 is first subjected to 1x1 convolutions to bring down the number of channels to 256, this convolutions make the lateral connections. This is then added element-wise to the up-sampled output from the previous iteration.

All the outputs of this process are subjected to 3x3 convolution layer, to merge all layers in order to create the final 4 feature maps (P2, P3, P4, P5). The 5th feature map (P6) is generated from a max pooling operation from P5.

It is of uttermost importance to consider the following example: in 3.6, the size of the smallest feature map involved in the up-sampling operations is $(\frac{w}{32}, \frac{h}{32})$, where w is the image's width and h is the image's height. Therefore, it must be ensured that the input tensor has dimensions divisible by 32, otherwise an error will be thrown. For example, considering the case of $w = 800$ and $h = 1080$, as $\frac{w}{32} = 25$ and $\frac{h}{32} = 33.75$, the smallest feature map will be of size of $(25, 33)$. After the up-sampling operation, in the next iteration, dimensions would be $(50, 66)$. As $\frac{w}{16} = 50$ and $\frac{h}{16} = 67.5$, the element-wise addition between the feature map size outputted in the previous iteration $(50, 66)$ and the feature map size from the current iteration $(50, 67)$ is not possible.

3.3.3 Region proposal network

The output of the backbone, the feature map, is then used as the input to the Region Proposal Network. The goal of RPN is to generate the candidate bounding boxes or regions of interest (ROIs) whose probability of finding an object is higher.

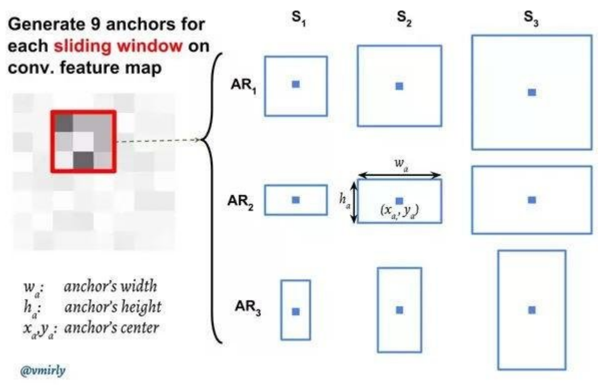


Figure 3.7: Each of the bounding boxes generated per anchor has different scales and aspect ratios [5]

To start, the feature map is scanned using a sliding window - a rectangular region of fixed dimensions. The center of the sliding window in each position is called the anchor. For each anchor, some bounding boxes are predefined to be used for reference when predicting the location of the different objects. The bounding boxes generated for each anchor vary in scale and aspect ratio as the Figure 3.7 illustrates. The authors of [HGDG17] use 5 scales and 3 aspect ratios at their implementation. Although anchors are defined over the feature map, they reference the original image.

When all bounding boxes are generated, the RPN takes them as input and performs two tasks, classification and regression, with each one of them. In the branch of classification, several classes (the classes that represent the objects to be detected on foreground and an extra class representing the background) are used to calculate the probability of an object being present in the anchor box. In the regression branch, the (x, y) coordinates of an anchor box point along with the width and height are predicted.

When to an anchor box is tagged a class that belongs to the foreground, the object that it contains may not be completely centered and the regression step takes this issue into account. At this point, the neighborhood anchor boxes have similar scores and are also considered as candidate regions, resulting in thousands of proposals that can't be all processed due to expensive computational processing. Besides that, the number of generated anchor boxes is simply uncorrelated with the real number of objects in the image [RGG14]. This leads to the need of applying a filter technique to reduce the number of anchor boxes to be processed. This technique is called Non-maximum Suppression (NMS).

3.3.4 Non-maximum suppression

The pseudo code of Non-maximum Suppression algorithm is presented in Figure 3.8. This method receives as input, a list of all proposed anchor boxes generated during the FPN execution, the corresponding confidence scores S, and the overlap threshold N.

[5] https://datascience.stackexchange.com/questions/27277/faster-rcnn-how-anchor-work-with-slider-in-rpn-layer
[6] https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

Algorithm 1 Non-Max Suppression

```
1: procedure NMS($B$,$c$)
2:     $B_{nms} \leftarrow \emptyset$
3:     for $b_i \in B$ do
4:         $discard \leftarrow$ False
5:         for $b_j \in B$ do
6:             if same$(b_i, b_j) > \boldsymbol{\lambda}_{\mathbf{nms}}$ then
7:                 if score$(c, b_j) >$ score$(c, b_i)$ then
8:                     $discard \leftarrow$ True
9:         if not $discard$ then
10:            $B_{nms} \leftarrow B_{nms} \cup b_i$
11:    return $B_{nms}$
```

Figure 3.8: Non-maximum Suppression algorithm [6]

At the beginning, the proposal with higher confidence score is selected to incorporate the list composed of the final proposals and is removed from the input list. After that, each proposal in the initial list is compared with the remaining ones by calculating the Intersection over Union (IoU). If the IoU is greater than the predefined threshold N, that proposal moved from the initial list to the list with the final proposals, otherwise it is removed from the initial list. This process lasts until there is no more proposals left in the initial list.

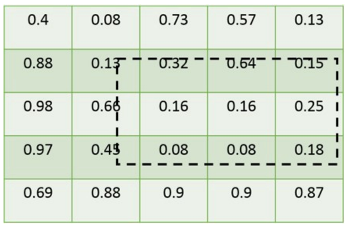
As can be seen, this whole filtering technique depends on the initial defined threshold, which means that the selection of the threshold value is crucial for the performance of the model. The default threshold value used in Mask R-CNN is 0.7.

3.3.5 RoI align

After the refined output anchor boxes being obtained by the RPN output, they are used as the input for the next step. To Mask R-CNN incorporate the ability of performing instance segmentation, some changes were made to its predecessor architecture (Faster R-CNN) - the RoI Pooling layer was replaced by a RoI Align layer.

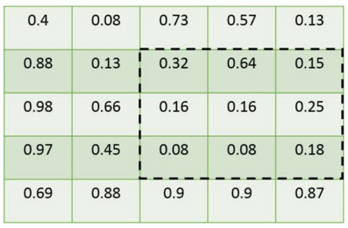
As the anchor boxes outputted from RPN have different sizes, they will generate feature maps with different sizes. In order to standardize the size of all anchor boxes, or the ROI Pooling operation or the ROI Align operation is applied.

For better understanding the difference between RoI Pooling and RoI Align, an example is shown in Figures 3.9 and 3.10. Starting with 3.9a, a feature map is represented with size of $(5,5)$ with the anchor box plotted on it. As can be noticed, the boundaries of RoI do not match with the granularity of the feature map (this is because the RoI has coordinates regarding to the original image and the feature map has a lower resolution than it). To make the boundaries of RoI to coincide with the feature map granularity 3.9b, they were aligned through a process called quantization. Depending on the size of the output (matrix M x M, for example), the ROI is divided into bins (the number of bins depends on M). Considering $M = 2$, max-pooling operation is applied to each bin separately like 3.9c and 3.9d shows. The values that result of this operation are $(0.32, 0.64, 0.16, 0.25)$.



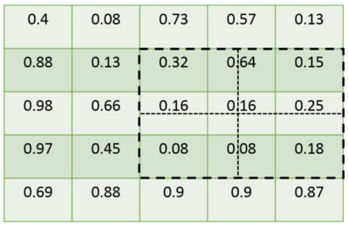
0.4	0.08	0.73	0.57	0.13
0.88	0.13	0.32	0.64	0.15
0.98	0.66	0.16	0.16	0.25
0.97	0.45	0.08	0.08	0.18
0.69	0.88	0.9	0.9	0.87

(a) Feature map with RoI plotted on it



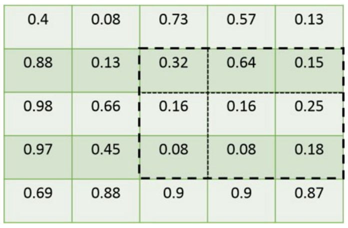
0.4	0.08	0.73	0.57	0.13
0.88	0.13	0.32	0.64	0.15
0.98	0.66	0.16	0.16	0.25
0.97	0.45	0.08	0.08	0.18
0.69	0.88	0.9	0.9	0.87

(b) RoI is scaled to match the feature map



0.4	0.08	0.73	0.57	0.13
0.88	0.13	0.32	0.64	0.15
0.98	0.66	0.16	0.16	0.25
0.97	0.45	0.08	0.08	0.18
0.69	0.88	0.9	0.9	0.87

(c) RoI is divided into bins



0.4	0.08	0.73	0.57	0.13
0.88	0.13	0.32	0.64	0.15
0.98	0.66	0.16	0.16	0.25
0.97	0.45	0.08	0.08	0.18
0.69	0.88	0.9	0.9	0.87

(d) Bins to which max-pooling is applied

Figure 3.9: RoI Pooling operation

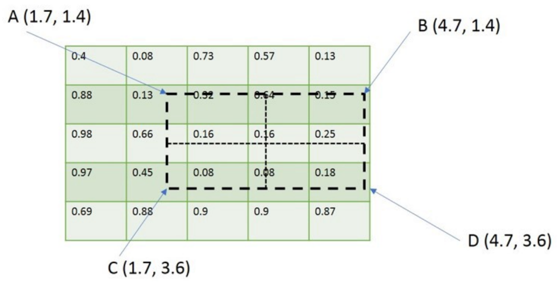
The RoI Pooling operation works well in case of object detection, but do not perform well in cases of instance segmentation due to the existence of many quantization steps which affect the accuracy of mask generation, since pixel to pixel correspondence is fundamental. In order to overcome the misalignment, the RoI pooling layer was replaced by the RoI Align layer.

In Figure 3.10, using the same initial RoI, bi-linear interpolation is used to compute the exact values of the input features at 4 regularly sampled points in each RoI bin 3.10b. According to the authors of Mask R-CNN, the position or number of the sample points is irrelevant as long as no quantization is performed. By using the Figure 3.10b equation, the coordinates of the 4 points are (S1, S2, S3, S4). After that, considering each point it is necessary to identify the 4 nearest points on the feature map which match to the resolution of feature map and compute bi-linear interpolation, in Figure 3.10c, S1 was selected. The next step is perform average pooling for all the sample points belonging in each bin, and the final result is in 3.10d.

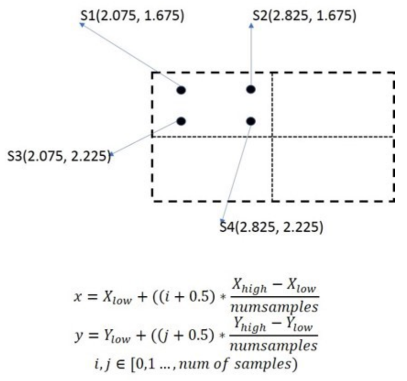
The new RoI Align leads to improved results because it preserves spatial pixel to pixel alignment for each region of interest and since there is no quantization, there is no loss of information.

3.3.6 Network head

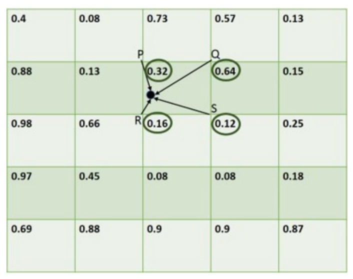
When all RoIs have the same dimensions, they are used as input to the final module known as the head of the network. This head performs the tasks of bounding box recognition, both regression and classification, and masks prediction. As already said in the beginning of 3.3, Mask R-CNN extended the Faster R-CNN head with an additional branch responsible for predict the masks for every region of interest. To predict the masks, each RoI is fed to a Fully Connected Network (FCN) [SLD16] and this way the object spatial layout is preserved. This module works in the same way that the previous one (RPN), but the difference here is that the anchors are not used to locate regions of interest, since at this point, they are already known.



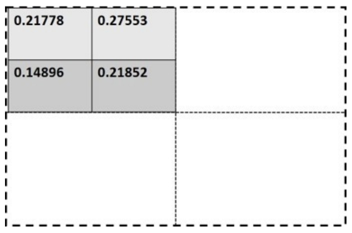
(a) Same RoI plotted in 3.9a


S1(2.075, 1.675)
S2(2.825, 1.675)
S3(2.075, 2.225)
S4(2.825, 2.225)
x = X_low + ((i + 0.5) * (X_high − X_low) / numsamples
y = Y_low + ((j + 0.5) * (Y_high − Y_low) / numsamples
i, j ∈ [0,1 ..., num of samples)

(b) 4 sampled points that belong to a bin



(c) Bi-linear interpolation applied to S1



(d) Result of applying average pooling to all sample points in each bin

Figure 3.10: RoI Align operation

3.3.7 Loss function

The loss function used to train Mask R-CNN is a multi-task loss that combines the tasks of classification, localization, and mask prediction for each sampled RoI. The equation 3.1 defines the loss function used in Mask R-CNN.

$$L = L_{cls} + L_{box} + L_{mask} \tag{3.1}$$

The classification loss L_{cls} and bounding-box loss L_{box} are very similar to the ones that were already used in Fast R-CNN [Gir15]. In its turn, each RoI of the mask branch is composed of K x m x m dimensional output, which encodes K binary masks with the resolution of m x m for each K classes. The mask loss L_{mask} is the average binary cross entropy loss and only includes the k^{th} mask if the region is associated with the ground-truth class k.

Since the model tries to learn a mask for each class, there is no competition among classes when the masks are generated. This approach is different from the ones that were being used since the mask prediction is decoupled from the class prediction.

3.4 RetinaNet

In February 2018, Facebook Artificial Intelligence Research (FAIR) released a paper called Focal Loss for Dense Object Detection [LGG+18] which is the basis of the RetinaNet network.

RetinaNet integrates a group of networks whose purpose is to perform the **object detection** task. This computer vision technique allows to find multiple objects, classify them, and locate them in the image.

Nowadays, there are two kinds of object detectors: the one is called **two-stage** detector and the other is **one-stage** detector. Models in the R-CNN family such as Mask R-CNN 3.3 belong to the first ones, they are all region-based meaning that the detection happens in two stages. The Region Proposal Network 3.3.3 and the RoI Align operation 3.3.5 constitute the **two-stage detector** configuration, since the RPN phase proposes a large set of regions of interest and the RoI Align operation extracts features from each candidate region to perform classification and bounding-box regression tasks. The other approach, **one-stage detector** skips the region proposal step and runs detection directly over a dense sampling of possible locations. Examples of one-stage detector are used in YOLO [RDGF16] and SSD [LAE$^+$16] object detection networks.

Both kind of detectors have advantages and disadvantages, if on one hand one-stage detectors run inference faster, on the other hand two-stage detectors provide higher accuracy in both object localization and recognition.

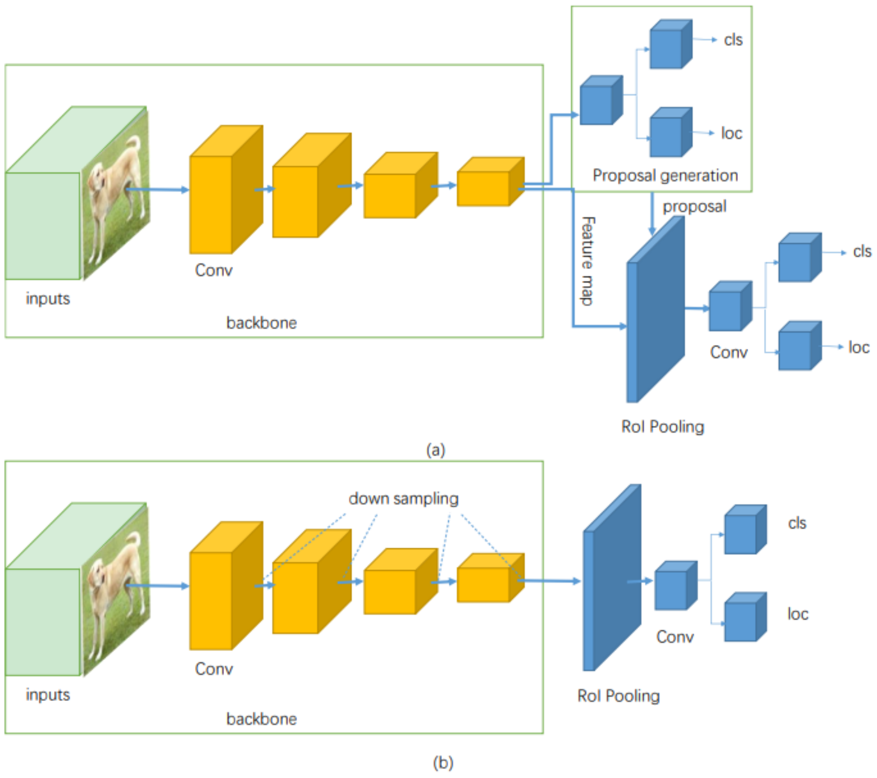


Figure 3.11: Basic architecture of two and one-stage detectors [JZL$^+$19]

In the first part of Figure 3.11 (a), it can be seen the architecture of two-stage detectors and in 3.11 (b), it is illustrated the basic architecture of one-stage detectors, which predicts bounding boxes from input images directly [JZL$^+$19].

An object detection model predicts the bounding boxes coordinates, one for each object it finds in the image, as well as the classification probabilities for each object. Object detection often predicts a huge number of bounding boxes, subsequently, the application of a filter technique is needed in order to discard the bounding boxes whose confidence score is lower than a predefined threshold. The non-maximum suppression technique described in 3.3.4 is used for this purpose.

RetinaNet authors wanted to understand why one stage-detectors are so fast running inference but no so good in the accuracy results when comparing to two-stage detectors. Therefore, they tried to design a one-stage detector that was faster and at least so accurate as the state-of-the-art two-stage detectors [Vid17]. They realized that if there are more bounding boxes covering all locations that may contain objects, one-stage detectors can improve their performance. This way, RetinaNet detector would require approximately 100000 of bounding boxes to densely cover all spatial positions, scales, and aspect ratios [LGG+18]. Per contra, this number is extremely higher than the number of boxes used on the existing one-stage detectors: YOLO detector only requires from 98 boxes to 1000 boxes and SSD detector requires from 8000 to 26000, respectively.

The exponential increase in the required number of bounding boxes brings another major issue: the imbalance of data distribution for foreground and background examples, also called **class imbalance**. Most of the boxes correspond to background examples, and sometimes the background examples can be hard to recognize, but very useful to help recognize the intended objects. However, in general, the background examples are easy (to recognize) and uninformative, thus the large number of easy examples can sabotage the training process and lead to degenerate models.

3.4.1 Focal loss

Lin et. al concluded that the loss function is the central cause for the training process being distracted with easy examples and because of that, they have proposed a new loss function called focal loss [LGG+18]. Instead of reducing the contribution of outliers by down-weighting the loss of hard examples (the ones with large errors), they designed a loss function, the focal loss, that down-weights the contribution of the inliers (also referred as easy examples), this way the novel loss focuses on training on an infrequent set of hard examples and prevents the vast number of easy examples from overwhelming the detector during the training process.

The starting point of focal loss is the **cross entropy (CE)** loss function for binary classification (this loss function can also be extended to multi-class classification by calculating a separate loss for each class label per observation and then sum the result) [7]:

$$CE(p,y) = \begin{cases} -\log(p), & \text{if } y = 1 \\ -\log(1-p), & \text{otherwise.} \end{cases} \tag{3.2}$$

As the authors stated in their paper, in the equation 3.2, $y \in \{\pm\ 1\}$ is the ground-truth class and $p \in [0,1]$ is the probability estimated by the model for the class with label $y = 1$. For notation

[7] https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

convenience, p_t can be defined as 3.3:

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{otherwise,} \end{cases} \tag{3.3}$$

and this way, CE is rewritten as $CE(p,y) = CE(p_t) = \log(p_t)$.

By looking at Figure 3.12, CE is represented by the blue curve and the well-classified examples are on the right side of the figure, since the loss values are lower and the probabilities of the estimated class being the ground truth class is higher. However, due to class imbalance, the sum of the easy examples loss values can overwhelm the rare class as can be seen in the following example. Assuming that there are 100 000 easy examples and just 100 hard examples with loss values of 0.1 and 2.3 respectively, the CE loss will be the sum over the easy and hard examples. In this case, the loss from easy examples is $100000 * 0.1 = 10000$ and the loss coming from hard examples is $100 * 2.3 = 230$, which means that the loss of easy examples is approximately 40 times bigger than the loss of hard examples ($10000/230 = 43$). As reported by Lin et. al: "this is distracting, since hard examples contain more informative signal and the signal is very likely to be overwhelmed by the loss generated by the easy examples" [Vid17].

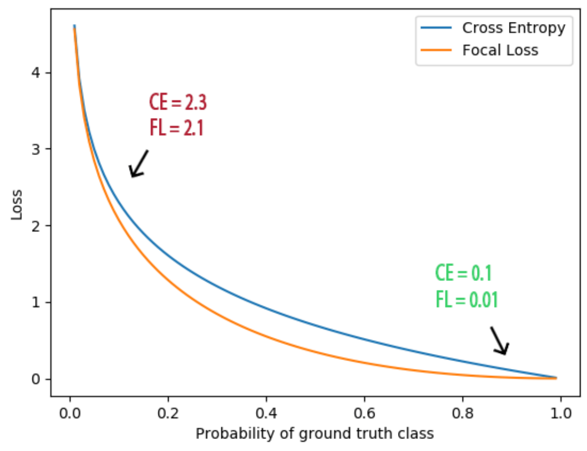


Figure 3.12: Comparison between Focal loss and Cross Entropy loss [KHB+18]

To address this issue, the authors designed a new modulated function that reduces the loss of easy examples and can be defined as in 3.4.

$$FL(p_t) = (1p_t)^{\gamma} log(p_t) \tag{3.4}$$

This function differs from standard cross entropy by adding a factor $(1 - p_t)^{\gamma}$ with tunable **focusing** parameter $\gamma \geq 0$. At Figure 3.13, it is possible to visualize the focal loss for different tested values of $\gamma \in [0,5]$. It is notorious that with higher γ values, more importance, or focus is given to the hard examples. It is important to notice two properties of the focal loss:

1. When an example is wrongly classified and p_t is small, the modulating factor is near 1 and the loss is not affected. As p_t becomes approximately 1, the factor goes to 0 and the loss for well-classified examples is reduced.

2. The focusing parameter γ is responsible for adjusting the rate at which easy examples are down-weighted. If $\gamma = 0$, focal loss is equivalent to cross entropy. $\gamma = 2$ works well in the experiments of the authors.

By using focal loss equation 3.4 instead of cross entropy equation 3.2, the loss of easy examples is reduced (about 10 times in the previous example), whereas the loss of hard examples remains practically unchanged, allowing the training process to focus on hard examples.

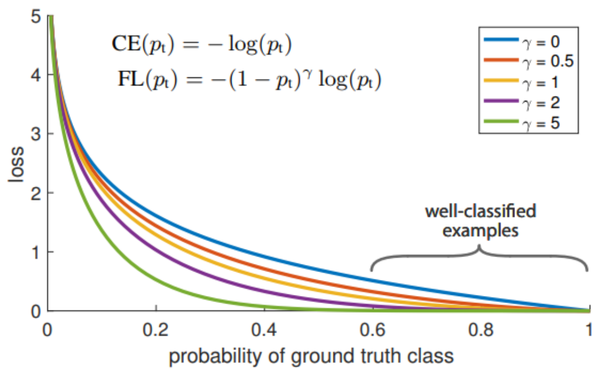


Figure 3.13: Focal Loss plots with different values of γ [LGG+18]

To address the class imbalance, another parameter is used: a weighting factor, $\alpha \in [0, 1]$ and works like foreground-to-background ratio in two-stage detectors, it simply weights the examples of foreground and background in a different way.

3.4.2 RetinaNet detector

RetinaNet is an one-stage detector composed of a backbone network, the network responsible for extracting features from the entire input image, and two specific subnetworks: the task of the first one is to perform object classification while the task of the second one is to perform bounding box regression.

Lin et. al adopted a Feature Pyramid Network backbone upon a feedforward ResNet 3.1 architecture. ResNet performs deep feature extraction and FPN is used for building a rich multi-scale feature pyramid from a single image with a fixed resolution. Besides being semantically strong at all scales, FPN is also fast to compute. There are some minor differences regarding the network architecture described in 3.3.2, but the authors emphasize that the design choices of the architecture are not so crucial as the use of this network is, since they obtained poorly results when they used only a ResNet layer to extract features.

Similar to what happens in Mask R-CNN, the feature maps outputted by the FPN are fed to the RPN 3.3.3 in order to generate the bounding boxes. The anchor boxes are linked to each pyramid level, thus there is no need to use multi-scale anchors at each level. In each pyramid level, the areas of the anchors vary from 32^2 to 512^2 and three aspect ratios {1:2, 1:1, 2:1} are used. To cover the scales more densely, anchors of sizes $\{2^0, 2^{(1/3)}, 2^{(2/3)}\}$ are added at each level, totalizing 9 anchors per level. Across all levels, scale ranges from 32 to 813 pixels.

As Figure 3.14 shows, after the ResNet network (a) and the Feature Pyramid Network (b), two subnetworks are attached to the backbone: the classification subnetwork (c) and the regression subnetwork (d).

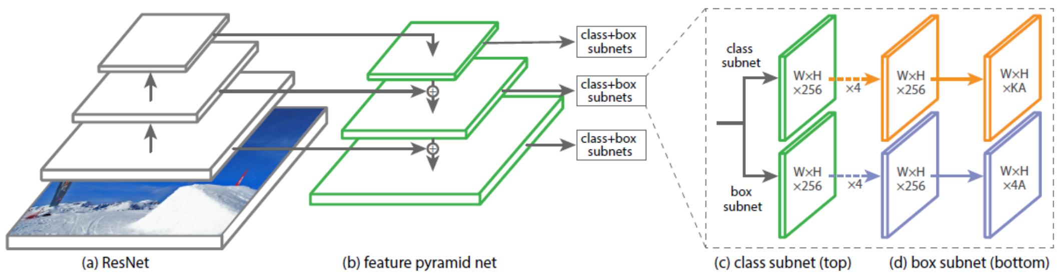


Figure 3.14: RetinaNet network architecture [LGG+18]

The Classification subnet predicts the probability of an object being present in each spatial position for each anchor and for each object class. This fully connected network is attached to each FPN level, sharing parameters, and applies four 3*x*3 convolutional layers, each with C filters and each followed by ReLU activations, followed by a 3*x*3 convolutional layer with KA filters (K classes, A=9 anchors, and C=256 filters). Finally sigmoid activations are attached to the outputs and Focal loss is applied as the loss function.

The Box Regression subnet is another FCN attached to FPN in parallel with the Classification subnet and its purpose is to output the location of an anchor box to a nearby object, if one ground-truth object exists. Despite running in parallel with the Classification subnet and sharing a common structure, no parameters are shared between these two networks. Both architectures are similar, except that Box Regression subnet ends with $4A$ linear outputs (these outputs predict the relative offset between the anchor and the ground-truth object) per spatial location.

3.4.3 Loss function

The loss function applied in the RetinaNet network considers two different tasks: localization (L_{loc}) and classification (L_{cls}) of objects in the image. The equation of the loss (L) can be written as 3.5 and is the sum of the terms related to each one of the tasks:

$$L = \lambda L_{loc} + L_{cls} \tag{3.5}$$

where hyper-parameter λ is the weight that controls the balance between the two task losses.

Regression loss (L_{loc}), as well as the Classification loss, is calculated based on the match of the predicted bounding boxes with the ground-truth bounding boxes. The matching pairs can be denoted as $(A^i, G^i)_{i=1,...N}$, where A is an anchor, G is the ground-truth and N is the total number of matches. As previously mentioned, the regression subnetwork predicts four numbers represented by $P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$, where P^i_x and P^i_y correspond to the offset of the centers of A^i and G^i, while P^i_w and P^i_h specifiy the offset between the width and height of the anchor and the ground-truth bounding box, so the regression loss can be written as 3.6:

$$L_{loc} = \sum_{j \in \{x,y,w,h\}} smooth_{L1}(P^i_j T^i_j) \tag{3.6}$$

where T^i is a regression target associated to each one of the predicted numbers and $smooth_{L1}(x)$ can be defined as 3.7:

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x|0.5 & |x| \geq 1. \end{cases} \tag{3.7}$$

$Smooth_{L1}(x)$ loss function is used since it is less sensitive to outliers [Gir15].

The Classification loss (L_{cls}) adopted by RetinaNet is the **Focal loss**, described in detail in 3.4.1, that is the most innovative part of the network since it **reduces the accuracy gap between one-stage detectors and two-stage detectors**.

3.5 Deeplab

Deeplab is a state-of-the-art semantic segmentation model and its first version was released in 2015 by Google. As the evolution of this network has many changes, this section will mention only the most important ones. More detailed information can be found at [CPK+14, CPK+16, CPSA17, CZP+18].

The idea behind semantic segmentation is to recognize what is in an image at the pixel level, in other words, the goal is to label each pixel with a corresponding class of what is being represented. Unlike instance segmentation, where each instance of an object is segmented separately, semantic segmentation only cares about the category of each pixel.

The semantic segmentation task in an image can be seen as a dense prediction task, since from an input image is expected to be generated an output map of the same dimensions as the input.

3.5.1 Atrous convolution

A weakness of most segmentation models is that the sequence of max-pooling and striding operations at consecutive layers leads to significantly minimization of the spatial resolution of the outputted feature maps. As workaround of this issue, the use of deconvolution layers to up-sample the final feature map could be considered, but it would allocate more time and memory. Thus, instead of using deconvolutional layers, atrous convolutions were introduced in [CPK+14] to amplify the field of view of filters without the need of increase the number of parameters or computational effort.

The atrous rate, also called dilation rate, is the amount of space between the considered pixels in the dilated kernel. As can be seen at Figure 3.15, a higher rate leads to more enlarged feature maps. By applying atrous convolutions, it is possible to achieve large scale features without loosing spatial information.

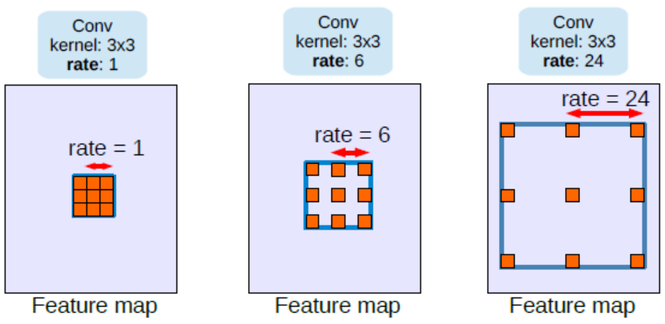


Figure 3.15: Atrous Convolution with different rates

3.5.2 Conditional random fields

Deep convolutional networks have proven its success in classification tasks, but due to its invariance and large receptive fields, the problem of inferring the right position of an object has become more challenging.

With the purpose of capturing finer edges in the image, a fully connected Conditional Random Fields (CRF) was applied. The benefit of using them is because CRF incorporates smoothness terms that maximize label agreement between similar pixels, and can integrate more elaborate terms that model contextual relationships between object classes. Figure 3.16 shows the clearly improvement of a segmentation map after a few iterations of CRF have been applied.

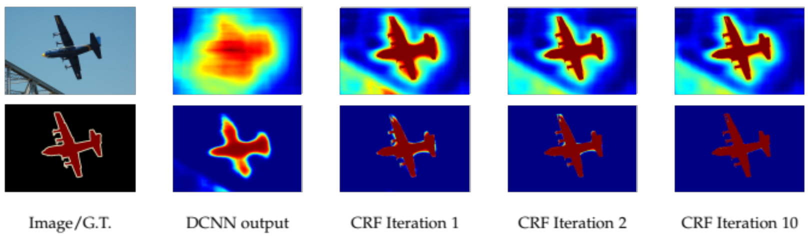


Figure 3.16: Score map (input before softmax function) and belief map (output of softmax function) for Aeroplane. [CPK$^+$14]

Since CRF is a post-processing process and is used in DeepLabv1 and DeepLabv2, it makes this networks no longer be an end-to-end learning framework. Therefore, its use has been discontinued in subsequent versions of Deeplab.

3.5.3 Atrous spatial pyramid pooling (ASPP)

The next problem to solve during the evolution of Deeplab is the existence of objects on multiple scales. An intuitive way to address this issue, represent an object in multiple scales, would be provide re-scaled versions of the same image to the deep convolutional network and then merge the feature or score maps. Chen et. al, saw significantly improvements by using this approach, however the cost of computing feature responses at all layers for multiple scales of the input is extremely high [CPK+16].

As alternative method, Chen et. al proposed the Atrous Spatial Pyramid Pooling (ASPP) that was triggered by the gain of the R-CNN spatial pyramid pooling method of [HZRS14], which demonstrated that "regions of an arbitrary scale can be accurately and efficiently classified by resampling convolutional features extracted at a single scale" [CPK+16].

With ASSP method, multiple atrous convolution layers with different sampling rates are applied to the input feature map, in parallel. The features extracted in each branch for each sampling rate are further processed and merged to generate the final result. At Figure 3.17 is illustrated the ASSP method.

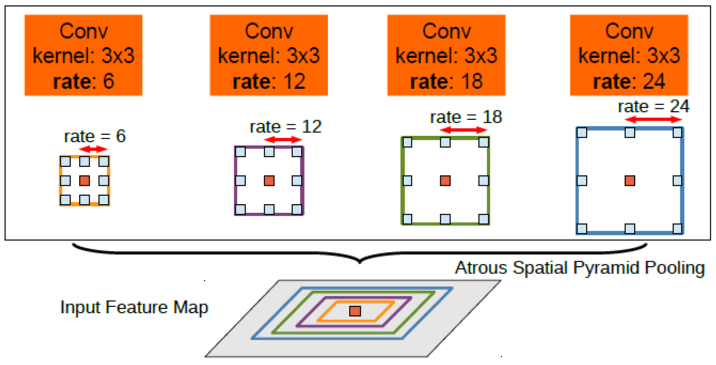


Figure 3.17: Atrous Spatial Pyramid Pooling (ASPP) [CPK+16]

To classify the center pixel that is represented with orange color, ASPP exploits multi-scale features by using multiple parallel filters with different rates. The different colors represent the effective Fields-Of-View.

3.5.4 Encoder-decoder with atrous separable convolution

Deeplabv1 and Deeplabv2 were able to capture contextual information at multiple scales by using atrous convolution at multiple rates. Yet, another challenge remains unresolved, it is needed to capture sharper object boundaries by gradually recovering the spatial information. This issue was addressed by adopting a novel encoder-decoder model with atrous separable convolution which is able to obtain sharp object boundaries. As a rule, the encoder-decoder networks are composed of:

1. An encoder module that gradually reduces the feature maps from the input image while capturing more semantic information.

2. A decoder module that gradually recovers the spatial information, until generates the final output.

Apart from applying the encoder-decoder structure, Deeplabv3 uses a depthwise separable convolution to increase the computational efficiency. The depthwise separable convolution is so named because it deals not only with the spatial dimension, it also deals with the depth dimension, that is the number of channels. A depthwise separable convolution splits a kernel into two separate kernels and then performs two convolutions, the depthwise convolution and the pointwise convolution. More concretely, the depthwise convolution performs a spatial convolution for each of the input channels, while the pointwise convolution has the job of combining the output from the depthwise convolution. The "atrous separable convolution" introduced in [CZP+18] is the result of applying atrous convolution to the depthwise convolution as can be seen at Figure 3.18.

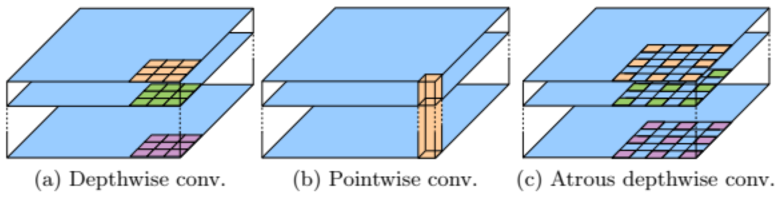


Figure 3.18: $3x3$ Atrous separable convolution - results from applying atrous convolution to depthwise convolution [CZP+18]

3.5.5 Deeplabv3+

The network used in this work is called Deeplab v3+ and it extends Deeplab v3 by adding a decoder module to obtain refined segmentation results along the object boundaries.

Deeplabv3+ encoder uses Aligned Xception as backbone to extract features, instead of ResNet-101 which is used in the previous versions. With this modification, all max-pooling operations were replaced by depthwise separable convolutions with striding, which made possible the application of atrous separable convolution to extract feature maps at an arbitrary resolution [CZP+18].

Deeplabv3+ decoder is the most valuable change when the new network was designed. The encoder features from the previous version of the network are based on an output stride of 16. But up-sampling the features by a factor of 16 using bi-linear operations could be considered a naive decoder module. So, instead of doing it, the approach proposed in Deeplab v3+ firstly bilinearly up-sample the encoded features by a factor of 4 and then concatenate them with the corresponding low-level features from the backbone module that have spatial dimensions of the same size. Before the concatenation, a $1x1$ convolution is applied on the low-level features with the purpose of reducing the number of channels. After the concatenation, some $3x3$ convolutions are applied and the features are up-sampled by a factor of 4 in order to get an output of the same size of the input image. The structure of the encoder-decoder used in Deeplabv3+ architecture can be seen in Figure 3.19.

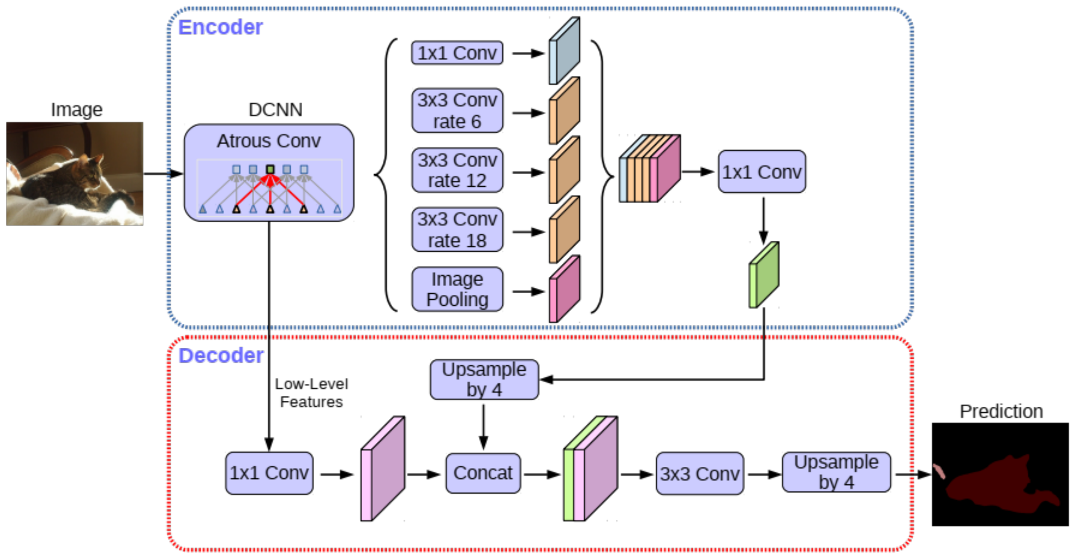


Figure 3.19: Encoder-Decoder structure of Deeplabv3+ architecture [CZP+18]

3.5.6 Loss function

The loss function that is used in the Deeplabv3+ implementation of this work is called Sparse Categorical Cross Entropy loss. The intuition behind this loss is the same that was already explained in 3.4.1 about the cross entropy loss. The Sparse Categorical Cross Entropy loss only differs from Cross Entropy loss because the former does not need to one-hot encode the data, while the second requires the data in a categorical format.

3.6 Single image depth estimation

It is known that CNNs can work well with lower resolution than the resolutions of the image inputs because of their ability to recover spatial information after performing up-sampling operations. In the case of single image depth estimation, the features learned at different layers of the CNNs should reflect different depth cues, since they represent information at different scales. Therefore, the details of the specific object shapes should be confined to the lower layer features while the global depth information should be processed by the higher layer features. In order to achieve more accurate depth maps, the use of both layer features must be complementary.

Considering this, Junjie Hu et. al [HOZO18] proposed a feature fusion module that through the use of skip connections is able to up-scale low level features and then combines the different scale features by using a convolutional layer. To deal with high level features, a decoder module is used to untangle them. After this step, both high and low level features are fused together to provide an image depth estimation.

3.6.1 Architecture

Figure 3.20 illustrates the architecture that was designed by Junjie Hu et. al. As can be seen, it is composed by four different modules: an **E**ncoder, a **D**ecoder, a **M**ulti-scale **F**eature **F**usion module, and a **R**efinement.

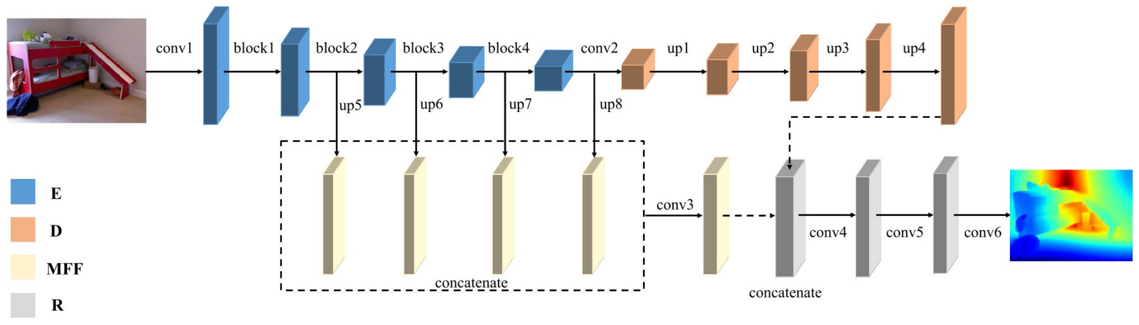


Figure 3.20: Single Image Depth Estimation network architecture [HOZO18]

Given an input image, the encoder module is responsible for extracting features at different scales ($\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, and $\frac{1}{32}$) while the decoder employs four up-sampling operations to up-scale the final features obtained by the encoder gradually, at the same time that the number of channels is reduced. In this work, a ResNet architecture was used as backbone encoder. The intermediate module, MFF is the one that integrates the features with different scales by up-sampling operations and the outputs of this module are fed to the refinement module, composed if three convolutional layers to provide the final estimations.

3.6.2 Loss function

Previous studies in the field of single image depth estimation used as loss, a function that mirrors the difference between the depth estimation (d_i) and the the depth of a ground-truth image (g_i) 3.8.

$$l_1 = \frac{1}{n}\sum_{i=1}^{n} e_i, \qquad \text{where } e_i = |d_i - g_i|. \tag{3.8}$$

However there are two issues when employing this loss function. On the one hand, a unit depth difference, for example $1cm$, weights the loss in the same way for distant points and for nearby points in a scene. The contribution to the loss should be greater on nearby points and smaller on points with far away location. In RetinaNet, the focal loss address a similar issue when it uses a factor term to focus more on hard examples instead of easy examples. In Junjie Hu et. al implementation, a new version of the loss function was proposed and it can be defined by the equation 3.9:

$$l_{depth} = \frac{1}{n}\sum_{i=1}^{n} F(e_i), \qquad \text{where } F(x) = \ln(x+\alpha), \tag{3.9}$$

where $\alpha > 0$ is a tunable parameter.

On the other hand, it is more difficult to find a solution to address the second issue: for a step edge structure of depth, the 3.9 is sensitive to shifts in the depth direction, but it is insensitive to shifts in the conventional directions, x and y. Besides that, the loss function is also insensitive to distorted or blurry edges, and because of that, the implementations that use this loss tend to produce depth maps with more distorted or blurry edges.

For this reason, the penalty of the errors around the edges should be greater as suggested by 3.10:

$$l_{grad} = \frac{1}{n}\sum_{i=1}^{n}(F(\bigtriangledown_x(e_i)) + F(\bigtriangledown_y(e_i))), \tag{3.10}$$

where $\bigtriangledown_x(e_i)$ is the spatial derivative of e_i computed at the i^{th} pixel regarding to x, and so on. This loss function is now sensitive to the shifts of the edges in both x and y directions.

Depth maps of natural scenes can also be modeled by smooth surfaces and not only the step edges. With this in mind, another loss function was defined to measure the accuracy of the normal to the surface of a predicted depth map with respect to its ground-truth. The surface normal of a predicted depth map can be denoted by $n_i^g \equiv [-\bigtriangledown_x(d_i), -\bigtriangledown_y(d_i), 1]^\top$ and the surface normal of the ground-truth can be defined as $n_i^g \equiv [-\bigtriangledown_x(g_i), -\bigtriangledown_y(g_i), 1]^\top$, the normal loss can be defined as 3.11:

$$l_{normal} = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \frac{\langle n_i^d, n_i^g\rangle}{\sqrt{\langle n_i^d, n_i^d\rangle}\sqrt{\langle n_i^g, n_i^g\rangle}}\right), \tag{3.11}$$

where $\langle .,. \rangle$ is denoting the inner product of vectors. Finally, since each of these three losses complement each other, the final loss is a weighted sum of all losses 3.12:

$$L = l_{depth} + \lambda l_{grad} + \mu l_{normal}, \qquad \text{where } \lambda \text{ and } \mu \text{ are weighting coefficients.} \tag{3.12}$$

The combination of the three losses made possible to achieve a model that is able to estimate depth maps with finer resolution reconstruction and thus, can beat the accuracy of the state-of-the-art models for single depth estimation.

3.7 Unsupervised segmentation

The work of Asako Kanezaki [Kan18] explores the use of CNN for the task for unsupervised image segmentation. As the supervised learning networks that perform the task of image segmentation like Deeplab, the CNN proposed by Kanezaki assigns labels to pixels that denote the cluster to which the pixel belongs. The difference is that in the scenario of unsupervised learning, training images and ground truth labels of pixels are not given beforehand. Therefore, once when a target image is input, the pixel labels together with the feature representations are jointly optimized, while their parameters are updated by gradient descent.

With the proposed approach, the label prediction and network parameter learning are iterated to meet the following criteria:

1. pixels of similar features are desired to be assigned the same label;
2. spatially continuous pixels are desired to be assigned the same label,
3. and, the number of unique labels is desired to be large.

In order to address the first criterion, which consists in assigning the same label to the pixels of similar features, the author applied a linear classifier that classifies the features of each pixel into q classes.

The concept of image pixel clustering is to group similar pixels into clusters, however, in the segmentation task is preferable that the clusters of image pixels are spatially continuous. To undertake the second constraint on spatial continuity, the implementation of the author favors cluster labels that are the same as those of the neighboring pixels by extracting K fine superpixels, then the pixels that belong to the superpixel are forced to have the same cluster label.

When dealing with unsupervised image segmentation, there is no clue as to how many segments should be generated in the image. Thus, the number of unique labels should be adaptive to the image content. To restrict the number of unique cluster labels to be large, the author inserts the $intra-axis$ normalization process before assigning cluster labels via argmax classification.

Once the input image is fed to the network, two sub-problems are being solved simultaneously: the prediction of cluster labels with fixed network parameters and the training of network parameters with the (fixed) predicted cluster labels. The former corresponds to the forward process of a network followed by the superpixel refinement, while the second is achieved by calculating the softmax loss and backpropagating the error signals by stochastic gradient descendent with momentum to update the parameters. The parameters are initialized with Xavier initialization [GB10], which samples values from the uniform distribution normalized according to the output and layer size.

The proposed method was evaluated by using 200 test images from the Berkeley Segmentation Dataset and Benchmark (BSDS500) [AMFM11, MFTM01] and it can be concluded that many meaningful segments with various colors and textures (such as a tiger and a giraffe) are successfully detected as shown in Figure 3.21.

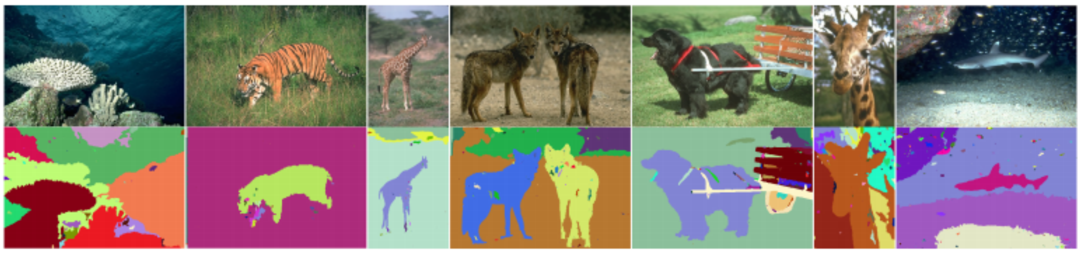


Figure 3.21: Detections made by the unsupervised learning segmentation approach [Kan18]

Chapter 4

Proposed Solution

Despite all the attempts to develop a similar system, there is still no fully reliable system to support diabetic patients in automatically counting carbohydrates in food. The present Chapter explores the solution found in order to address this problem in Section 4.1. The dataset used in most experiments is described in Section 4.1.1.2 and the necessary conversion from volume to weight to get the nutritional values of the food is covered in Section 4.1.2.

In Chapter 2, it was possible to see the efforts made by many researchers to address the problem of food recognition and volume estimation on food images. However, due to the nature of food images, their recognition involves big challenges mainly because food mostly consists of deformable objects and so it becomes difficult to define their structure.

Furthermore, some food types can have a low inter-class (different foods have similar appearance) and high intra-class (similar foods have many differences in their appearance) variance, making the task of food recognition even more challenging.

Besides that, recognizing and identifying food from images, is even more difficult because of the image acquisition conditions. As already mentioned, variables such as background, lighting conditions, angle or color distribution affect the quality of image processing.

Although there are several studies addressing the problem of image classification, there are still few approaches for calculating the volume of food present in photographs. And there are still fewer studies that deal with both problems simultaneously.

4.1 Solution

Considering the literature review described in Chapters 2 and 3, the preference for the deep learning models that have been observed in recent years is remarkable.

As can be seen in 2.1, the traditional approaches require the definition of complex features that may take a long time to be manually set, and if automated feature engineering techniques are applied, the time invested in the feature extraction process is significantly reduced.

Thus, since there are already studies (for example, the one done by Ciocca et al. [CNS17a]) that prove that the features extracted by Convolution Neural Networks lead to better results in the classification and segmentation of food images than the traditional methods described in 2.1, we decided to use CNNs to perform the feature extraction and subsequent image classification and segmentation tasks.

As can be seen in Figure 4.1 the proposed solution has two big steps: the first one is composed of two tasks, segmentation and classification of the foods that belong to an image, while in the second step the volume is estimated.

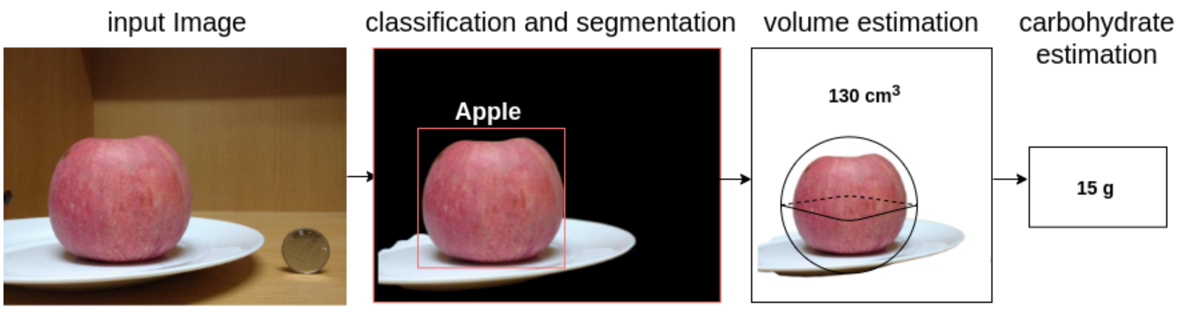


Figure 4.1: Workflow of the proposed solution

It was decided to use the state-of-the-art methods already described in Chapter 3 to perform the first step in the proposed solution. Therefore, to segment and classify the foods present in an image, Mask R-CNN, RetinaNet, and Deeplabv3+ are used. The choice of these networks was not in vain, that is, the difficulty of finding a proper dataset (issue that is approached in 4.1.1) was considered and so, the chosen networks perform distinct tasks. Mask R-CNN was designed to deal with instance segmentation, RetinaNet focuses in the task of object detection, and finally, Deeplabv3+ attacks the problem of semantic segmentation. Our goal, estimating the volume of the food items, would be achieved more accurately if the used dataset was able to perform the instance segmentation task. However it is only designed for object detection since the labels provided only include the coordinates of the bounding boxes that delimit the food objects.

To estimate the volume of food present in the images, two different implementations were tested. In the first one, the 3D models of the objects were generated from 2D images, however, the results obtained from a test image were not promising as described in section 5.5.1.

Therefore, the task of estimating the volume of the food was achieved by generating the depth maps of the original images as described in 5.5.2. With the depth maps of the images that contain a calibration object, whose dimensions are known, it is possible to extract the scale and know what is the real distance that is represented by each pixel in the image. The object used to extract the scale is the plate and knowing its real depth, it is also possible to infer the depth that is represented by each color in the generated depth maps. This way, it is possible to estimate the volume of the foods that are present in the images. The conversion of the volume values to the nutritional values of each food is explained below in section 4.1.2.

Each one of the networks tested in this project was trained using one of the three NVIDIA Tesla V100 PCIe GPU [1] modules, with memory size of 16GB in two of them and 32GB in the other one.

4.1.1 Datasets

Datasets used to address the object recognition problem can be split into three different groups according to the task to they are used for [LMB+14]:

- Object classification - the dataset used for this task must contain a label indicating what objects belong to the image.
- Object detection - this task requires a classification label as well as the location of the object. This location is often provided by the bounding boxes coordinates of the rectangle that surrounds the object.
- Object segmentation - this task requires a dataset that has images labelled at the pixel level. This is, along with the original images, mask images are required. If mask images are not available, at least a document in any format that contains the coordinates of the approaching polygon to each object.

In Appendix A, an image of the table taken from [SAM19] is found, where some of the most commonly used datasets for food recognition are listed.

The purpose of this work goes beyond the classification task, it embraces also the task of image segmentation. This way, it is required a dataset that contains images and their corresponding labels in terms of segmentation. In addition, to accomplish the last step of the proposed solution, estimate the volume of the food, extra information such volume information of the food is also required. Of all datasets listed in Appendix A, only the dataset called UNIMIB2016 [CNS17a] contains a document with polygonal boundaries that can be used to mask the food present in the image. However this dataset is annotated in Italian and contains much food that is not regularly eaten in our region. Besides that, it could not be used in the task of volume estimation, since there is no information regarding the volume of the food.

After an extensive search, a dataset that contemplates the requirements of volume annotations was found. The ECUSTFD dataset is described in 4.1.1.1 and it was exploited in a work with a purpose very close to the purpose of this one. Liang et. al [LL17a, LL17b] created the ECUSTFD dataset to build an application that would be able to deduce the calories information of the food present in an image. In fact, the workflow of their work is very related to ours, the only difference is that after the volume has been estimated we intend to infer the amount of carbohydrates and they calculated the amount of calories in the foods in the image.

Note that, at the beginning of the project we still did not know how we would proceed to estimate the food volume and that is the main reason why we restricted the search for a dataset

[1] GPU stands for Graphics Processing Unit

that contained volume annotations as well as a common object belonging to the images that could be used for calibration. Since our work aims to find a solution to a problem closely related to the one that is being addressed by Liang et. al [LL17a, LL17b], their project is considered as a baseline to the present project.

The dataset that was used in most experiments is an extension of the ECUSTFD dataset 4.1.1.1 and it is described in detail in 4.1.1.2.

4.1.1.1 ECUSTFD

ECUSTFD dataset was collected by Liang et. al [LL17a] and it contains 2978 images distributed in 20 classes: *apple, banana, bread, bun, coin, doughnut, egg, fired dough twist, grape, lemon, litchi, mango, mooncake, orange, peach, pear, plum, qiwi, sachima, and tomato*. For a single food portion, they took several groups of images by using smart phones and each group of images contains a top view and a side view of the food. For each image, there is only one coin as calibration object and no more than two items of foods in it. If there are two items of food in the same image, the type of one food is different from another. Figure 4.2 has a sample of the images present in the ECUSTFD dataset.

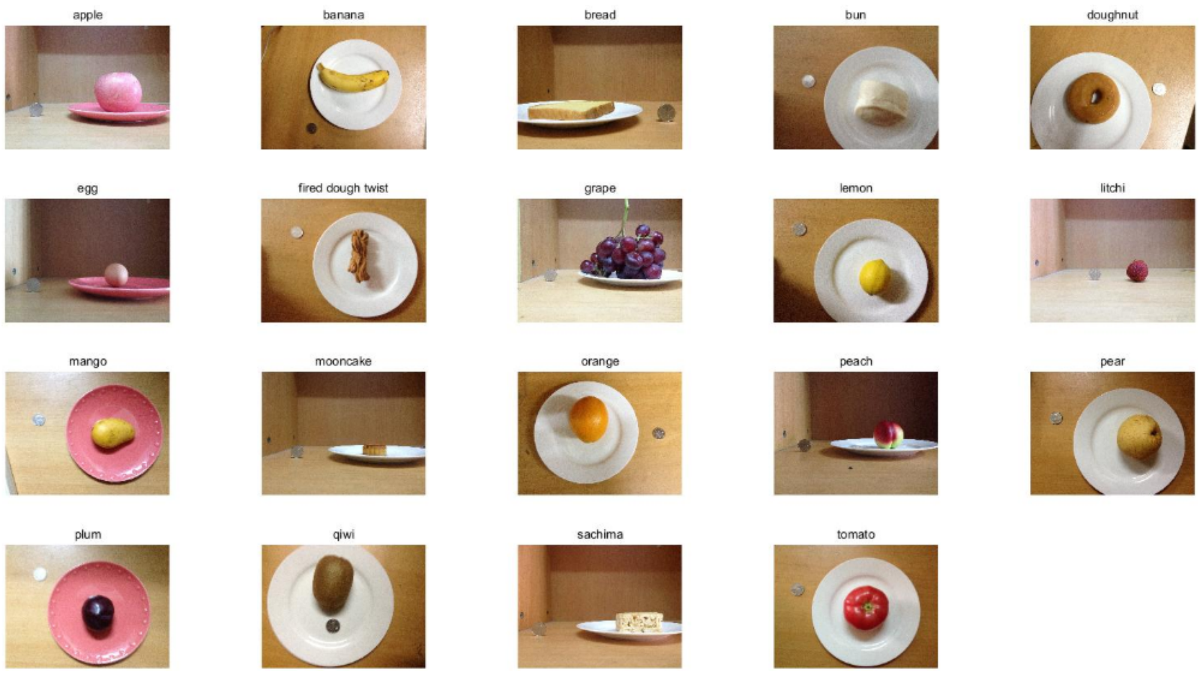


Figure 4.2: Examples of images of the **ECUSTFD** dataset [2]

The dataset contains annotations in *.xml* files for each image with information regarding the width and height of the image, as well as the bounding box positions of both the food and the coin used for calibration. The dimensions of both plates used in the captured images are known: the white plate has 20.7 *cm* of diameter and its height, or depth is 2.0 *cm*, while the red plate has 18.7 *cm* of diameter and its height, or depth is 2.0 *cm*.

[2] https://raw.githubusercontent.com/Liang-yc/images4readme/master/food_sample.jpg

What makes this dataset differentiating from the rest is the existence of a *csv* file with the volume in mm^3 and the weight in g of the food belonging to the images.

4.1.1.2 Used dataset

Initially, the dataset to be used would be the ECUSTFD dataset 4.1.1.1, since it contained the indispensable annotations to estimate the food volume.

The first experiments, that are explained in 5.2, showed that only the ECUSTFD dataset 4.1.1.1 was not sufficient. Therefore, it was decided that the number of images in the dataset should be increased and the food background should be different in order to train more generalist models that are able to get better results. Figure 4.3 shows some of the images collected from the OID dataset and it is possible to visualize the diversity of backgrounds introduced with the new images.

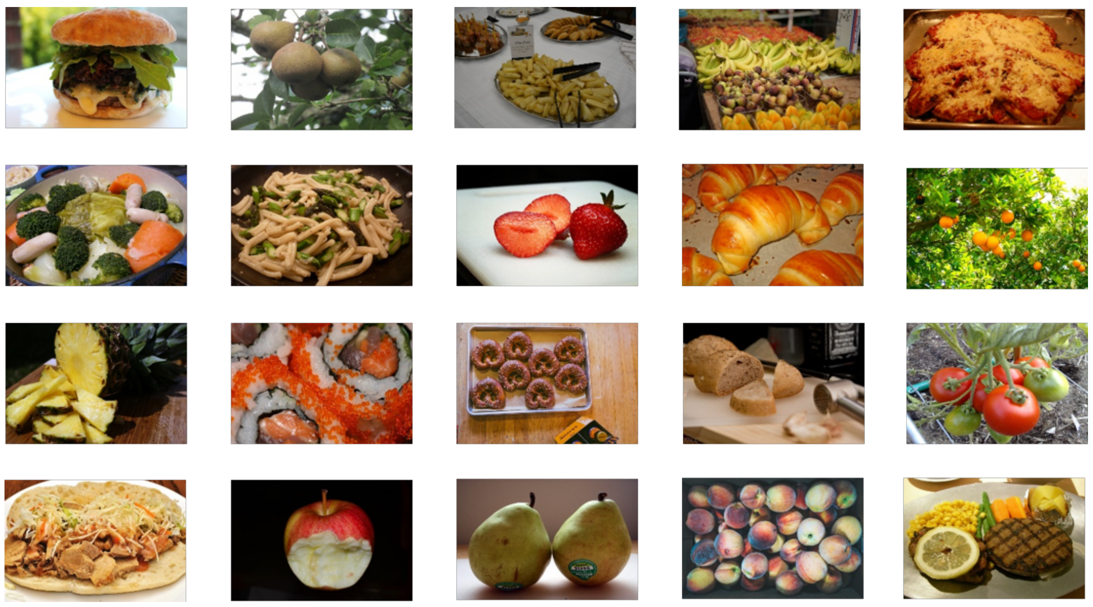


Figure 4.3: Examples of images of the **OID** dataset

Beyond using the complete ECUSTFD dataset, some images were collected with the help of OIDv4_ToolKit [Vit18]. This toolkit makes possible to download images from Open Images Dataset (OID) [KDA+16] belonging to specific classes and the corresponding *.txt* files with the bounding boxes of the food objects in each image. OID is a public image dataset with about 9 million images that have objects belonging to approximately 600 classes. The images extracted from the OID dataset are all food related (except the coin class) and belong to the following classes: *apple, bagel, banana, bread, broccoli, burrito, cabbage, cake, carrot, cheese, coconut, coffee, coin, cookie, croissant, cucumber, doughnut, egg, french fries, grape, grapefruit, hamburger, hot dog, ice cream, lemon, mango, milk, muffin, mushroom, orange, pancake, pasta, peach, pear, pineapple, pizza, pomegranate, popcorn, potato, pretzel, radish, strawberry, sushi, taco, tart, tea, tomato, waffle, watermelon, and winter melon*.

From this dataset, approximately the same number of images were extracted per class, however these images were added to the ECUSTFD dataset. As a result, the 20 classes that were represented in the initial dataset have more images than the remaining ones. While the initial dataset had 20 classes, the new one has 57 classes.

Images extracted from OID [Vit18] have only one category per image. However, some ECUSTFD 4.1.1.1 images have objects of various categories in the same image. Thus, the sum of the values for each column of Table 4.1 is slightly different of the sum of the images present in each of the datasets.

For clarity, the dataset used has 6260 images in total, divided into three subsets: the training, the validation and the test datasets with 4047, 1041, 1172 images, respectively.

The complete dataset was split so that the ECUSTFD 4.1.1.1 images remained the same for the subsets that the authors defined in [LL17a]. The remaining images collected from the OID [Vit18] were randomly separated. Therefore the percentage of images in the training subset is 60% and the percentage of images in the validation and test subset is 20% in each.

Table 4.1 shows for each of the 57 categories the total number of images present in the dataset, the number of images in each subset (training, validation, and test) and their percentage relative to the complete dataset.

The images that make up this dataset have different size, however, the maximum dimension for height and width is fixed on 1024 pixels.

4.1.2 Converting volume into nutritional information

After estimating the food volume, a way to convert it into mass values is required in order to get the nutritional information of the food and retrieve the amount of carbohydrates that it has.

The density formula relates the concepts of volume and mass, and the density equation can be defined as 4.1, where ρ represent the density, m is the mass and v is the volume:

$$\rho = \frac{m}{v} \tag{4.1}$$

which means that we can calculate the mass value from a certain food present in an image, if we know the density for that food.

To obtain the density ρ values, a table called *AUSNUT 2011-13 AHS Food Measures File* from FSANZ was consulted. FSANZ stands for Food Standards Australia New Zealand and is "an independent statutory agency" that works for the Australian Government with the purpose of developing food standards for Australia and New Zealand. On their website [Zea19], it is possible to find several food related information.

The next step after having the information about the mass of food present in the image is to estimate the amount of carbohydrates that the food has. In order to do it, it was needed to examine a nutritional table, which contains the nutritional facts of the food. We decided to look into the same nutritional table that Liang et. al used when they were developing their work. Therefore, we considered the tables published in [oC13] by the Canadian Government since they detail the

Table 4.1: Distribution of images across categories in the used dataset

Class	Total nº images	Train nº images	Validation nº images	Test nº images	% Train	% Validation	% Test
Apple	412	215	94	103	52.18	22.82	25.00
Bagel	75	46	15	14	61.33	20.00	18.67
Banana	285	151	68	66	52.98	23.86	23.16
Bread	144	85	33	26	59.03	22.92	18.06
Bun	90	58	16	16	64.44	17.78	17.78
Broccoli	71	48	13	10	67.61	18.31	14.08
Burrito	82	47	14	21	57.32	17.07	25.61
Cabbage	67	48	10	9	71.64	14.93	13.43
Cake	89	49	15	25	55.06	16.85	28.09
Carrot	108	49	16	43	45.37	14.81	39.81
Cheese	85	49	17	19	57.65	20.00	22.35
Coconut	107	49	21	37	45.79	19.63	34.58
Coffee	52	33	8	11	63.46	15.38	21.15
Coin	3057	1782	636	639	58.29	20.80	20.90
Cookie	97	47	21	29	48.45	21.65	29.90
Croissant	145	48	19	78	33.10	13.10	53.79
Cucumber	193	48	29	116	24.87	15.03	60.10
Doughnut	289	170	60	59	58.82	20.76	20.42
Egg	192	109	39	44	56.77	20.31	22.92
Fired dough twist	124	66	29	29	53.23	23.39	23.39
French fries	68	48	10	10	70.59	14.71	14.71
Grape	167	83	54	30	49.70	32.34	17.96
Grapefruit	97	49	20	28	50.52	20.62	28.87
Hamburger	44	14	9	21	31.82	20.45	47.73
Hot dog	107	47	12	48	43.93	11.21	44.86
Ice cream	97	49	26	22	50.52	26.80	22.68
Lemon	275	198	39	38	72.00	14.18	13.82
Litchi	78	48	15	15	61.54	19.23	19.23
Mango	334	176	75	83	52.69	22.46	24.85
Milk	71	49	10	12	69.01	14.08	16.90
Mooncake	134	68	33	33	50.75	24.63	24.63
Muffin	81	45	15	21	55.56	18.52	25.93
Mushroom	106	49	32	25	46.23	30.19	23.58
Orange	318	175	71	72	55.03	22.33	22.64
Pancake	80	46	16	18	57.50	20.00	22.50
Pasta	68	48	10	10	70.59	14.71	14.71
Peach	221	127	55	39	57.47	24.89	17.65
Pear	257	159	47	51	61.87	18.29	19.84
Plum	176	94	41	41	53.41	23.30	23.30
Pineapple	86	48	11	27	55.81	12.79	31.40
Pizza	66	48	9	9	72.73	13.64	13.64
Pomegranate	69	48	11	10	69.57	15.94	14.49
Popcorn	68	49	10	9	72.06	14.71	13.24
Potato	68	48	10	10	70.59	14.71	14.71
Pretzel	75	46	17	12	61.33	22.67	16.00
Qiwi	137	66	36	35	48.18	26.28	25.55
Sachima	150	96	27	27	64.00	18.00	18.00
Radish	127	49	12	66	38.58	9.45	51.97
Strawberry	139	49	51	39	35.25	36.69	28.06
Sushi	83	49	16	18	59.04	19.28	21.69
Taco	75	49	13	13	65.33	17.33	17.33
Tart	64	46	9	9	71.88	14.06	14.06
Tea	68	48	9	11	70.59	13.24	16.18
Tomato	327	200	53	74	61.16	16.21	22.63
Waffle	90	48	13	29	53.33	14.44	32.22
Watermelon	72	49	12	11	68.06	16.67	15.28
Wintermelon	47	36	3	8	76.60	6.38	17.02

nutrition facts of most food classes that we cared about while we were developing the current work. These tables define the amount of each nutrient for a specific mass, so we need to relate the mass values from the table with the determined values from our system and then we obtain the final result for the carbohydrate estimation. The calculations that are performed to get the amount of carbohydrates are the same that are done to estimate the quantities of other nutrients.

Although the goal of this dissertation is to estimate the carbohydrate amount, it is also possible to estimate the quantities of other nutrients like proteins, calories, sugars, and fats with the proposed solution.

Chapter 5

Results and discussion

The results and its discussion are presented separately for each neural network used: Mask R-CNN in Section 5.2, RetinaNet in Section 5.3, and Deeplab in Section 5.4. The volume estimation results are in Section 5.5.2 and the results of the refined segmentation that may be incorporated in the system in the future are in Section 5.6.

5.1 Performance metrics

If the proposed methods in this work are used to develop a mobile application, the diabetic patients who will use it, have to be sure that it is a reliable system.

The first step in the described approach is composed of two tasks that should be assessed: image classification and image segmentation as shown in Figure 4.1. On the one hand, the model can detect different objects, but it may or may not assigning the correct class to them. On the other hand, even if the class is correctly predicted for each object, the model can be predicting wrong detections in terms of spatial location in the image.

Therefore, it is fundamental to use metrics that can evaluate the model's performance in both segmentation and classification tasks.

In order to have a baseline to compare the performance of the proposed system with the performance of the state-of-the-art implementations, the main metric reported in/for the PASCAL Visual Object Classes (VOC) [EVGW$^+$] and COCO challenges [LMB$^+$14] is mean average precision (mAP) and it was calculated to evaluate the achieved models.

This metric is based on the Intersection over Union (IoU) that measures how the boundaries of the predicted bounding boxes overlap in relation to the ground truth bounding boxes. The calculation of IoU is done by dividing the intersection of both bounding boxes by their union [RTG+19]. The equation to compute the IoU between two arbitrary shapes or volumes is 5.1.

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{5.1}$$

Once IoU is determined and compared to a predefined IoU threshold, it is possible to know what detections, the algorithm has considered as true positive (TP), false positive (FP) and false negative (FN). To clarify, a prediction is considered to be true positive if $IoU > threshold$ and false positive if $IoU < threshold$, assuming that in both cases the assignment of the class was correct. The prediction is considered a false negative when $IoU > threshold$, but has the wrong classification. Therefore, it is possible to estimate the most common machine learning metrics such as precision and recall, whose equations are 5.2 and 5.3, respectively. Recall is also called the True Positive Rate because it shows of all the actual positives, how many are true positives predictions. The precision is also known as the Positive Predictive Value because it reflects of all the positive predictions, how many are true positives predictions.

$$Precision = \frac{TP}{TP+FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP+FN} \tag{5.3}$$

The average precision (AP) is computed from the model's precision-recall curve as the area under the monotonically decreasing part of the curve.

Contrary to Pascal VOC challenge [EVGW+] that uses a single fixed IoU threshold value, to determine COCO's [LMB+14] mean average precision (mAP), several AP values are computed by averaging the precision values for all the recall levels found during the evaluation process. Each AP is associated with a distinct IoU threshold in the 0.5–0.95 range (using a step of 0.05), and the mean of all the resulting AP values is computed to attain the mAP score.

Since evaluating the models for ten different thresholds and then calculate the average of the results is time consuming, it was decided to choose only two different thresholds to make these calculations, 0.5 and 0.75. Each threshold is independently evaluated, this is, we do not average the results obtained with the different thresholds, we simply perform the calculation by using one threshold at a time.

5.2 Mask R-CNN

The base implementation of Mask R-CNN [1] is a Python3, Keras and Tensorflow implementation of the network.

Originally, Mask R-CNN was trained with the COCO dataset [LMB+14] and for that reason, the used implementation requires the input data to be in the same format.

Since the data from the collected dataset 4.1.1.2 had different sources, it had also different formats: the annotations that came from the ECUSTFD dataset 4.1.1.1 were written in a *.xml* file, while the annotations from the images gathered from the OID [KDA+16] were written in a *.txt* file. Yet, the annotations in the COCO dataset are stored in a *.json* file and follow different rules, this way it was mandatory to reformulate all the annotations to match the rules imposed by COCO.

COCO defines five annotation types used for different tasks: object detection, keypoint detection, stuff segmentation, panoptic segmentation, and image captioning. We used the annotation type for the object detection type that contemplates the use of bounding boxes and the segmentation mask points. Figure B.1 illustrates the required data format by COCO, and consequently the data format of the annotations of the used dataset 4.1.1.2.

Each one of the Mask R-CNN models was trained for 100 epochs and the results obtained using the original dataset are impressive and surpass the results that Liang et. al [LL17a, LL17b] obtained using Mask R-CNN's predecessor network, Faster R-CNN. As already mentioned in 4.1.1.1, the dataset division into training, validation and testing subsets was made according to the division that Liang et. al used and yet the test subset evaluation **improves from 93% to 96.2%**.

The results of the evaluation on the train and test subsets can be seen in Table 5.1 and in this model, no data augmentation was used during the training process.

Table 5.1: mAP with the ECUSTFD dataset using Mask R-CNN

Subset	*mAP @ 0.50 IoU*
Train	0.974
Test	**0.962**

However, when using the model obtained to make the inference from a food image that was very different from those in the test dataset, differences in the surrounding background, the model was unable to make any detection, which led us to think that it was too specific for the dataset in which it had been trained and it **had not the ability to generalize**.

[1] https://github.com/matterport/Mask_RCNN

To deal with this problem, we expanded the dataset by including images from different sources collected in different environments, we make use of some images from the Open Images Dataset [KDA+16]. From this point, we started to use the dataset described in 4.1.1.2.

With the use of the new dataset, the number of classes to be recognized has increased from 20 to 57. The results of the evaluation on the train, validation, and test subsets are attached to the Table 5.2. By looking into it, it can be seen that the model obtained using the new dataset gives worse results than the previous one.

Table 5.2: mAP with the modified dataset using Mask R-CNN without data augmentation during the training process

Subset	*mAP @ 0.50 IoU*	*mAP @ 0.75 IoU*
Train	0.389	0.372
Validation	0.511	0.485
Test	0.447	0.426

Since the increase in the number of classes to be identified is not directly proportional to the increase in the number of images in the new dataset, we decided to apply data augmentation during the training process to verify if it was possible to improve the obtained results.

Data augmentation was applied to each sample (the input image and the corresponding mask image) by using the following data augmenters:

- Flip: the input image and the mask image are both horizontally and vertically flipped with a probability of 0.5.
- Rotation: the input image and the mask image are both rotated by a random degree, $r \in [-180, 180]$.
- Scaling: the input image and the mask image are both randomly scaled in a factor $f \in [0.5, 1.2]$ in both directions x and y.
- Translation: the input image and the mask image are both translated with a percent value $p \in [-0.2, 0.2]$ relative to height/width both horizontally and vertically.
- Contrast Normalization: the input image and the mask image contrast was normalized by a factor $f \in [0.75, 1.5]$ sampled randomly per image.
- Gaussian Blur: it was added noise to the input image and the mask image.

As result, the obtained results improved with the application of data augmentation but not significantly as Table 5.3 shows.

Table 5.3: mAP with the modified dataset using Mask R-CNN with data augmentation during the training process

Subset	*mAP @ 0.50 IoU*	*mAP @ 0.75 IoU*
Train	0.426	0.409
Validation	0.563	0.527
Test	0.502	0.466

We were trying to segment and classify objects from 57 different classes and whenever an image was given for inference, each model outputted an image with multiple detections for the same object. These detections corresponded to different classes and for the same class only the detection with higher confidence value was outputted. Most of the bounding boxes that surrounded an object did not intersect with the bounding boxes that surrounded a different object. Therefore, to get the final result, the bounding box with higher confidence value was considered, and if there was one bounding box that overlapped the considered bounding box with an intersection superior to 30%, it was excluded. This filtering is done because there are images that have more than one object, and if we only considered the detection with higher confidence value, the remaining objects would have no associated detections. This post processing filtering was applied both on Mask R-CNN and RetinaNet.

Figure 5.1 shows the performance evolution of the models. The first image in the figure is the input image that was provided to the models to make the inference. This picture has two apples, two oranges and four bananas. The output of the first model, the one that was trained using only the ECUSTFD dataset, is the same as the input which means that the first model was not able to make any detection on this image. The second model, trained with the modified dataset without using data augmentation outputted the middle image of the figure. Here, two detections were made, however the predicted classes, tart and coin, do not match with the ground-truth classes. The input image was fed to the model trained with the modified dataset by using data augmentation and it outputted the last image. With this model, five detections were made and four of them predicted the class correctly. Both apples, an orange and the bananas were correctly classified and the other orange was misclassified as being a lemon.

It is important to note that all bananas were detected with only one bounding box. This brings us to another serious failure of the used dataset: some of the images extracted from the OID dataset [Vit18] apply the bounding box to a set of foods of the same class instead of considering them individually. Consequently, during the training phase, the models are learning the representation of the objects as a whole and will have more difficulty to detect a single object in the image.

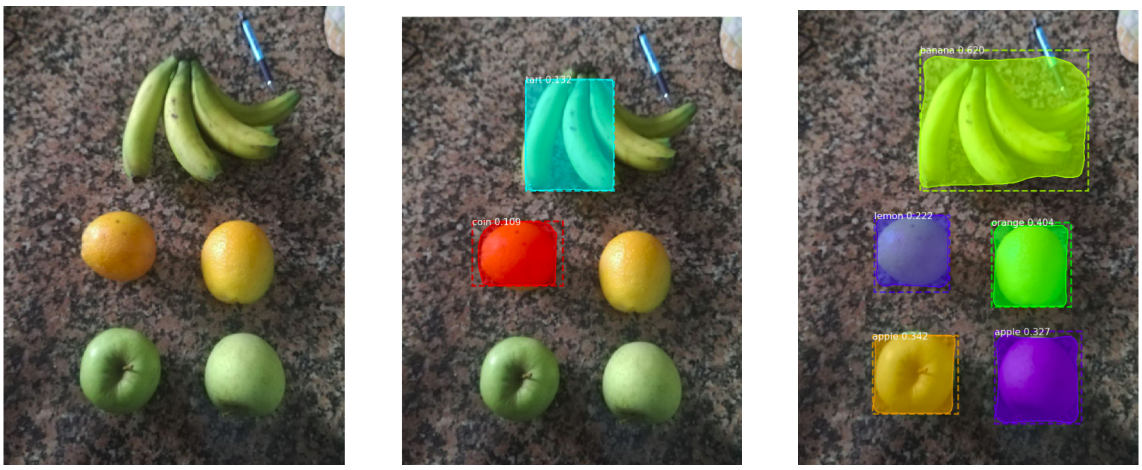


Figure 5.1: Mask R-CNN detections on the same image by using three different models whose evaluations are on Tables 5.1, 5.2, and 5.3

The Mask R-CNN outputs the bounding boxes that surround the object and in addition to it, a mask surrounding each object detected. Another aspect that is highlighted by Figure 5.1 is that, most of the time, the shape of the masks tends to be rectangular since none of the used datasets contained the labels to generate the correct masks and to overcome this misstep, the segmentation masks provided for the network to train coincide with the bounding boxes that delimit the objects.

In Figure D.0 and Figure E.0, more examples of detections made by the best model of the Mask R-CNN can be found. In the first one, there are some misclassified images and in the second there are well classified examples. These images are all part of the test subset, and in all of them it is possible to see the diversity of the background behind the food items introduced by using the modified dataset.

The authors of the original implementation [HGDG17] reported that the best results were achieved by training the models with the ResNet-101 architecture as backbone. To obtain the results of the Tables 5.1, 5.2 and 5.3, a ResNet-50 architecture was used as backbone. We also trained a model using ResNet-101 as backbone, but the results were not improved as can be seen in table 5.4. Therefore, the results presented hereinafter in the RetinaNet implementation 5.3 are all obtained using ResNet-50 as backbone.

Table 5.4: mAP with the modified dataset using Mask R-CNN with data augmentation during the training process and ResNet-101 as backbone

Subset	*mAP @ 0.50 IoU*	*mAP @ 0.75 IoU*
Train	0.321	0.314
Validation	0.434	0.417
Test	0.382	0.372

5.3 RetinaNet

The source-code of RetinaNet implementation [2] is a Python3 and Keras implementation of the network.

At this point, only the modified dataset was used to train the RetinaNet network. Each RetinaNet model was trained by 1000 epochs. The results obtained when using RetinaNet include mean Average Precision (mAP) with the thresholds of 0.5 and 0.75 and mAP using the weighted average of precisions among classes with the same thresholds. The difference between both metrics is that while the weighted metric computes the mAP as the average precision when all the classes are treated as a single class, the other metric is computed as the mean of the per-class average precisions [ten19], thus the weighted metric reflects the frequency of each class.

The results obtained by running RetinaNet implementation without using data augmentation are stored at Table 5.5, and the results obtained using the same configuration, but with data augmentation are attached to Table 5.6.

Table 5.5: mAP and corresponding mAP using the weighted average of precisions among classes using RetinaNet without data augmentation

Subset	*mAP @ 0.50 IoU*	mAP @ 0.50 IoU weighted	*mAP @ 0.75 IoU*	mAP @ 0.75 IoU weighted
Train	0.184	0.352	0.169	0.335
Validation	0.186	0.488	0.163	0.457
Test	0.185	0.429	0.158	0.398

Table 5.6: mAP and corresponding mAP using the weighted average of precisions among classes using RetinaNet with data augmentation

Subset	*mAP @ 0.50 IoU*	mAP @ 0.50 IoU weighted	*mAP @ 0.75 IoU*	mAP @ 0.75 IoU weighted
Train	0.707	0.776	0.586	0.672
Validation	0.466	0.715	0.383	0.656
Test	0.473	0.650	0.374	0.580

In this case, it is possible to verify the huge difference in the results when data augmentation is or is not being applied. The augmenters used were the ones already predefined into the used implementation and they can perform geometric or color modifications. The rotation, translation or shearing operations all range from [-0.1, 0.1]. The scaling operation is applied using a factor $f \in [0.9, 1.1]$ and the images and corresponding bounding boxes are flipped both horizontally and vertically with a probability of 0.5. To transform the images color, the contrast of the image varies

[2] `https://github.com/fizyr/keras-retinanet`

from 0.9 to 1.1; the brightness varies from -1 to 1; the hues shifting ranges from -0.05 to 0.05 and the saturation is changed by multiplying a term ranging from 0.95 to 1.05.

The geometric changes are shorter than those applied to Mask R-CNN, but the color changes may be responsible for improving the results obtained. Considering our specific task, recognizing food images that are mostly fruits, color is often a relevant feature.

Similar to what we did in Section 5.2, the same image was provided to both models to make the inference, and the resulting detections can be seen in Figure 5.2.

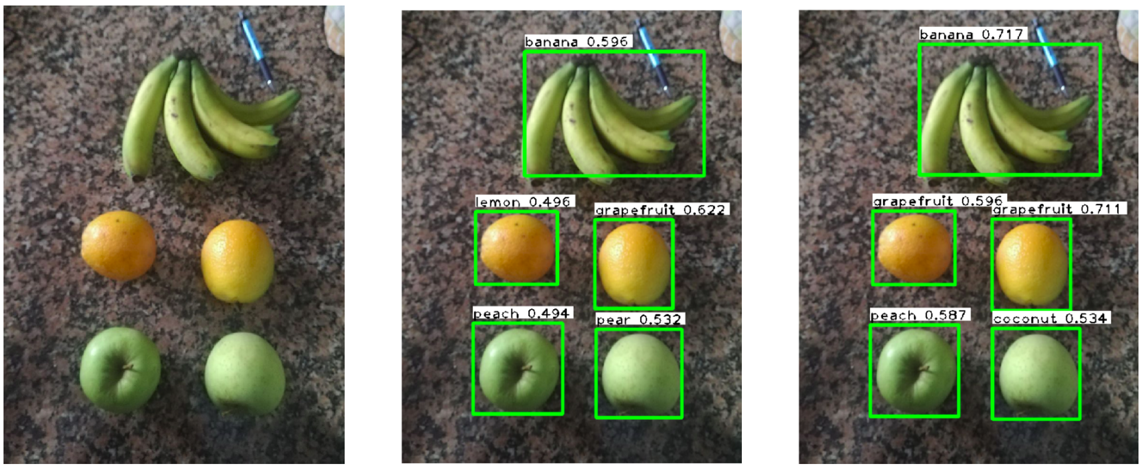


Figure 5.2: Detections on the same image by using both RetinaNet models whose evaluations are on Tables 5.5, and 5.6

In order to know which classes contribute negatively to mean Average Precision, mAP was calculated for each of the classes in the test subset, based on the model that gave the best results, the one in which data augmentation was applied 5.6. The mAP results for each class can be seen at Table 5.7.

Firstly, it appears that the classes that positively contribute to the global mAP are those that contain images of the ECUSTFD dataset 4.1.1.1. The column designated by *Number of instances* indicates the number of instances of each class that the test subset images contain and is clearly visible that the used dataset is affected by the **class imbalance** problem. Considering this problem, the classes that should affect mAP values the most would be those with fewer instances, which is not true as can be seen in Table 5.7, for example, the classes pineapple, taco, pasta, and pizza have values of mAP equal to or greater than 0.5 (considering the column where the *IoU* theshold is 0.5) and were found only 12, 13, 10 or 9 instances of this classes. Yet, classes such as carrot, cucumber, hot dog have 43, 116, and 48 instances per class, respectively and have lower values of mAP when compared to the values of the previously mentioned classes, all are less than 0.5.

Table 5.7: Mean Average Precision per class using RetinaNet

Number of instances	Class	mAP IoU=0.50	mAP IoU=0.75
103	apple	0.85	0.81
639	coin	0.99	0.99
66	banana	0.86	0.56
26	bread	0.62	0.62
16	bun	0.97	0.97
59	doughnut	0.86	0.82
44	egg	0.72	0.59
29	fired twist	1.00	0.93
30	grape	0.53	0.23
38	lemon	0.69	0.62
15	litchi	1.00	1.00
83	mango	0.86	0.83
72	orange	0.91	0.88
35	qiwi	0.97	0.97
33	mooncake	0.99	0.99
39	peach	0.81	0.77
51	pear	0.85	0.81
41	plum	0.91	0.91
27	sachima	1.00	1.00
74	tomato	0.42	0.36
78	croissant	0.19	0.07
9	tart	0.31	0.30
14	bagel	0.02	0.02
66	radish	0.27	0.16
39	strawberry	0.34	0.12
21	hamburger	0.26	0.04
21	burrito	0.05	0.01
37	coconut	0.14	0.00
9	pizza	0.50	0.47
11	coffee	0.56	0.45
11	tea	0.39	0.09
22	ice cream	0.13	0.01
25	waffle	0.34	0.02
43	carrot	0.11	0.05
19	cheese	0.03	0.02
12	pineapple	0.54	0.42
10	broccoli	0.43	0.26
9	cabbage	0.36	0.32
18	sushi	0.26	0.12
10	pomegranate	0.34	0.34
25	mushroom	0.18	0.02
28	grapefruit	0.14	0.12
12	milk	0.28	0.04
11	watermelon	0.07	0.07
48	hot dog	0.33	0.12
18	pancake	0.15	0.11
116	cucumber	0.23	0.07
13	taco	0.65	0.08
9	popcorn	0.45	0.42
10	pasta	0.54	0.30
10	potato	0.05	0.05
29	cookie	0.23	0.16
25	cake	0.16	0.12
10	french fries	0.46	0.23
12	pretzel	0.42	0.03
8	winter melon	0.18	0.13

5.4 Deeplabv3+

To train a Deeplab network, unlike the two previous networks mentioned above, besides the input images, the annotations must be provided in the form of mask images, where the pixels of each object represented in the image must have an associated color. As we are recognizing 57 different classes, the values of the pixels that contain a food object range from 0 to 57, as the background is represented by 0. Figure 5.3 shows two images provided to train the network and the corresponding segmentation labels.

For the Deeplab network, implemented by using Tensorflow, it was not possible to calculate the mAP performance metric as in previous networks. For each region with different color in the image for which the inference was made, we tried to find the bounding box that delimited that region in order to calculate the intersection between the bounding boxes of the predicted image

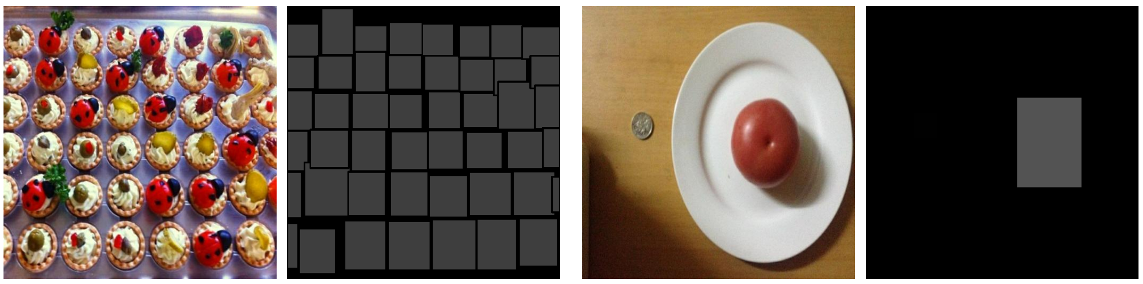


Figure 5.3: Deeplab training images and corresponding labels

and the bounding boxes of the ground truth image. To do it, we try to look at the implementation code of Mask R-CNN and reuse a function called extract_bboxes(mask) that receives a mask as a parameter, and returns the coordinates of the corresponding bounding box. The problem here is that the received mask has to be binary, and since the predicted masks have lots of colors, we would have to binarize the mask for each color which would be time consuming and would require computational effort. Therefore, we decided to present the results only in a visual way and they can be seen in Figure 5.4.

The results obtained with the application of this network were worse than those obtained with the use of the previous networks. This network was designed to perform image segmentation tasks and its goal is to understand an image at the pixel level. The predicted classes never coincide with the ground truth class even in the detections that are closest to the object at the spatial level. By providing masks that are bounding boxes for the training network, many pixels inside the bounding boxes do not represent the food object and this may explain the difficulty of this network to achieve better results.

The authors of the network in their publication [CZP+18] also reported that the training process was done using 50 GPUs asynchronously. Our Deeplab network was trained using only one of the three GPUs that Fraunhofer Institution made available to us. This leads us to believe that this network needed more training to give us better results.

5.5 Volume estimation

In this work, the food volume estimation is determined by using a single image depth map estimation. However, while the final solution was not found, other approaches were tested in order to estimate the volume of a food.

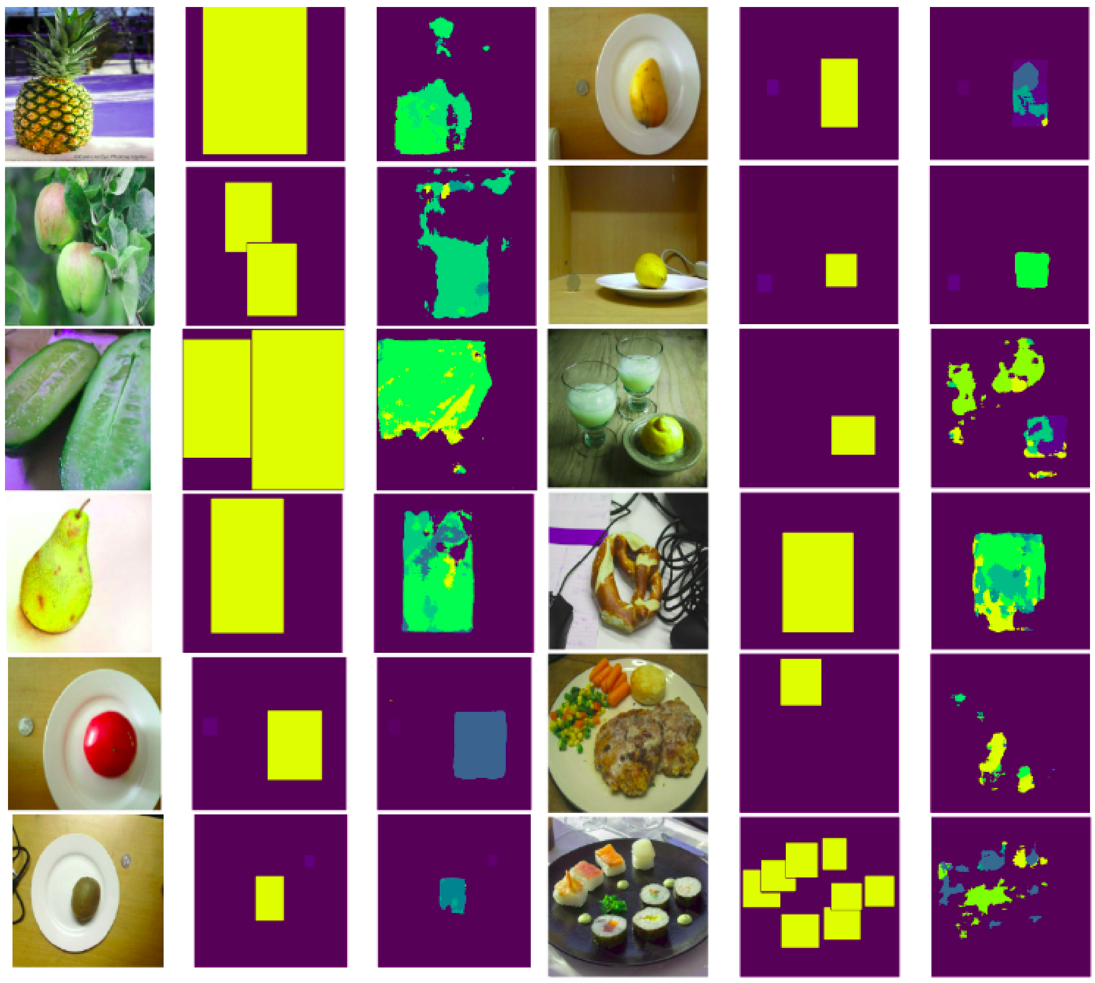


Figure 5.4: Deeplab visual segmentation results

5.5.1 GenRe-ShapeHD

The other alternative came with the discovery of the code repository [3] which is an implementation based on three different publications [ZZZ+18, WZZ+18, WWX+17].

This code implementation was designed to generate 3D models for unseen classes from models trained with only three classes: cars, chairs, and airplanes. There are three main modules in this implementation. The first module estimates a single-view depth map from an input colored image. The depth map is then converted into a spherical map to build the complete surface of an object. Zhang et. al [ZZZ+18] claim that: "the spherical maps only capture the outermost surface towards the sphere, they cannot handle self-occlusion along the sphere's radius" and to tackle this issue, they designed a third module to refine the voxel space by recovering the lost information.

To see if this implementation that generates 3D models could be feasible to the problem we

[3] https://github.com/xiumingzhang/GenRe-ShapeHD

want to address, we run the code and observe the outputs illustrated in Figure 5.5. The image 5.5a provided to this code implementation contained two objects, a coin and an external disk whose 3D models we wanted to obtain. An external disk was used instead of a food object because it was easier to calculate the volume occupied by the external disk. Figures 5.5d and 5.5e show the output of the 3D models generated from the coin and external disk objects. The 3D models were stored using a *.obj* extension [4] and in order to visualize the models, a website called *3D Viewer* [5] was utilized. As can be seen, both 3D models have similar shapes, and both of them are far away from the representations we expected to see.

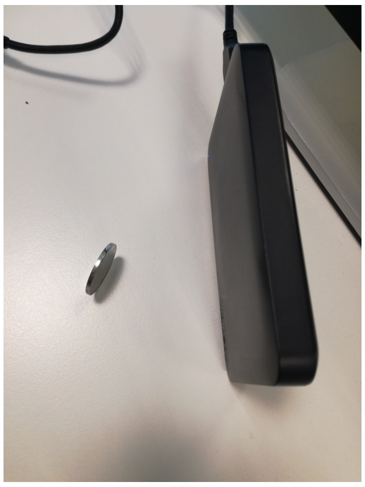
To extract the volume from the 3D models, we could convert them into voxels, and since we know the real volume of the coin, we could estimate what would be the volume of the other objects by simply count the number of voxels. A voxel is like a pixel but instead of representing a 2D image, it represents a 3D volume. To perform the voxelization, a website called *Online Voxelizer* [6] was used and the results for the coin and for the external disk voxelization can be seen in Figure 5.5b and 5.5c.

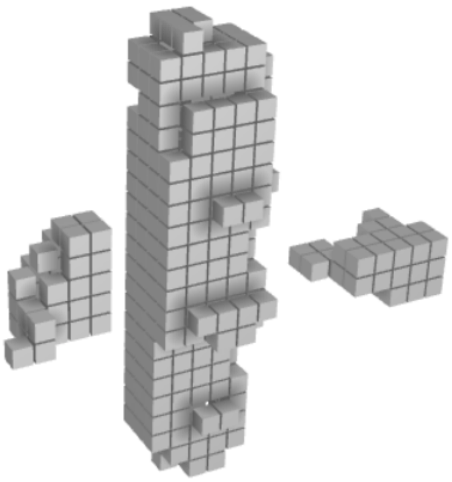
A coin has a cylindrical shape and the external disk has a parallelepiped shape. The real volume of the coin used in this experiment is approximately 981 mm^3, while the volume of the disk is much greater: 74240 mm^3. According to the voxelization obtained, the coin representation has 372 voxels and the external disk has only 313 voxels, meaning that the estimated volume for the coin using this method would be much bigger than the estimated volume for the disk, which makes no sense. Thus, we had to found out a new way for estimating the food object volume.

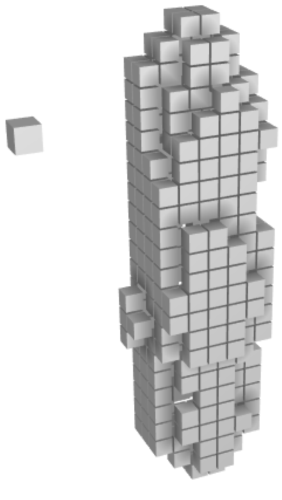
[4] *.obj* stands for Object File Wavefront 3D

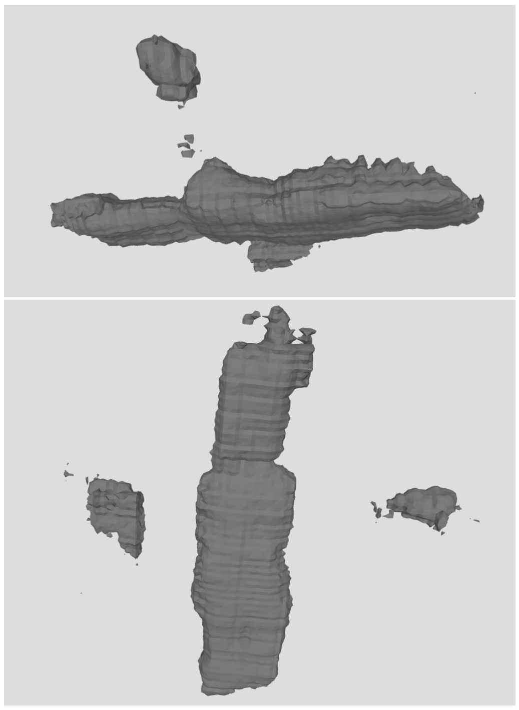
[5] https://3dviewer.net/

[6] https://drububu.com/miscellaneous/voxelizer/?out=obj


(a) Input image containing a coin and an external disk


(b) Voxelization of the coin 3D model


(c) Voxelization of the external disk 3D model


(d) 3D model generated from the coin

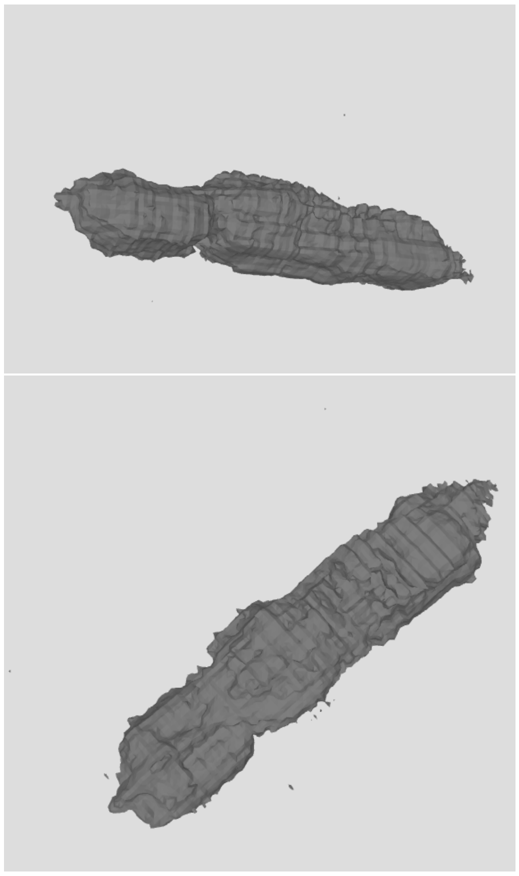

(e) 3D model generated from the external disk

Figure 5.5: Experiment with GenRE-ShapeHD

5.5.2 Depth Map Estimation

To estimate the volume of the food present in the image, a Pytorch code implementation [7] was used. In fact, this implementation is based on the implementation that supports the paper called *Revisiting Single Image Depth Estimation: Toward Higher Resolution Maps with Accurate Object Boundaries* by Hu et. al [HOZO18], whose main contributions have already been described in 3.6.

To perform the volume estimation, we need to provide the input image that contains the food objects for which we want to estimate the volume as well as a *.json* file containing the output of the detection networks, that is, the identified class for each detected object and the corresponding coordinates of the bounding boxes.

Through the input image provided, this code implementation generates the corresponding depth map image. The model that generates the depth images was trained using the NYU-Depth V2 dataset [SHKF12], which consists in indoor scenes images. The starting point for the volume estimation is the image that contains the depth map of the original image.

To perform the calculations, the image must contain an object whose dimensions are known in order to compare it with the object whose dimensions we want to find out. At the beginning, the calibration object to use would be the coin that is present in most of the images that belong to the used dataset. However, the depth of One Yuan coin is approximately $2mm$ and as can be seen in the depth map images 5.6, in most cases it is not even possible to identify it. Apart from the foods objects, the other object that can be easily identified in the depth map images is the plate. Therefore, the plate became the object that we use as a calibrator, so we need to know what are the real dimensions of the plate (diameter and height, or depth) that is in the image.

But with this decision, another problem arose: we did not try to recognize the plate in the first stage when we trained the detection networks, and worse, we do not have the labels of the bounding boxes of the plate in the images of the ECUSTFD dataset.

The Hough Transform algorithm was implemented with the help of the OpenCV library. This algorithm takes as input the original image and then tries to find a circle in the image that surrounds the plate [8]. More details about this algorithm can be consulted in Section 2.1.2.1.

By convention, the circles to be found have as minimum radium the smallest dimension of the image (width or height) divided by four and the maximum radium was defined as the maximum

[7] https://github.com/KaiwenZha/Food-Volume-Estimation/tree/master/Volume/Estimation

[8] https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html

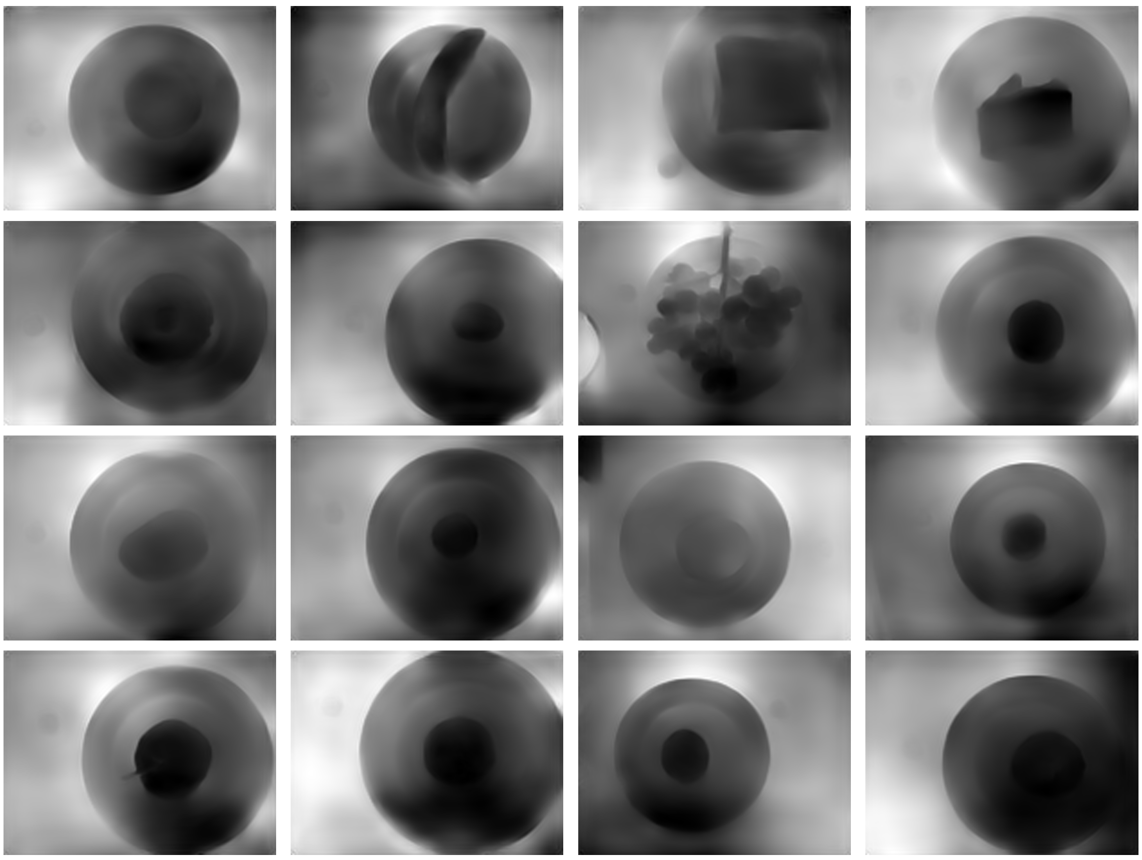


Figure 5.6: Depth map images generated from the same input images used to estimate the Table 5.8 volumes

dimension of the image, which is a valid assumption since, it is expected that the input image is a photograph in which the plate and food are fully captured in the image.

By applying this method, we can find the coordinates of the bounding box associated with the plate, since we knew the coordinates of the center of the circle relative to the image as well as its radius. The images in Figure 5.7 show the plate detection using the algorithm described above. As the implemented algorithm only finds circles in the image that correspond to the plate, only the images captured from the top view were chosen for the step of the volume estimation.

As the true diameter of the plate is known and we have the bounding box that surrounds the plate, we can find out what is the actual distance represented in a pixel, for convenience we can designate it as len_scale.

Following the same idea, we know what is the real plate depth, and we know that is represented by a certain colour in the depth map, so we can assign a depth to each color in the depth map and use it to estimate the volume by using the scale term called $depth\_scale$.

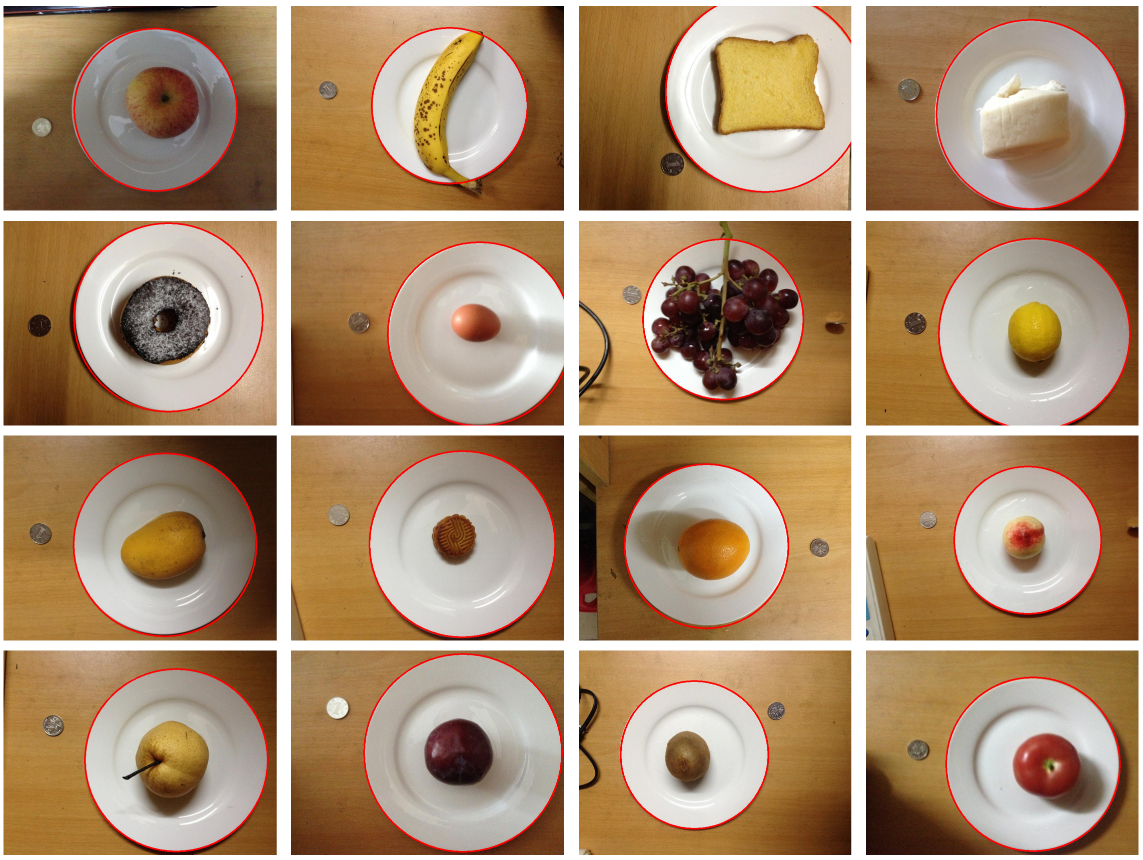


Figure 5.7: Images used to estimate the volume of Table 5.8 after plate detection

The mathematical formula that allows us to calculate volume is $Volume = height$ x $width$ x $depth$.

To estimate the volume of the food object, we iterate over the piece of the image that corresponds to the food through its bounding box and multiply the value of each pixel by $depth\_scale$ (to discover the real $depth$ of that pixel) and by len_scale^2 that is the size of $height$ x $width$.

The dimensions of the plate used in the ECUSTFD dataset are known, thus, we can use some of that dataset images to estimate the food volume and evaluate the results.

The bounding boxes of the food objects provided to estimate the volume are the ones that are attached to the ECUSTFD dataset, which means that the results obtained in this step are independent from the food segmentation task. The plate bounding boxes were calculated through the algorithm described above, the final result of the plate segmentation can be seen in Figure 5.7 and the volume estimation results are in Table 5.8.

After the volume of a food is estimated, its weight must be also estimated in order to infer the nutritional values of that food. The conversion from volume to weight has already been described in Section 4.1.2. Table 5.8 represents ground truth values for volume and weight for ECUSTFD

Table 5.8: Comparison between volume and weight values in ECUSTFD dataset and the values estimated by our system

Image ID	ECUSTFD Volume (cm^3)	ECUSTFD Weight (g)	Estimated Volume (cm^3)	Estimated Weight (g)	Error % Volume	Error % Weight
apple001T(1)	310	244.50	301.50	238	2.74	2.66
banana009T(3)	210	183.50	362.60	333	- 72.67	- 81.47
bread004T(2)	170	27.70	318	57	-87.06	-105.78
bun003T(2)	270	93.20	218.40	74	19.11	20.60
doughnut003T(2)	220	67.10	190.90	59	13.22	12.07
egg003T(4)	50	54.50	102.90	119	-105.80	-118.35
grape001T(2)	415	448	584.60	561	-40.87	-25.22
lemon001T(2)	100	94.20	135.1	129	-35.10	-36.94
mango001T(4)	180	192.30	162.90	180	9.50	6.40
mooncake001T(13)	40	39.90	57.70	17	- 44.25	57.39
orange001T(2)	160	220.50	176.40	167	-10.25	24.26
peach002T(6)	110	105.80	68.30	69	37.90	34.78
pear001T(2)	260	249	194.46	198	25.21	20.48
plum001T(20)	150	151.80	157.70	157	-5.13	-3.43
qiwi003T(2)	170	144.60	113.5	111	33.23	23.24
tomato004T(24)	160	173.40	143.10	141	10.56	18.69

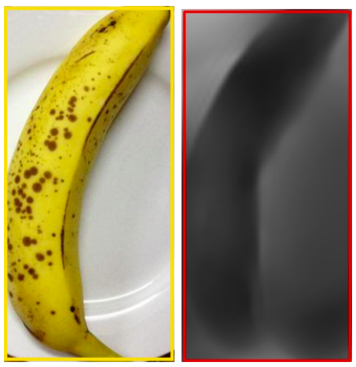
dataset images as well as the estimated volume and weight values by our implementation.

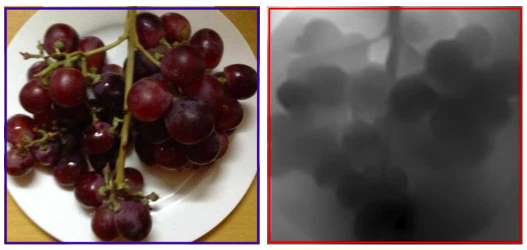
In addition, the percentage error for the volume and weight estimates has been calculated to make it easier to see how close or far the estimated values are from the ground-truth values. A negative value means that the estimated volume or weight was higher than the actual volume/weight of the food. Interestingly, it is also in these cases that the magnitude of the error is bigger.

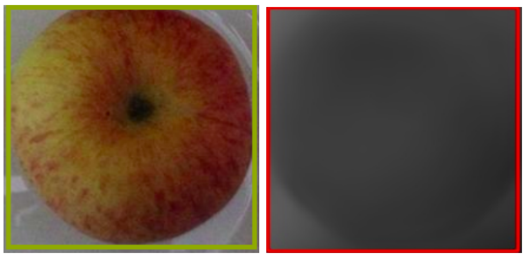
If we look at the pictures in Figure 5.8, in the top pictures, it is possible to verify that the bounding box that delimits the food object contains more pixels that do not belong to the food than the bounding boxes of the bottom pictures. It is probably for this reason that the volumes and weights of the foods in figure 5.8 were overestimated and the percentage error in these cases is of greater magnitude. This error could be alleviated if instead of considering the bounding boxes for volume calculation, the exact masks of each food were considered. In Section 5.6, an approach is suggested to obtain a reliable segmentation mask of food the objects.

Another strange aspect that can be noted when looking into the Table 5.8 is that for some images, the volume values were overestimated and on the contrary, the weight values were underestimated. When performing the conversion between volume values and weight values, we are considering the food densities of FSANZ as described in Section 4.1.2, but for the same category of food there are many species and each one has its specific density value. If we are not using the correct one, the weight estimate will not be correctly done.

There is another concern to have in mind in this step of volume estimation. In the experiments


(a) banana009T(3)


(b) grape001T(2)


(c) apple001T(1)

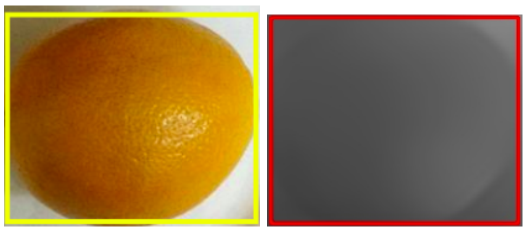

(d) orange001T(2)

Figure 5.8: Bounding boxes used to estimate the volume 5.8

of Liang et. al [LL17a, LL17b] food volume was measured using the drainage method and using their own words, the volume values that we are considering as ground-truth values may not be "accurate enough". Through this method, water is placed in a container in which the volume of water can be measured and is volume is noted. Then, the object whose volume is unknown is immersed in the water and the water volume is measured again, this time with the object. The difference between the two volumes corresponds to the volume of the object.

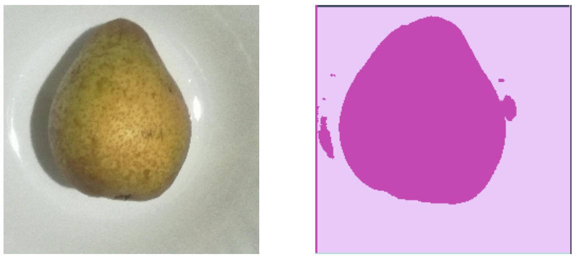
5.6 Segmentation refinement

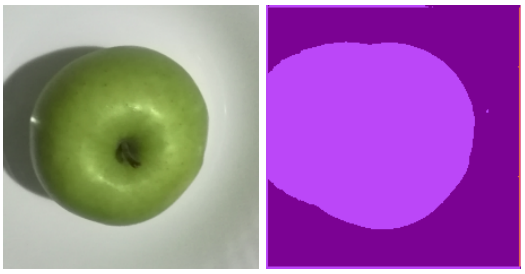
It is known that using the segmentation masks coinciding with the food objects instead of the bounding boxes that surround them could give better results in the volume estimation, however, with the dataset used for the first task, segmentation and classification of food objects in an image, it was not possible to obtain this accurate segmentation mask of the food objects. The network that gave us the best results in the first task was Mask R-CNN and this network outputs the bounding boxes that surround the food object as well as the segmentation mask. Yet, as already mentioned in 5.2, the segmentation masks obtained are very close to the bounding boxes that surround the food object, since the masks we provide for training the network also coincide with the bounding boxes. For this reason, we used the bounding boxes outputted by the network and discard the outputted masks for the task of volume estimation.

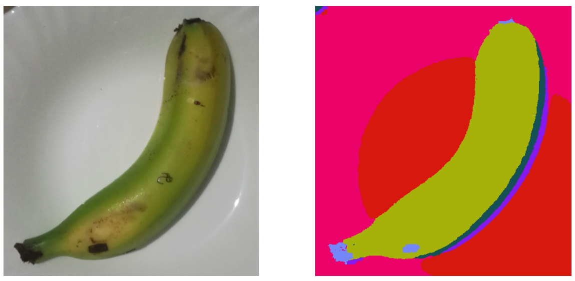
Considering this, another question arose: once we have the food objects surrounded by bounding boxes, will not be possible to obtain accurate segmentation masks of the food objects contained inside the region bounded by the bounding box? To answer this question, another code implementation [9] was explored and it supports the publication of Asako Kanezaki [Kan18].

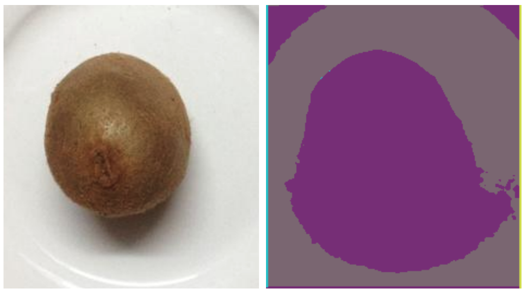
This implementation was used to check whether the segmentation food masks were accurate enough to be used later in the volume estimation. A brief description of the concepts behind this implementation can be found in Section 3.7.

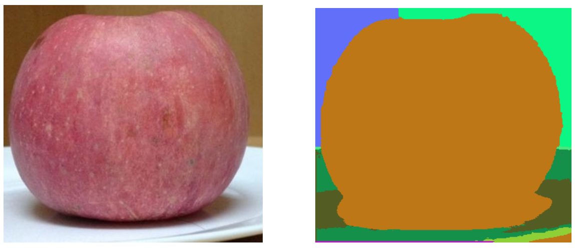
The results, the images on the right side, were obtained by using as input images the ones on the left side of each pair in Figure 5.9 and Figure 5.10.


(a) 1.png


(b) 4.png


(c) 9.png


(d) qiwi003T.png


(e) apple001T.png

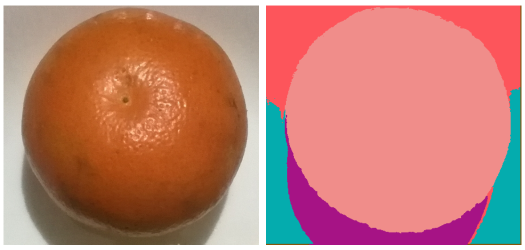
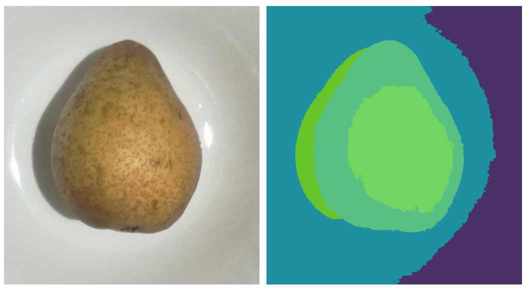
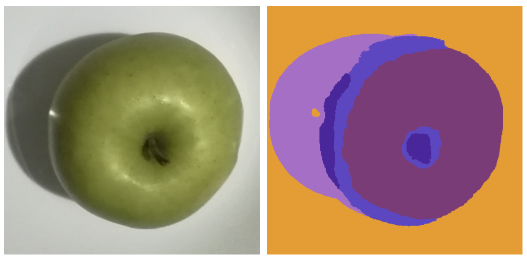

(f) 14.png

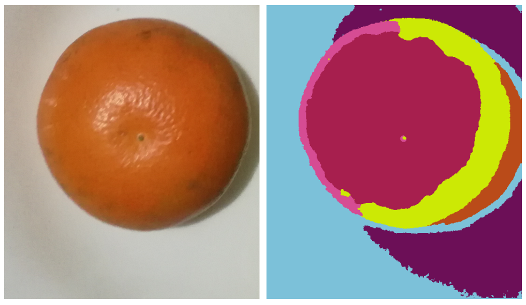
Figure 5.9: Images cropped by the contours of the bounding boxes and segmentation masks obtained by using Asako Kanezaki [Kan18] implementation

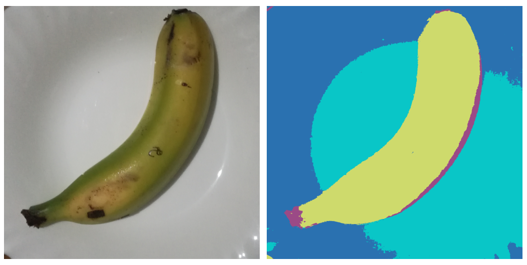
For each image tested, the process of obtaining the segmentation mask took about 2 minutes and 40 seconds providing the bounding box around the food object, if we used the entire image

[9]https://github.com/kanezaki/pytorch-unsupervised-segmentation


(a) 1.png


(b) 4.png


(c) 11.png


(d) 9.png

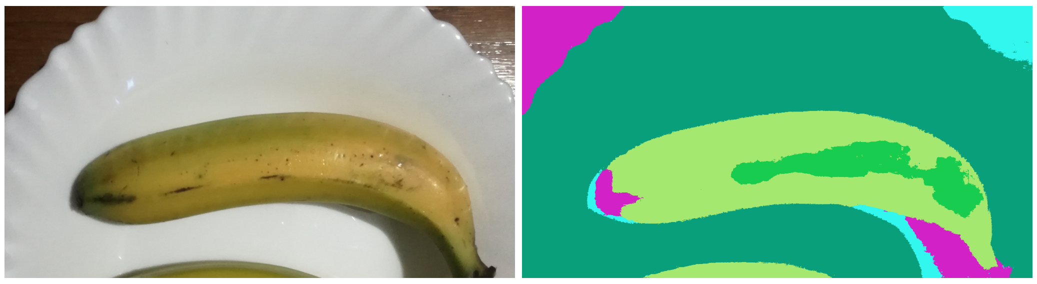

(e) 7.png

Figure 5.10: Objects cropped from the Figure C.1 and the corresponding segmentation masks obtained by using Asako Kanezaki [Kan18] implementation. This images were used to estimate the volumes in Table 5.9

this processing could take approximately up to 6 minutes. We must take into account that this value is high and that it is necessary to optimize the time spent during the segmentation of the food. The segmentation masks obtained are promising and could lead to better estimates of the food volume, since this way, only the pixels of the image that correspond to the food object would be considered.

As can be seen in Table 5.9, this assumption proved to be correct, because when using the segmentation masks on the objects in Figure 5.10, the error of the estimated volumes with the segmentation masks decreased in relation to the estimated volumes using only the bounding boxes that surrounded the objects. The original images of the objects are those shown in the Figure C.1.

Table 5.9: Comparison between the real volume and the estimated volume using the bounding box and the segmentation mask

Image ID	Food Item	Real Volume (cm^3)	Bbox Volume (cm^3)	Error Bbox %	Segmentation Volume (cm^3)	Error Segmentation %
1.png	Pear	179.60	266.43	48.35	112.67	-37.27
4.png	Apple	268	228.36	-17.36	192.94	- 28.00
7.png	Banana_1	188	283.81	50.96	193.64	3
9.png	Banana_2	175	466.62	62.49	209.02	19.44
11.png	Orange	113	193.83	71.53	165.55	46.50

The model chosen to do the first task of classifying and segmenting images is Mask R-CNN with data augmentation, since it is the model that had the best evaluation. When calculating the mean Average Precision with the different IoU thresholds, 0.50 and 0.75, the models trained by using Mask R-CNN are the ones which have less discrepant results when using bigger IoU thresholds. In fact, the best model of Mask R-CNN achieved a mAP of 50.2% when using $IoU = 0.5$ and 46.6% when $IoU = 0.75$ as can be seen in Table 5.3. On its turn, the best RetinaNet model achieved a mAP of 47.3% when using $IoU = 0.5$ and 37.64% when $IoU = 0.75$ as can be seen in Table 5.6.

In addition to being the one that gave the best results, it is the most complex network and the one that takes the longest time to be trained. RetinaNet models took three days to be trained, while the Mask R-CNN models took approximately ten days to be trained (each one) and the training process of Deeplab lasted for a week.

For estimating the volume, two approaches were followed: by reconstructing the 3D models of the objects and by using the depth map estimation of the input image. The first one proved to be very ineffective, so it was decided to explore the second approach.

Alongside with this implementation it was necessary to find an algorithm that was able to segment the plate in the image. The advantage of using this method instead of what is used in the implementation of Liang et. al [LL17a, LL17b] is that in this alternative, the volume calculation is done regardless of the shape of the food in the image and by using only one input image.

To evaluate the results obtained in the volume estimate, an image of each class of the ECUSTFD dataset was taken, and the value estimated by our implementation was compared with the real value that was in the annotations of the dataset. However, the authors report that the food volumes in their experiment are not reliable since they measured them though the use of the drainage method. For some classes, the volumes estimated by our implementation are very similar, however in some cases they are very divergent, which means that our approach to estimate the volume has to be improved before it can be used. The improvement could involve the use of food object segmentation masks instead of the bounding boxes as discussed in Section 5.6.

To sum up, all the entire system was tested with 16 images captured by a mobile phone, that contain 5 different food items: a pear, an apple, two bananas and an orange.

The input images are the first image of each row of Appendix C.1. When providing the input image to the system, the inference is made using the Mask R-CNN model that gave us better results and the class and bounding boxes coordinates are extracted to the next steps. The images of the detections are the second images in each row of the Figure C.1.

To estimate the volume, the coordinates of the bounding boxes that surround the plate are also needed. Therefore, the input images are fed to the algorithm that detects the plate and through it, we get the coordinates of the bounding boxes that bound the plate. In Figure C.1, the plate detection results are the images that are in the third position of each row. The plate used in this experiment has different dimensions: it has 22.5 *cm* of diameter and 2.3 *cm* of height. This means that the proposed method requires the actual dimensions of the plate used in the volume estimation phase and the absence of a plate in the image makes it impossible to estimate the volume of the food because there is no way to extract the scale from the images.

Once we have the bounding boxes of the food objects and the plate, we can move to the last stage of the project. The depth image estimations correspond to the last image of each row of the Figure C.1. From these images, the volume is estimated as already explained in Section 5.5.2. The estimated results for these 16 images are in Table 5.11. From the volume estimate, we are able to estimate the weight of each food item, as Section 4.1.2 describes, and after having the weight values, we can find out what the nutritional values of each food item. In addition to the carbohydrates amount, we also included the food energy, proteins and fibers.

To make any comparison possible, we provide the real weight of each food item in the 16 images, as well as an estimated volume from the approximation to geometric shapes in Table 5.10.

We decided use multiple images of the same food item to see if the volume estimates changed significantly. We can see that:

- The pear has a real weight of 198.5g and the weight estimates vary between 211g and 213g. The pear item was in images 1.png, 2.png, and 3.png.

- The apple has a real weight of 220g and the weight estimates vary between 174g and 269g. The apple item was in images 4.png, 5.png, 6.png, 14.png, 15.png, and 16.png. It was also present in the images 12.png and 13.png, but it was not detected in the previous stage, this way it was ignored in the volume estimation.

- The first banana, called $Banana_1$, has a real weight of $130g$ and the estimates ranges from $260g$ to $303g$. The $Banana_1$ item was in images 7.png, 8.png, 12.png, and 13.png.
- In its turn, $Banana_2$, is the food item were the estimates are more erroneous. Its real volume is $160g$ and the estimates vary between $429g$ to $629g$. The $Banana_2$ item was in images 7.png, 8.png, 9.png, 10.png, 11.png, 12.png and 13.png.
- The last food item was classified as an orange, but it is actually a tangerine with a real weight of $65g$ and the estimates vary from $113g$ to $193g$. The orange item was in images 10.png, 11.png, 12.png, 13.png, 14.png, 15.png and 16.png.

We can conclude that the volume estimates are more inaccurate for banana and orange foods and that, on the other hand, they work better for foods like pear and apple. The difference between the real value and the estimated value is much greater in the case of bananas. This can happen due to the bounding boxes that surround the bananas, being the ones that include more pixels that do not correspond to the food object.

An improvement on the volume estimation was achieved by using the Asako Kanezaki [Kan18] implementation, and some of the images of the Figure C.1 were chosen to obtain the segmentation masks of the objects. The refined segmentation results can be seen in Figure 5.10 and the results of the volume estimates in Table 5.9 show that using the object segmentation masks instead of their bounding boxes leads to a minor error in the volume estimation.

Table 5.10: Real volume (cm^3) and weight (g) of the foods in Figure C.1

	Pear	Apple	Banana$_1$	Banana$_2$	Orange
Volume (cm^3)	179.60	268	188	175	113
Weight (g)	198.50	220	130	160	65

Table 5.11: Final results: volume, weight, carbohydrates, energy, proteins, and fiber

Image_ID	Food Item	Volume	Weight	Carbohydrates	Energy	Proteins	Fiber
1.png	Pear	$266.43cm^3$	$213g$	$33g$	$123kcal$	$1g$	$6g$
2.png	Pear	$266.76cm^3$	$213g$	$33g$	$123kcal$	$1g$	$6g$
3.png	Pear	$263.93cm^3$	$211g$	$33g$	$122kcal$	$1g$	$6g$
4.png	Apple	$228.36cm^3$	$180g$	$24g$	$93kcal$	$0g$	$2g$
5.png	Apple	$275.80cm^3$	$217g$	$29g$	$133kcal$	$0g$	$3g$
6.png	Apple	$236.17cm^3$	$186g$	$25g$	$97kcal$	$0g$	$2g$
7.png	Banana_1	$283.81cm^3$	$261g$	$59g$	$232kcal$	$2g$	$4g$
	Banana_2	$502.28cm^3$	$462g$	$105g$	$411kcal$	$3g$	$7g$
8.png	Banana_1	$313.72cm^3$	$288g$	$65g$	$256kcal$	$2g$	$4g$
	Banana_2	$598.37cm^3$	$550g$	$125g$	$489kcal$	$4g$	$9g$
9.png	Banana_2	$466.62cm^3$	$429g$	$98g$	$381kcal$	$3g$	$7g$
10.png	Orange	$203.21cm^3$	$193g$	$22g$	$91kcal$	$1g$	$2g$
	Banana_2	$477.98cm^3$	$439g$	$100g$	$390kcal$	$3g$	$7g$
11.png	Orange	$193.83cm^3$	$184g$	$21g$	$87kcal$	$1g$	$2g$
	Banana_2	$471.74cm^3$	$434g$	$99g$	$386kcal$	$3g$	$7g$
12.png	Banana_1	$282.61cm^3$	$260g$	$28g$	$109kcal$	$1g$	$2g$
	Orange	$134.17cm^3$	$123g$	$128g$	$109kcal$	$1g$	$2g$
	Banana_2	$502.02cm^3$	$661g$	$105g$	$410kcal$	$3g$	$7g$
13.png	Banana_1	$329.86cm^3$	$303g$	$60g$	$269g$	$2g$	$5g$
	Orange	$114.13cm^3$	$104g$	$23g$	$92kcal$	$0g$	$1g$
	Banana_2	$425.60cm^3$	$629g$	$143g$	$550kcal$	$5g$	$10g$
14.png	Orange	$186.18cm^3$	$176g$	$20g$	$83kcal$	$1g$	$2g$
	Apple	$220.99cm^3$	$174g$	$23g$	$90kcal$	$0g$	$2g$
15.png	Orange	$167.60cm^3$	$150g$	$18g$	$75kcal$	$1g$	$2g$
	Apple	$341.35cm^3$	$269g$	$37g$	$140kcal$	$0g$	$3g$
16.png	Orange	$119.46cm^3$	$113g$	$12g$	$53kcal$	$0g$	$1g$
	Apple	$239.04cm^3$	$188g$	$25g$	$98kcal$	$0g$	$2g$

Chapter 6

Conclusions and future work

In this chapter, some conclusions which have already been discussed in chapter 5 will be highlighted and some proposals for improvements will be discussed, not forgetting that the ultimate goal of this dissertation is to develop a system that helps diabetics to automatically count the amount of carbohydrates present in their meals through a single photograph.

If we put a plate of food in front of an expert dietician, s/he cannot give an accurate measurement of its nutritional facts by simply looking at it or even examining it manually, because it is impossible to know the exact contents of the dish, such as if this dish contains salt, and if so how much, or contains oil, and if so what type (olive, corn, or animal-based), etc. Therefore, it is important to understand that a picture is not enough to extract the nutritional information of the food, when we are dealing with processed food.

6.1 Data Quality

It is undeniable that the success or failure of the developed work was influenced from the beginning by the dataset that was used.

Of the three networks used to classify and segment objects in an image, the used dataset only contemplates the requirements for training RetinaNet, that is, classification labels and bounding-boxes around the objects to segment. Both Deeplab and Mask R-CNN also required the segmentation masks of the objects to be trained and get the most out of their models. In addition to being incomplete, as it does not contain segmentation masks, the used dataset was also not annotated in the best way. There are some images that belong to the dataset with several instances of the same food and sometimes these instances are close to each other. In these cases, the bounding boxes cover all food objects and just one classification label is assigned to it. This reduces the performance of the models when they have to detect a single object belonging to that class.

Another aspect that should be reflected in the used dataset is that the classification labels should be assigned more carefully in order to define the object food that is being classified. For example, classifying an apple with the label *apple* is not enough. There are different species for apples, like *RedDelicious*,*Beacon* or *Cortland* and not all types have the same density (essential for the conversion of volume into weight), or the same nutritional values which directly affects the amount of carbohydrates present in the food. The same goes for the label *bread*, we know that there are many types of bread and all have different carbohydrates quantities, therefore, the classification labels should be accordingly to the type of the food.

In this project, some classes like *hamburger*,*pasta*,*pancake*,*pizza* or *taco* were considered and they should not have been because these foods are made up of different ingredients, and their nutritional value varies depending on the ingredients added, so the classes that should be considered for recognition are the classes of the ingredients and not the classes that correspond to the whole food.

In short, in order to develop a complete system like the one proposed in this project, it is essential to have a complete dataset: with the classification labels, the bounding-boxes that surround the food and the segmentation masks of the same. These are the requirements for training the Mask R-CNN network in order to obtain the greater efficiency and it is important to note that there are no public available datasets that meet these requirements. Besides that, the annotations of the dataset must respect the conditions that have been mentioned throughout this section: the labels should be assigned to each food individually, not to a set of foods of the same type; the labels used in the classification must specify the type of food, for example, *RedDelicious* instead of *apple*, and they should relate to the ingredients of the meal rather than classifying it as a whole.

To summarize, this project was divided into two parts: classifying and segmenting the input images and estimating the volume of food items present in these images.

The first task was achieved through the use of Mask R-CNN network, which revealed better results than the other networks, RetinaNet and Deeplab. The difficulty of having a dataset that did not include segmentation masks was overcome by using the regions defined by the bounding boxes as segmentation masks.

To estimate the volume of the food present in the image, the most promising approach was through the generation of depth maps of the input images instead of generating their 3D models. The drawback of this approach is the requirement for an object in the image that can be used to extract the scale. At the beginning, the object would be the One Yuan coin that belongs to all images of the ECUSTFD dataset 4.1.1.1, however, it was not visible in the depth maps images. As the plate and the food items are the ones that stands out in the depth map images, the plate became the calibration object. With this decision, the problem was that the plate class was not considered in the first part, thus, we had to develop an algorithm to identify it and provide the coordinates of the surrounding bounding box.

After the volume estimate, it was necessary to infer about the weight of the food items. In order to do so, the FSANZ density tables were consulted. As final result, in addition to being able to infer about the amount of carbohydrates, the developed system also allows to estimate the calories, proteins and fibers existing in the foods present in the input image.

6.2 Future work

The results of the overall system are good, but there is margin to improve them.

The existence of a dataset that fulfills the requirements described in 6.1 would improve the performance of the overall system.

In addition to the necessary dataset for the image classification and segmentation task, the results of the volume estimation process could be improved if the network that generates the depth map images was trained by using depth images of food instead of the indoor scenes images. There is also no public dataset that contains food depth images.

The same can be said for the first tested approach to estimate the food volume. The 3D models of the images were generated by a network that was only trained with 3 classes: cairs, chairs, and airplanes. If 3D models of food existed, this network could be trained with them and be able to generate accurate 3D models of food.

The creation of quality datasets with regard to food can boost the image classification and segmentation task as well as the food volume estimation.

If the results obtained were intensely improved, a mobile application could be developed and tested by diabetic patients under medical supervision.

References

[ADS+15] Marios Anthimopoulos, Joachim Dehais, Sergey Shevchik, Botwey H Ransford, David Duke, Peter Diem, and Stavroula Mougiakakou. Computer vision-based carbohydrate estimation for type 1 patients with diabetes using smartphones. *Journal of diabetes science and technology*, 9, 04 2015.

[AGS+14] M. M. Anthimopoulos, L. Gianola, L. Scarnato, P. Diem, and S. G. Mougiakakou. A food recognition system for diabetic patients based on an optimized bag-of-features model. *IEEE Journal of Biomedical and Health Informatics*, 18(4):1261–1271, July 2014.

[AMFM11] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[AVPS12] R. Almaghrabi, G. Villalobos, P. Pouladzadeh, and S. Shirmohammadi. A novel method for measuring nutrition intake based on food image. In *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pages 366–370, May 2012.

[Bal81] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111 – 122, 1981.

[BGVG14] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 446–461, Cham, 2014. Springer International Publishing.

[BMS+09] Franziska K. Bishop, David M. Maahs, Gail Spiegel, Darcy Owen, Georgeanna J. Klingensmith, Andrey Bortsov, Joan Thomas, and Elizabeth J. Mayer-Davis. The carbohydrate counting in adolescents with type 1 diabetes (ccat) study. *Diabetes Spectrum*, 22(1):56–62, 2009.

[BZK+11] M. Bosch, F. Zhu, N. Khanna, C. J. Boushey, and E. J. Delp. Combining global and local features for food identification in dietary assessment. In *2011 18th IEEE International Conference on Image Processing*, pages 1789–1792, Sep. 2011.

[CBD+90] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Advances in neural information processing systems 2. chapter Handwritten Digit Recognition with a Back-propagation Network, pages 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[Cho16] François Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2016.

[CM02] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[CN16] Jingjing Chen and Chong-wah Ngo. Deep-based ingredient recognition for cooking recipe retrieval. In *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, pages 32–41, New York, NY, USA, 2016. ACM.

[CNS17a] G. Ciocca, P. Napoletano, and R. Schettini. Food recognition: A new dataset, experiments, and results. *IEEE Journal of Biomedical and Health Informatics*, 21(3):588–598, May 2017.

[CNS17b] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. Learning cnn-based features for retrieval of food images. In Sebastiano Battiato, Giovanni Maria Farinella, Marco Leo, and Giovanni Gallo, editors, *New Trends in Image Analysis and Processing – ICIAP 2017*, pages 426–434, Cham, 2017. Springer International Publishing.

[CPK$^+$14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR. arXiv*, 12 2014.

[CPK$^+$16] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 06 2016.

[CPSA17] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *ArXiv*, abs/1706.05587, 2017.

[CZP$^+$18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. 02 2018.

[DH72] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[DM01] Yining Deng and B Manjunath. Manjunath, b.s.: Unsupervised segmentation of color-texture regions in images and video. ieee trans. on pami 23(8), 800-810. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 10 2001.

[DY14] Li Deng and Dong Yu. Deep learning: Methods and applications. *Found. Trends Signal Process.*, 7(3&#8211;4):197–387, June 2014.

[EVGW$^+$] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[FAM+16] Giovanni Maria Farinella, Dario Allegra, Marco Moltisanti, Filippo Stanco, and Sebastiano Battiato. Retrieval and classification of food images. *Computers in Biology and Medicine*, 77:23 – 39, 2016.

[FLD+16] Shimin Fu, Linjun Li, Shuhua Deng, Liping Zan, and Zhiping Liu. Effectiveness of advanced carbohydrate counting in type 1 diabetes mellitus: A systematic review and meta-analysis. *Scientific Reports*, 6:37067, 11 2016.

[FZJ+16] S. Fang, F. Zhu, C. Jiang, S. Zhang, C. J. Boushey, and E. J. Delp. A comparison of food portion size estimation using geometric models and depth images. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 26–30, Sep. 2016.

[GB10] Xavier Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 01 2010.

[GH95] C. A. Glasbey and G. W. Horgan. *Image Analysis for the Biological Sciences*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[Gir15] Ross B. Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[GLL18] A. Gao, F. P. . Lo, and B. Lo. Food volume estimation for quantifying dietary intake with a wearable camera. In *2018 IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pages 110–113, March 2018.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[HKT16] H. He, F. Kong, and J. Tan. Dietcam: Multiview food recognition using a multikernel svm. *IEEE Journal of Biomedical and Health Informatics*, 20(3):848–855, May 2016.

[HMC+16] Hamid Hassannejad, Guido Matrella, Paolo Ciampolini, Ilaria De Munari, Monica Mordonini, and Stefano Cagnoni. Food image recognition using very deep convolutional networks. In *Proceedings of the 2Nd International Workshop on Multimedia Assisted Dietary Management*, MADiMa '16, pages 41–49, New York, NY, USA, 2016. ACM.

[Hos19] Nationwide Children's Hospital. Calculating bolus injections. https://www.nationwidechildrens.org/family-resources-education/health-wellness-and-safety-resources/resources-for-parents-and-kids/managing-your-diabetes/chapter-seven-calculating-bolus-injections, 2019. Accessed: June 2019.

[HOZO18] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1043–1051, 2018.

[HW62] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154, 1962.

[HXK$^+$14] Y. He, C. Xu, N. Khanna, C. J. Boushey, and E. J. Delp. Analysis of food images: Features and classification. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2744–2748, Oct 2014.

[HZRS14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37, 06 2014.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[HZZ$^+$18] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, 2018.

[JF91] Anil K. Jain and Farshid Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern Recognition*, 24(12):1167 – 1186, 1991.

[JKRL09] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153, Sep. 2009.

[JM00] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000.

[JY09] Taichi Joutou and Keiji Yanai. A food image recognition system with multiple kernel learning. volume 285-288, pages 285 – 288, 12 2009.

[JZL$^+$19] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.

[Kan18] A. Kanezaki. Unsupervised image segmentation by backpropagation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1543–1547, April 2018.

[KDA$^+$16] Ivan Krasin, Tom Duerig, Neil Alldrin, Andreas Veit, Sami Abu-El-Haija, Serge Belongie, David Cai, Zheyun Feng, Vittorio Ferrari, Victor Gomes, Abhinav Gupta, Dhyanesh Narayanan, Chen Sun, Gal Chechik, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2016.

[KHB$^+$18] Sateesh Kumar, Sanjay Haresh, Bahram Baloch, Abeera Rehman, and Tahir Syed. Focused-anchors loss for imbalanced classification. 12 2018.

[KS00] Kiriakos Kutulakos and Steven Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38:199–218, 01 2000.

[KSEH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[KSZS19] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed. A survey of the recent architectures of deep convolutional neural networks. 01 2019.

[KY15] Yoshiyuki Kawano and Keiji Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In Lourdes Agapito, Michael M. Bronstein, and Carsten Rother, editors, *Computer Vision - ECCV 2014 Workshops*, pages 3–17, Cham, 2015. Springer International Publishing.

[LAE$^+$16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. Ssd: Single shot multibox detector. volume 9905, pages 21–37, 10 2016.

[LBH15] Yann LeCun, Y Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[LDG$^+$17] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, July 2017.

[LGG$^+$18] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2018.

[LL17a] Yanchao Liang and Jianhua Li. Computer vision-based food calorie estimation: dataset, method, and experiment. *ArXiv*, abs/1705.07632, 2017.

[LL17b] Yanchao Liang and Jianhua Li. Deep learning-based food calorie estimation method in dietary assessment. *ArXiv*, abs/1706.04062, 2017.

[LMB$^+$14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[LSQL18] Po Wen Lo, Yingnan Sun, Jianing Qiu, and Benny Lo. Food volume estimation based on deep learning view synthesis from a single depth map. *Nutrients*, 10:2005, 12 2018.

[Mat16] MathWorks. Local feature detection and extraction. https://www.mathworks.com/help/vision/ug/local-feature-detection-and-extraction.html, 2016. Accessed: June 2019.

[MFTM01] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423 vol.2, July 2001.

[MJR$^+$15] A. Myers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. Murphy. Im2calories: Towards an automated mobile vision food diary. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1233–1241, Dec 2015.

[Mor19] João Pedro Carvalho Leal Mendes Moreira. *A general introduction to data analytics*. 2019.

[NMLJ15] E Nagapavani, C.H. Manvitha, and J Lydia Jeba. Large scale learning for food image classification. *International Journal of Applied Engineering Research*, 10:9167–9174, 01 2015.

[oC13] Government of Canada. Nutrient value of some common foods. https://www.canada.ca/en/health-canada/services/food-nutrition/healthy-eating/nutrient-data/nutrient-value-some-common-foods-2008.html, 2013. Accessed: December 2019.

[oM19] U.S. National Library of Medicine. Insulin injection. https://medlineplus.gov/druginfo/meds/a682611.html, April 2019. Accessed: June 2019.

[Org16] World Health Organization. Global report on diabetes. http://www.who.int/iris/handle/10665/204871, 2016. Accessed: May 2019.

[PSA14] P. Pouladzadeh, S. Shirmohammadi, and R. Al-Maghrabi. Measuring calorie and nutrition from food image. *IEEE Transactions on Instrumentation and Measurement*, 63(8):1947–1956, Aug 2014.

[PZY$^+$09] M. Puri, Zhiwei Zhu, Q. Yu, A. Divakaran, and H. Sawhney. Recognition and volume estimation of food intake using a mobile device. In *2009 Workshop on Applications of Computer Vision (WACV)*, pages 1–8, Dec 2009.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. pages 779–788, 06 2016.

[RGG14] Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. Non-maximum suppression for object detection by passing messages between windows. In *ACCV*, 2014.

[RGGEJC00] Marilyn R Graff, Todd Gross, Suzanne E Juth, and Jean Charlson. How well are individuals on intensive insulin therapy counting carbohydrates? *Diabetes Research and Clinical Practice - DIABETES RES CLIN PRACT*, 50:238–239, 09 2000.

[RTG$^+$19] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression, 02 2019.

[RW17] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29:1–98, 06 2017.

[SAM19] M. A. Subhi, S. H. Ali, and M. A. Mohammed. Vision-based approaches for automatic food recognition and dietary assessment: A survey. *IEEE Access*, 7:35370–35381, 2019.

[SBB$^+$15] Mingui Sun, Lora Burke, Tom Baranowski, John Fernstrom, Hong Zhang, Hsin-Chen Chen, Yicheng Bai, Yuecheng Li, Chengliu Li, Yaofeng Yue, Zhen Li, Jie Nie, Robert Sclabassi, Zhi-Hong Mao, and Wenyan Jia. An exploratory study on a chest-worn computer for evaluation of diet, physical activity and lifestyle. *Journal of healthcare engineering*, 6:1–22, 03 2015.

[SHKF12] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[SKMC12] C. E. Smart, B. R. King, P. McElduff, and C. E. Collins. In children using intensive insulin therapy, a 20-g variation in carbohydrate amount significantly impacts on postprandial glycaemia. *Diabetic Medicine*, 29(7):e21–e24, 2012.

[SLD16] Evan Shelhamer, Jonathon Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1–1, 05 2016.

[SN14] Signe Schmidt and Kirsten Nørgaard. Bolus calculators. *Journal of Diabetes Science and Technology*, 8(5):1035–1041, 2014. PMID: 24876436.

[SQ17] Ehab Salahat and Murad Qasaimeh. Recent advances in features extraction and description algorithms: A comprehensive survey. 03 2017.

[SRE+09] C. E. Smart, K. Ross, J. A. Edge, C. E. Collins, K. Colyvas, and B. R. King. Children and adolescents on intensive insulin therapy maintain postprandial glycaemic control without precise carbohydrate counting. *Diabetic Medicine*, 26(3):279–285, 2009.

[SRE+10] C. E. Smart, K. Ross, J. A. Edge, B. R. King, P. McElduff, and C. E. Collins. Can children with type 1 diabetes and their caregivers estimate the carbohydrate content of meals and snacks? *Diabetic Medicine*, 27(3):348–353, 2010.

[ten19] tensorflow. Supported object detection evaluation protocols. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/evaluation_protocols.md, 2019. Accessed: November 2019.

[TKP16] Naveen Tokas, Shruti Karkra, and Manoj Kumar Pandey. Comparison of digital image segmentation techniques-a research review. 2016.

[TU17] P. Temdee and S. Uttama. Food recognition on smartphone using transfer learning of convolution neural network. In *2017 Global Wireless Summit (GWS)*, pages 132–135, Oct 2017.

[VAPS12] Gregorio Villalobos, Rana Almaghrabi, Parisa Pouladzadeh, and Shervin Shirmohammadi. An image processing approach for calorie intake measurement. 05 2012.

[Vid17] ComputerVisionFoundation Videos. Focal loss for dense object detection. https://www.youtube.com/watch?v=44tlnmmt3h0&t=212s, November 2017. Accessed: December 2019.

[Vit18] Angelo Vittorio. Toolkit to download and visualize single or multiple classes from the huge open images v4 dataset. https://github.com/EscVM/OIDv4_ToolKit, 2018. Accessed: October 2019.

[WWX+17] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William Freeman, and Joshua Tenenbaum. Marrnet: 3d shape reconstruction via 2.5d sketches. 11 2017.

[WZZ+18] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William Freeman, and Joshua Tenenbaum. Learning shape priors for single-view 3d completion and reconstruction, 09 2018.

[XHK+13] C. Xu, Y. He, N. Khanna, C. J. Boushey, and E. J. Delp. Model-based food volume estimation using 3d pose. In *2013 IEEE International Conference on Image Processing*, pages 2534–2538, Sep. 2013.

[Xu17] Joyce Xu. An intuitive guide to deep network architectures. https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41, 2017. Accessed: January 2020.

[You07] Abdou Youssef. Image downsampling and upsampling methods 1. 2007.

[ZBK+15] F. Zhu, M. Bosch, N. Khanna, C. J. Boushey, and E. J. Delp. Multiple hypotheses image segmentation and classification with application to dietary assessment. *IEEE Journal of Biomedical and Health Informatics*, 19(1):377–388, Jan 2015.

[ZBS+11] Fengqing Zhu, Marc Bosch, Tusarebecca Schap, Nitin Khanna, David Ebert, Carol Boushey, and Edward Delp. Segmentation assisted food classification for dietary assessment. *Proceedings of SPIE*, 7873:78730B, 01 2011.

[ZBW+10] Fengqing Zhu, Marc Bosch, Insoo Woo, SungYe Kim, Carol Boushey, David Ebert, and E.J. Delp. The use of mobile devices in aiding dietary assessment and evaluation. *Selected Topics in Signal Processing, IEEE Journal of*, 4:756 – 766, 09 2010.

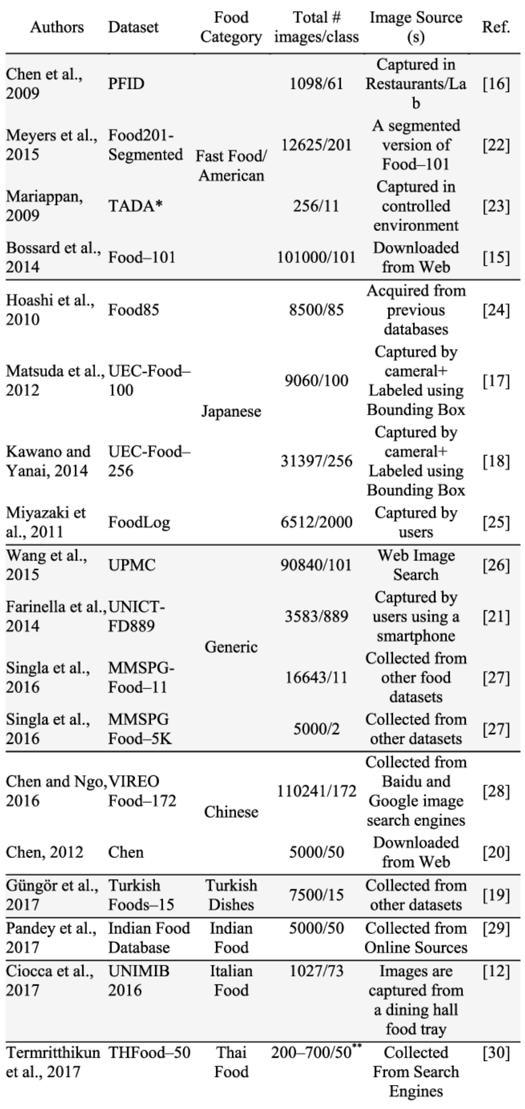
[Zea19] Food Standards Australia New Zealand. Food standards australia new zealand. https://www.foodstandards.gov.au/Pages/default.aspx, 2019. Accessed: December 2019.

[ZZG+15] W. Zhang, D. Zhao, W. Gong, Z. Li, Q. Lu, and S. Yang. Food image recognition with convolutional neural networks. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 690–693, Aug 2015.

[ZZZ+18] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Joshua Tenenbaum, William Freeman, and Jiajun Wu. Learning to reconstruct shapes from unseen classes, 12 2018.

Appendix A

Public Food Image Datasets

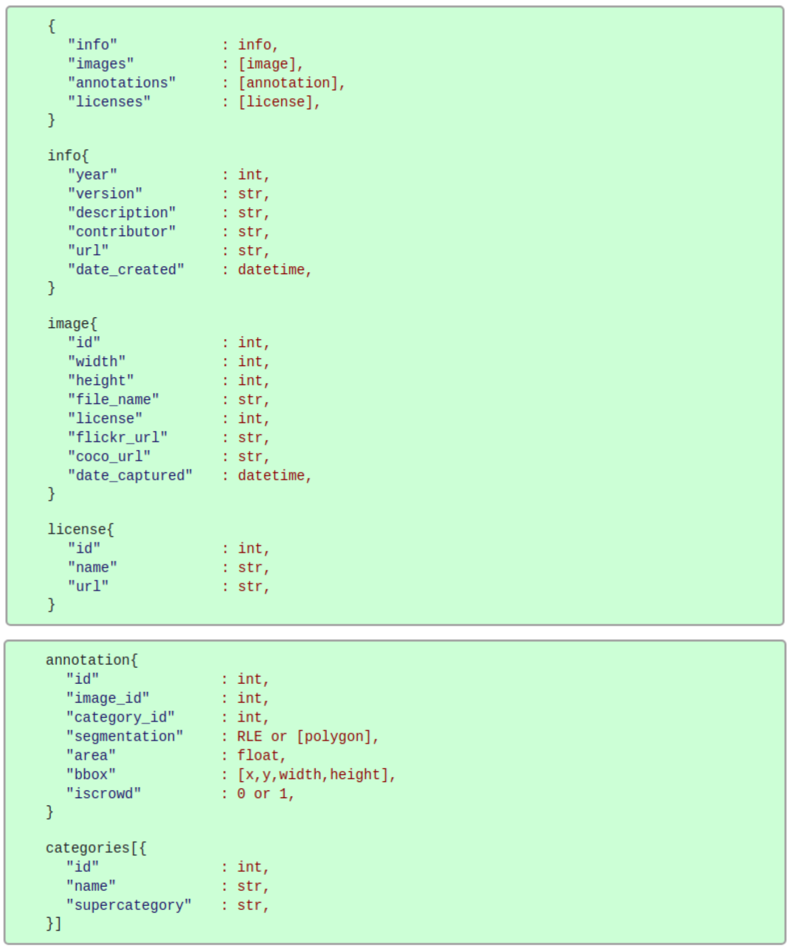


Authors	Dataset	Food Category	Total # images/class	Image Source (s)	Ref.
Chen et al., 2009	PFID		1098/61	Captured in Restaurants/Lab	[16]
Meyers et al., 2015	Food201-Segmented	Fast Food/ American	12625/201	A segmented version of Food–101	[22]
Mariappan, 2009	TADA*		256/11	Captured in controlled environment	[23]
Bossard et al., 2014	Food–101		101000/101	Downloaded from Web	[15]
Hoashi et al., 2010	Food85		8500/85	Acquired from previous databases	[24]
Matsuda et al., 2012	UEC-Food–100	Japanese	9060/100	Captured by cameral+ Labeled using Bounding Box	[17]
Kawano and Yanai, 2014	UEC-Food–256		31397/256	Captured by cameral+ Labeled using Bounding Box	[18]
Miyazaki et al., 2011	FoodLog		6512/2000	Captured by users	[25]
Wang et al., 2015	UPMC		90840/101	Web Image Search	[26]
Farinella et al., 2014	UNICT-FD889	Generic	3583/889	Captured by users using a smartphone	[21]
Singla et al., 2016	MMSPG-Food–11		16643/11	Collected from other food datasets	[27]
Singla et al., 2016	MMSPG Food–5K		5000/2	Collected from other datasets	[27]
Chen and Ngo, 2016	VIREO Food–172	Chinese	110241/172	Collected from Baidu and Google image search engines	[28]
Chen, 2012	Chen		5000/50	Downloaded from Web	[20]
Güngör et al., 2017	Turkish Foods–15	Turkish Dishes	7500/15	Collected from other datasets	[19]
Pandey et al., 2017	Indian Food Database	Indian Food	5000/50	Collected from Online Sources	[29]
Ciocca et al., 2017	UNIMIB 2016	Italian Food	1027/73	Images are captured from a dining hall food tray	[12]
Termritthikun et al., 2017	THFood–50	Thai Food	200–700/50**	Collected From Search Engines	[30]

Figure A.1: Public food image datasets [SAM19]

Appendix B

COCO Data Format



```
{
   "info"              : info,
   "images"            : [image],
   "annotations"       : [annotation],
   "licenses"          : [license],
}

info{
   "year"              : int,
   "version"           : str,
   "description"       : str,
   "contributor"       : str,
   "url"               : str,
   "date_created"      : datetime,
}

image{
   "id"                : int,
   "width"             : int,
   "height"            : int,
   "file_name"         : str,
   "license"           : int,
   "flickr_url"        : str,
   "coco_url"          : str,
   "date_captured"     : datetime,
}

license{
   "id"                : int,
   "name"              : str,
   "url"               : str,
}
```

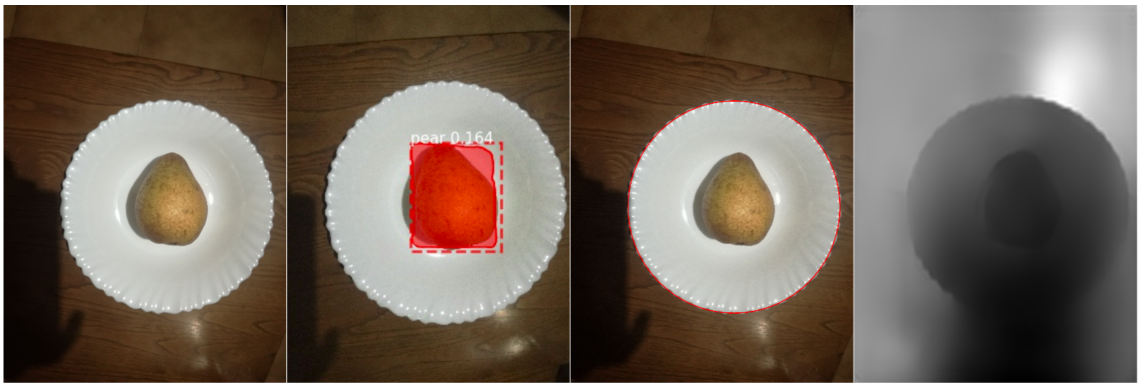
```
annotation{
   "id"                : int,
   "image_id"          : int,
   "category_id"       : int,
   "segmentation"      : RLE or [polygon],
   "area"              : float,
   "bbox"              : [x,y,width,height],
   "iscrowd"           : 0 or 1,
}

categories[{
   "id"                : int,
   "name"              : str,
   "supercategory"     : str,
}]
```

Figure B.1: COCO data format

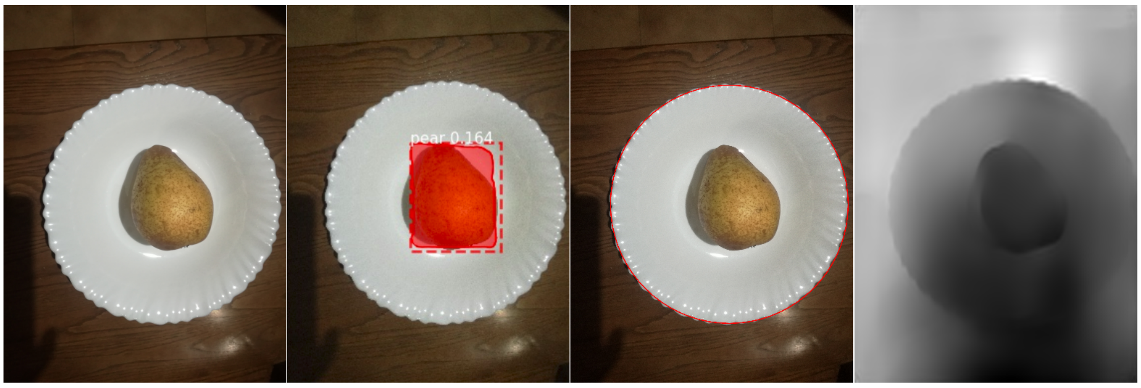
The first part of B.1 is common to all COCO types of annotations, however, the second part is specific for the task of object detection. The annotations used by Mask R-CNN follow this example.

Appendix C

Images used to estimate the volume

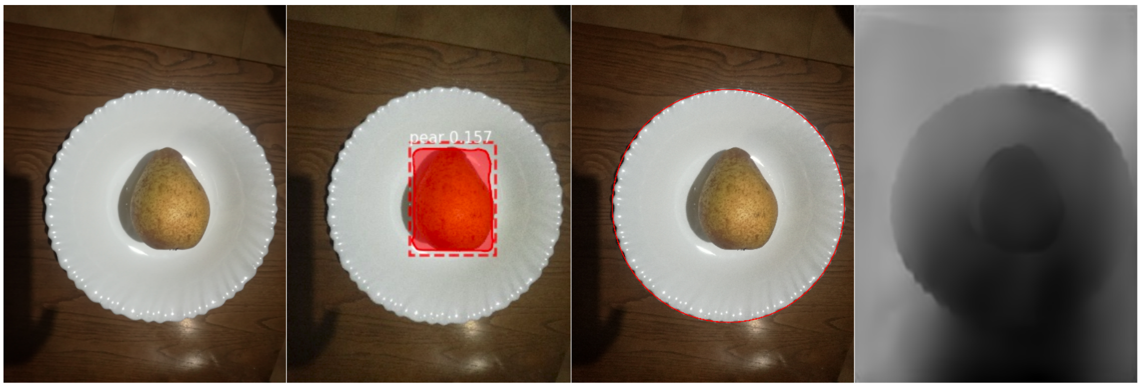


(a) 1.png

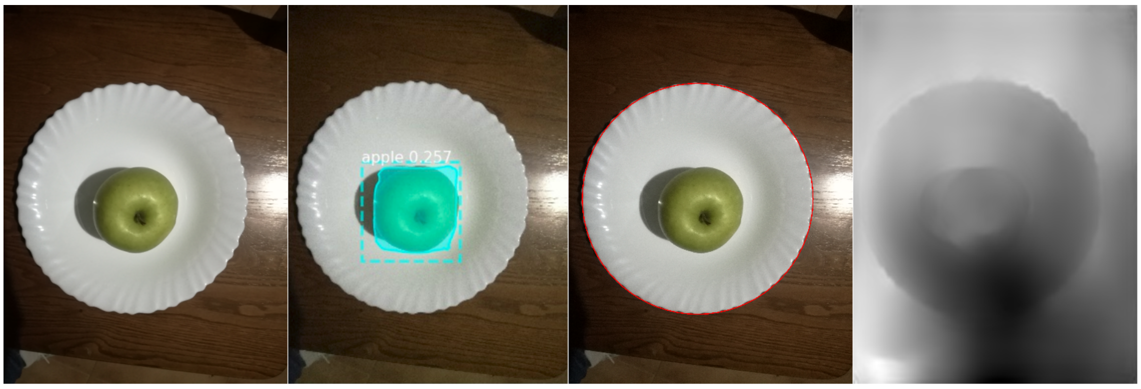


(b) 2.png

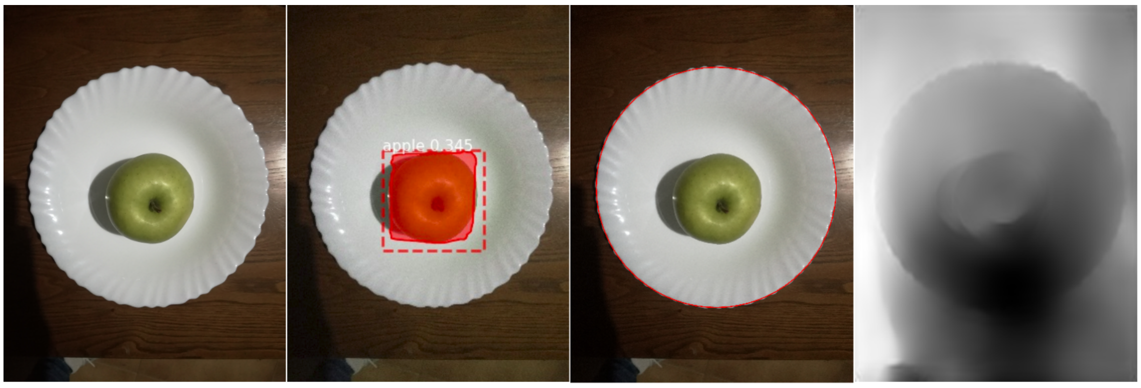
Images used to estimate the volume



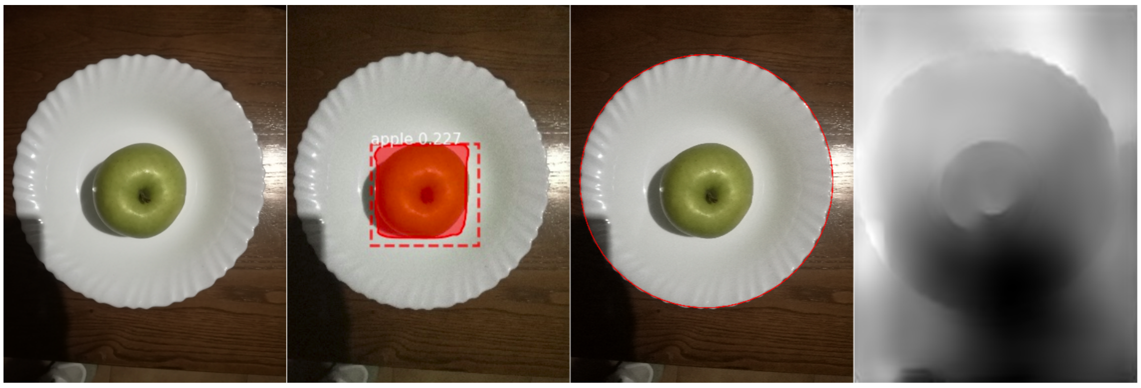
(c) 3.png



(d) 4.png

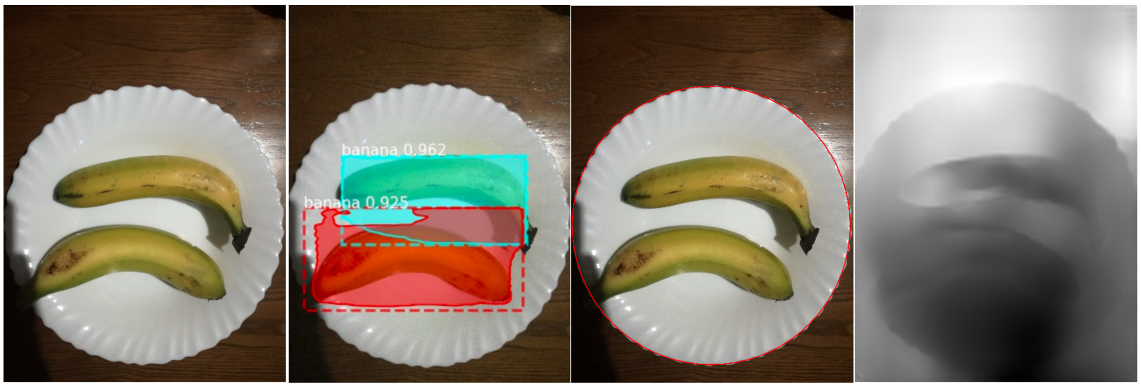


(e) 5.png

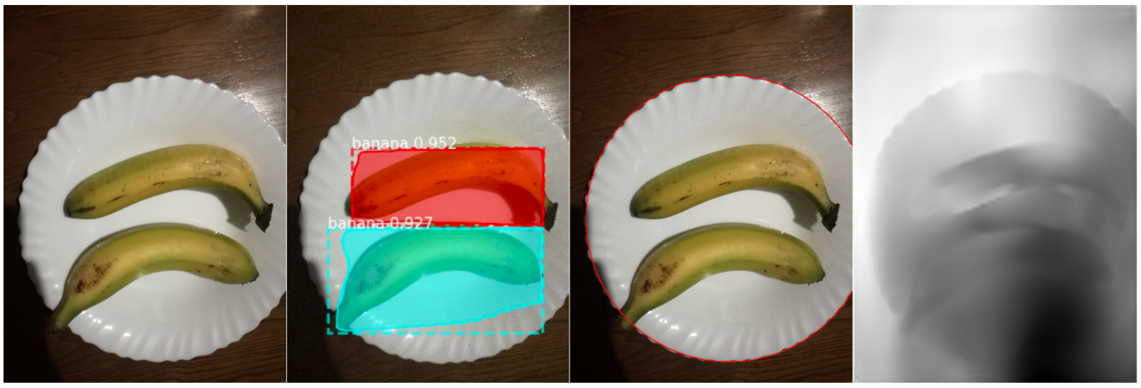


(f) 6.png

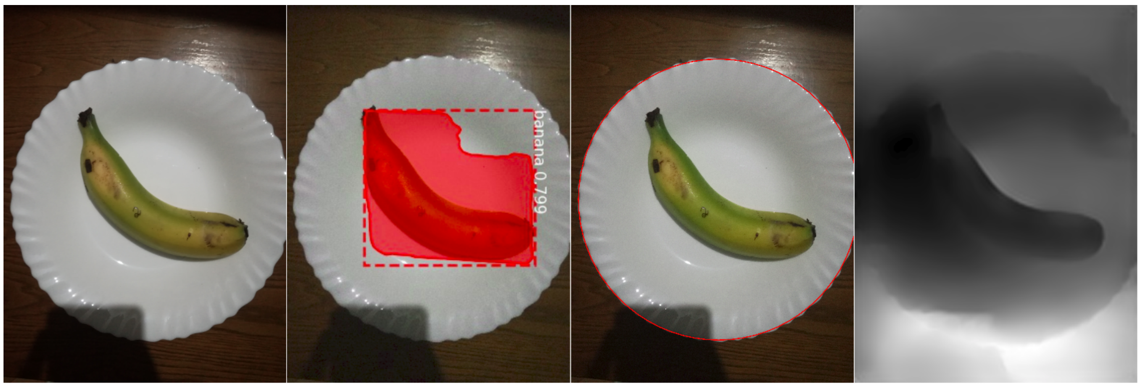
Images used to estimate the volume



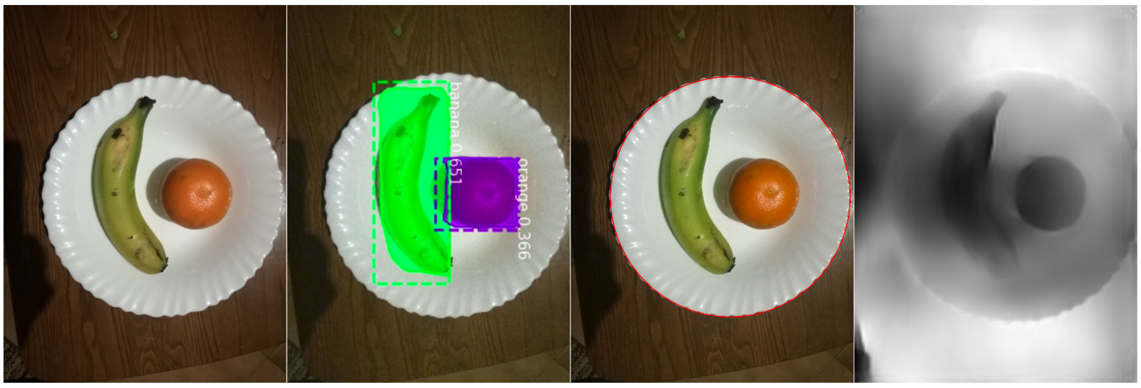
(g) 7.png



(h) 8.png

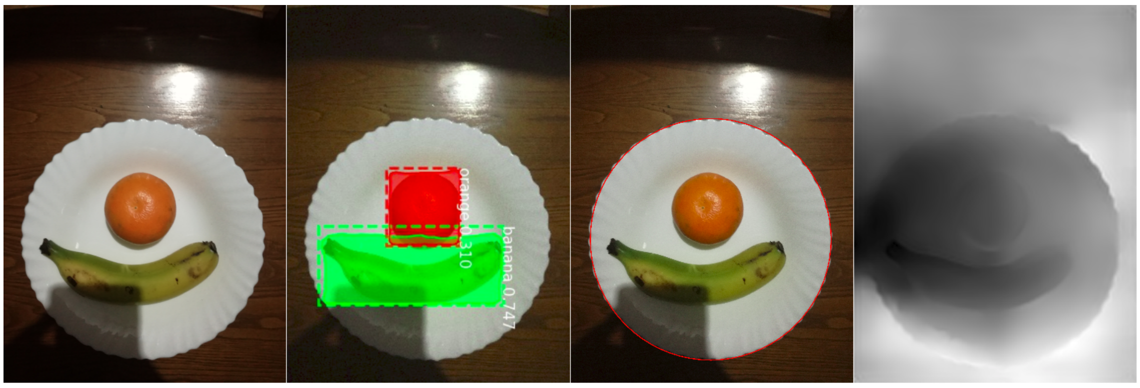


(i) 9.png

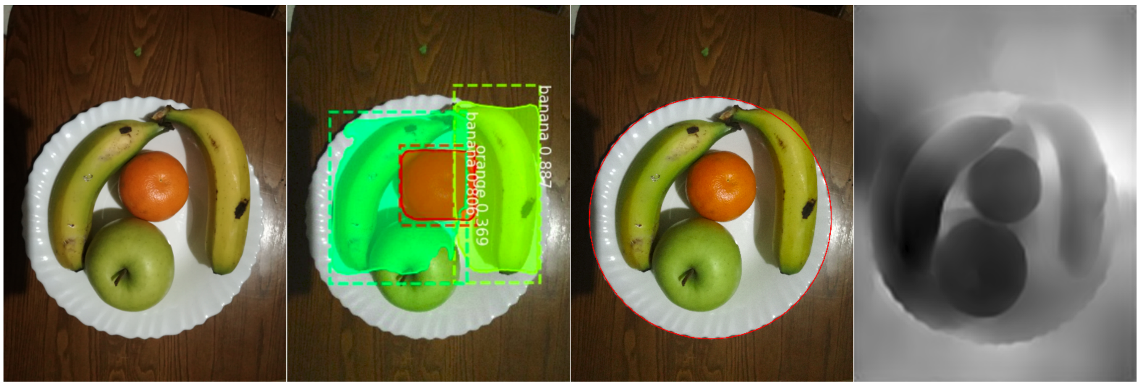


(j) 10.png

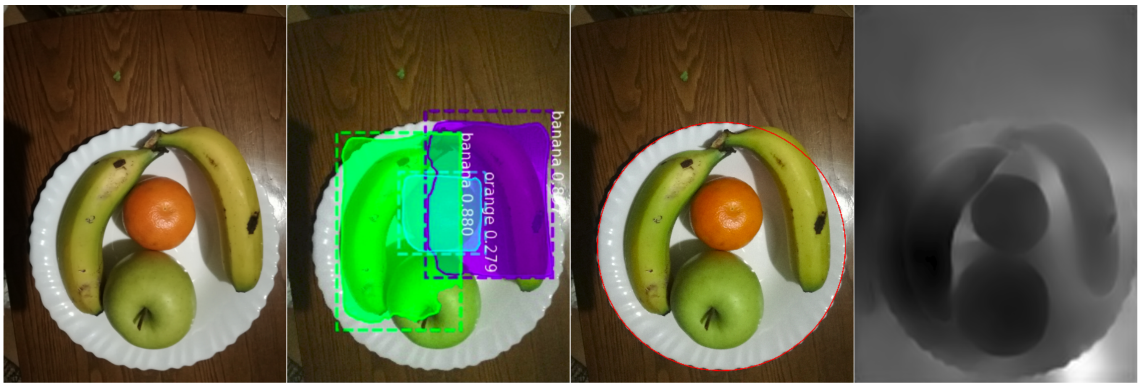
Images used to estimate the volume



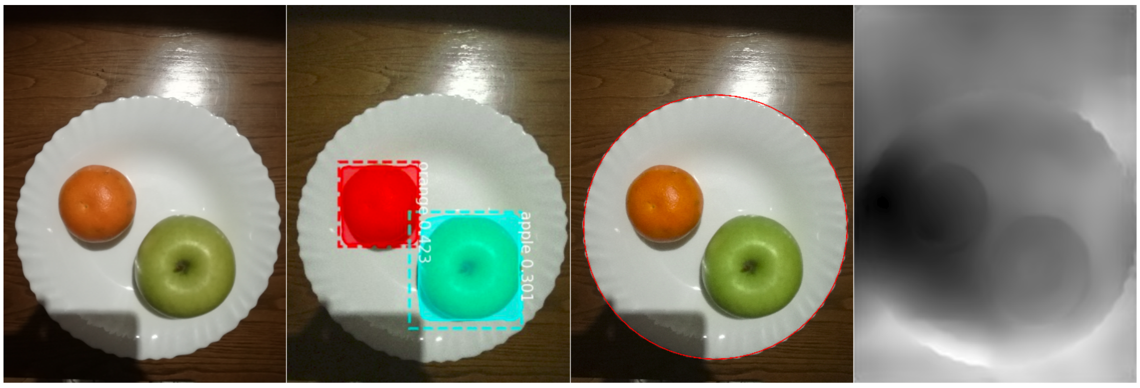
(k) 11.png



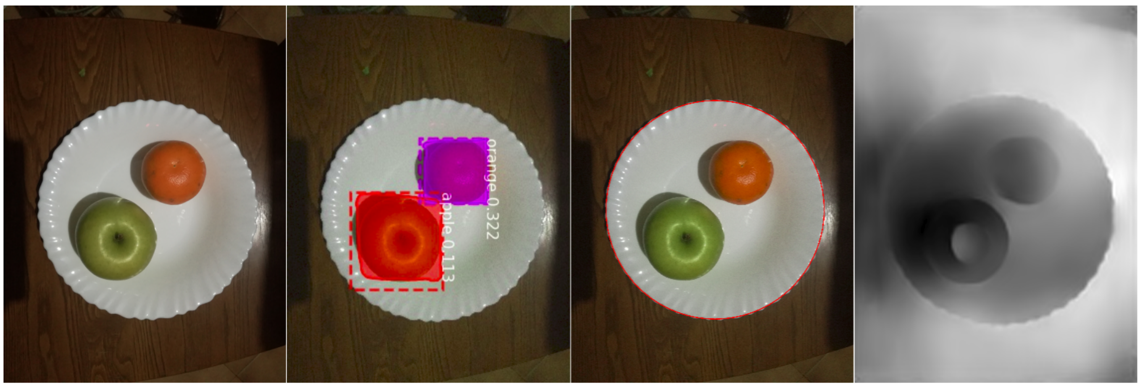
(l) 12.png



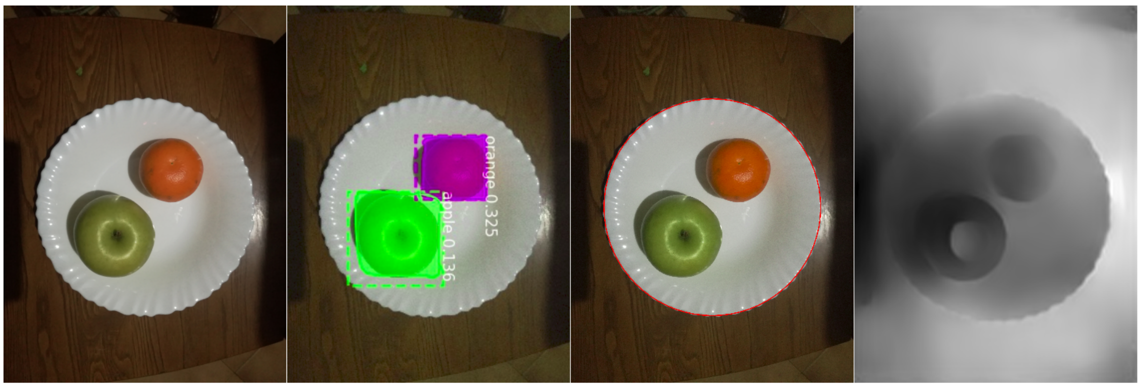
(m) 13.png



(n) 14.png



(o) 15.png

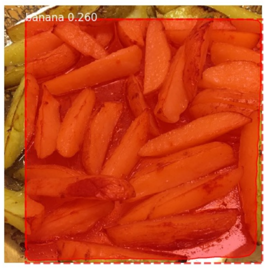


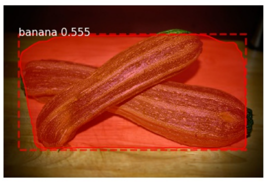
(p) 16.png

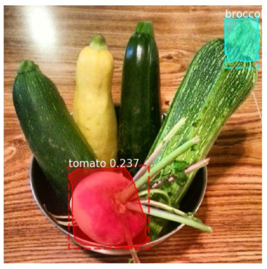
Figure C.1: Images used in the pipeline to estimate the volume

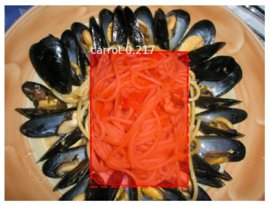
Appendix D

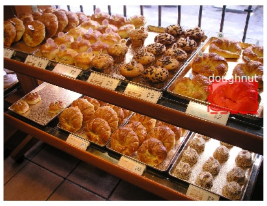
Misclassified images using the Mask R-CNN

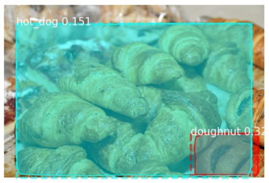


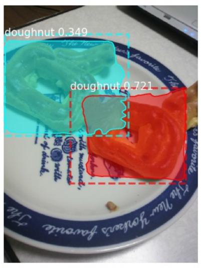


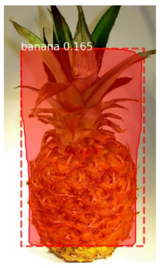


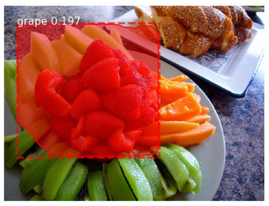












Misclassified images using the Mask R-CNN

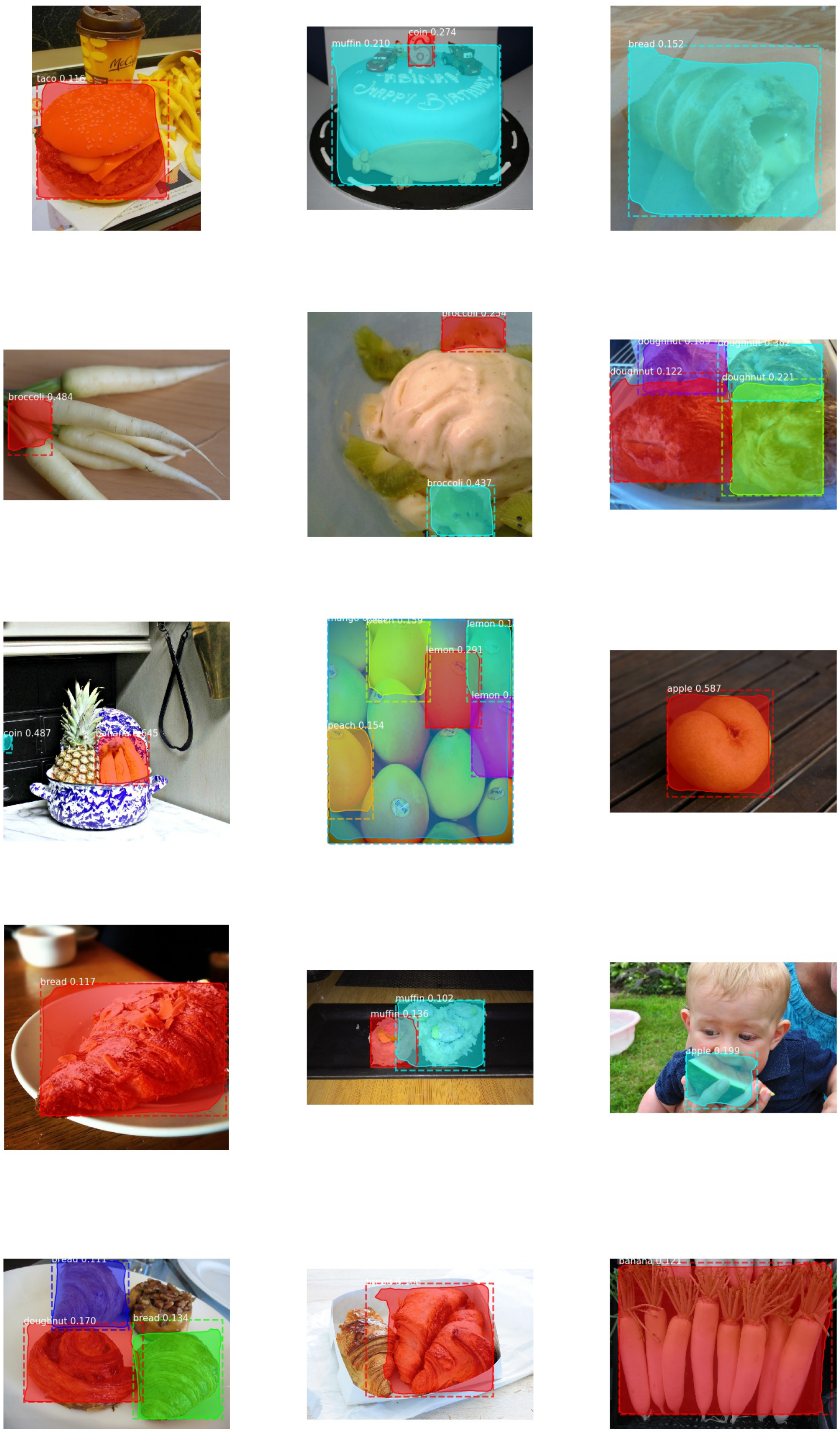
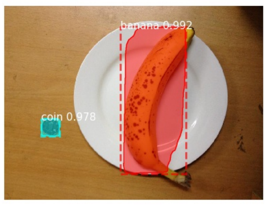
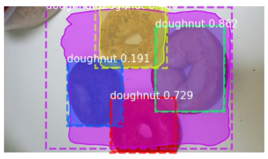


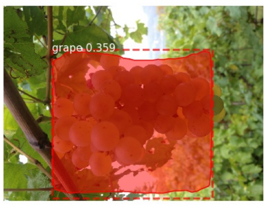
Figure D.0: Misclassified images using Mask R-CNN

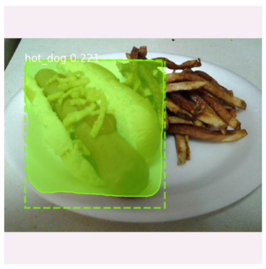
Appendix E

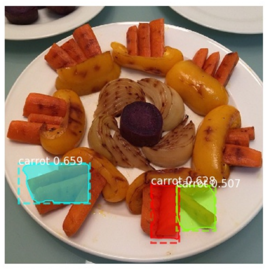
Well classified images using the Mask R-CNN

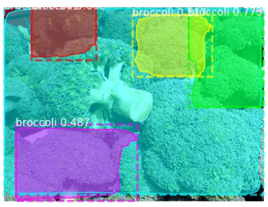












Well classified images using the Mask R-CNN

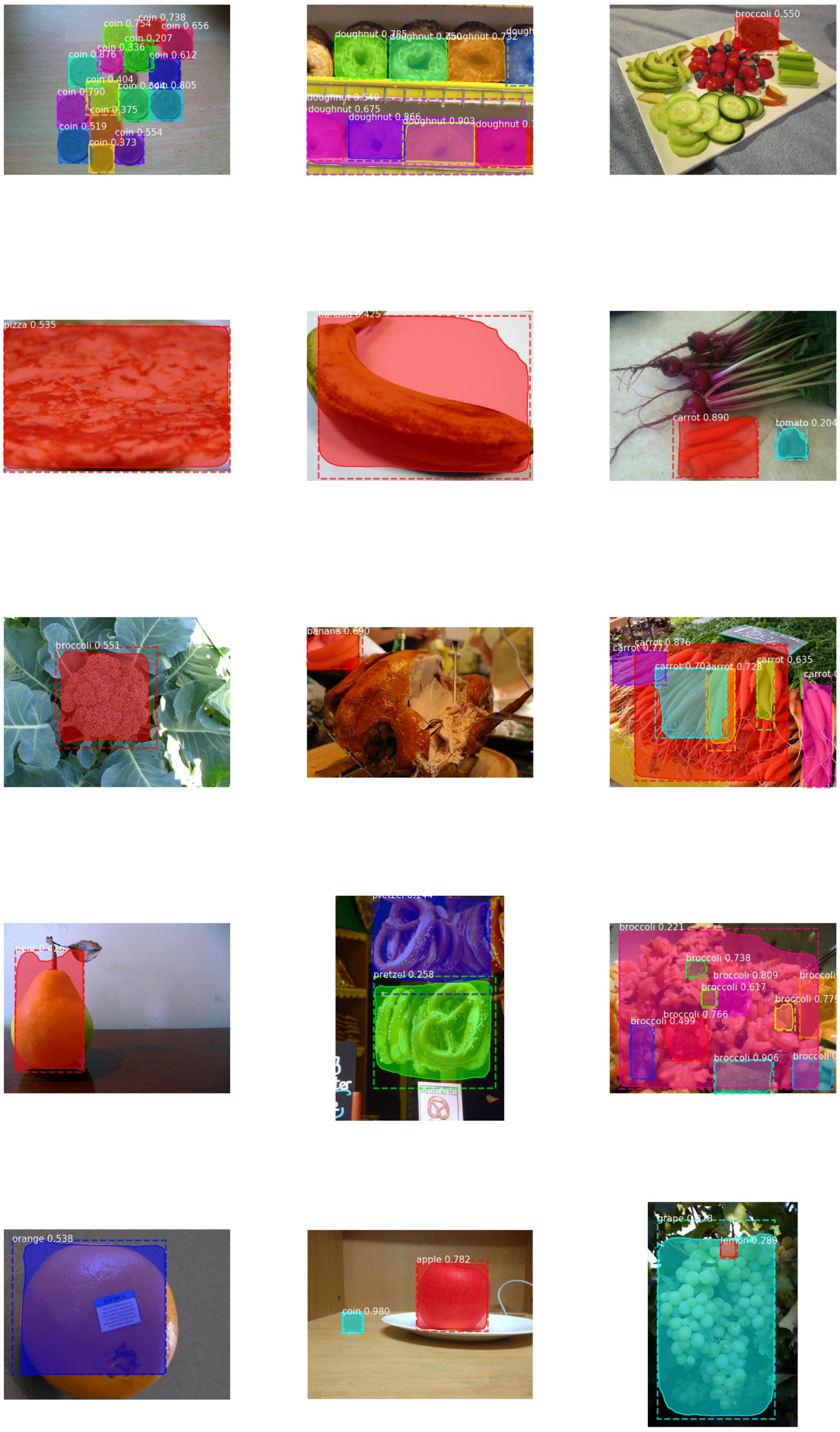


Figure E.0: Well classified images using Mask R-CNN