

**CRANFIELD UNIVERSITY**

**NOGARET BAPTISTE**

**AUTOMATED FOOD LOG ANALYSIS**

**SCHOOL OF AEROSPACE, TRANSPORT AND  
MANUFACTURING**

**Computational and Software Techniques in Engineering**

**Master of Science  
Academic Year: 2015–2016**

**Supervisor: Dr RÜGER Stefan  
August 6, 2016**



**CRANFIELD UNIVERSITY**

**SCHOOL OF AEROSPACE, TRANSPORT AND  
MANUFACTURING**

**Computational and Software Techniques in Engineering**

**Master of Science**

**Academic Year: 2015–2016**

**NOGARET BAPTISTE**

**Automated food log analysis**

**Supervisor: Dr RÜGER Stefan  
August 6, 2016**

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science.

© Cranfield University 2016. All rights reserved. No part of this publication may be reproduced without the written permission of the copyright owner.



# **Declaration of authorship**



# **Abstract**

Type your abstract here.

## **Keywords**

Keyword 1; keyword 2; keyword 3.



# Contents

<b>Declaration of authorship</b>	<b>v</b>
<b>Abstract</b>	<b>vii</b>
<b>Table of Contents</b>	<b>ix</b>
<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Abbreviations</b>	<b>xv</b>
<b>Acknowledgements</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Previous work</b>	<b>7</b>
<b>3 Feature descriptor</b>	<b>9</b>
3.1 Local binary pattern . . . . .	9
3.2 Color descriptor . . . . .	11
3.3 Bag-of-Words . . . . .	14
<b>4 Classifier</b>	<b>19</b>
4.1 Decision tree and random forest . . . . .	19
4.2 Support Vector Machine . . . . .	21
4.3 Convolutional neural network . . . . .	24
<b>5 Dataset</b>	<b>27</b>
5.1 Choice of the datatset . . . . .	27
5.2 UEC FOOD-100 and UEC FOOD-256 . . . . .	28

<b>6</b>	<b>Methodology</b>	<b>31</b>
6.1	Hyperparameter optimization . . . . .	31
6.2	Localisation . . . . .	32
6.3	Food recognition . . . . .	34
6.4	Code . . . . .	35
<b>7</b>	<b>Evaluation</b>	<b>37</b>
7.1	Environment . . . . .	37
7.2	Segmentation metrics . . . . .	38
7.3	Cross validation . . . . .	39
7.4	Results . . . . .	40
<b>8</b>	<b>Future work</b>	<b>45</b>
<b>A</b>	<b>Appendix</b>	<b>47</b>
A.1	RGB to HSV . . . . .	47
A.2	HSV to RGB . . . . .	48

# List of Figures

1.1	Obesity and overweight rate of the adult population in the uk between 1980 and 2030. <i>Source: World Health Organisation</i> . . . . .	1
1.2	Average classification and localization error of the best results for different ImageNet challenges . . . . .	3
1.3	Examples of high intra-class variability for kaya toast . . . . .	5
1.4	Examples of low inter-class variability for kaya toast . . . . .	5
3.1	Illustration of the LBP descriptor's process . . . . .	10
3.2	Illustration of the Bag-Of-Visual-Words model . . . . .	15
3.3	Illustration of SIFT as a local image descriptor . . . . .	16
4.1	Decision tree of for ten elements belonging to two classes . . . . .	20
4.2	A regular 3-layer neural network . . . . .	24
4.3	Example of a 16-layer deep convolutional neural network . . . . .	25
4.4	Illustration of a max pooling layer of stride 2 . . . . .	26
5.1	Pictures with multiple food items from UEC FOOD 256 . . . . .	30
6.1	General process of the localisation and classification . . . . .	32
6.2	Picture of the 100 possible bounding boxes that the salient CNN will try to recognise . . . . .	33
7.1	Illustration of 4-fold cross validation . . . . .	39
7.2	Curves of Accuracy over IoU (top), Precision over IoU (centre) and Recall over IoU (bottom) . . . . .	41
7.3	Classes having the highest accuracy . . . . .	43
7.4	Classes having the lowest accuracy . . . . .	43
7.5	Most confused classes . . . . .	44



# List of Tables

5.1	Summary of some available food datasets according to the criteria . . . . .	28
7.1	Average localisation accuracy result for UEC FOOD 256 . . . . .	40
7.2	Average classification accuracy result for UEC FOOD 256 . . . . .	42
7.3	Average accuracy result for UEC FOOD 256 . . . . .	42
7.4	Average accuracy result for UEC FOOD 100 . . . . .	43



# List of Abbreviations

BoW	Bag of Words
CNN	Convolutional Neural Network
LBP	Local Binary Pattern
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
SVM	Support Vector Machine



# Acknowledgements

I am really grateful to Dr. Stefan Rüger, my supervisor for the project, to have proposed this subject. His guidance and valuable advice were particularly helpful to realise the thesis.

Moreover, I would like to thank the University of Technology of Compiègne for giving me the opportunity to study one year in Cranfield University. I would also like to thank Cranfield University for its facilities.

I would like to express my gratitude to M. Kazu Shimoda and Pr. Keiji Yanai of the University of Tokyo that made their datasets available and provided enlightenments and further details on their work.



# Chapter 1

## Introduction

Over the last few decades, the rate of obesity and overweight people in the World has greatly increased. As presented for the UK case in the figure 1.1, the obesity rate has increased by 12 % between 1980 and 2013, and by 13 % for the overweight rate. It is forecasted by the World Health Organisation to continue to grow.

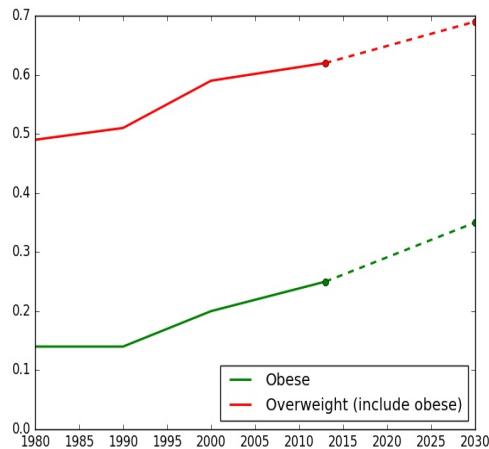


Figure 1.1: Obesity and overweight rate of the adult population in the uk between 1980 and 2030

Being “overweight” is defined as having a Body Mass Index (BMI) – a person’s weight

in kilograms divided by the square of his height in meters ( $kg/m^2$ ) – of between 25 and 29.9, and “obese” by a BMI of 30 and above.

As stated in [1]), obesity is strongly associated with several major health risk factors such as stroke, high blood pressure, type 2 diabetes and high cholesterol. Thus, it has a great human and economic ([2] in 2010, 12 % of the total worldwide health expenditure is spent on diabetes and will continue to increase) cost for societies.

Associated with lifestyle changes, recording what we eat is one way to control our eating. Studies such as [3] show the benefit of reporting its daily diet to lose weight and improve the quality of its food intake. And more generally, it can be a way to treat eat disorders

Yet, manually recording detailed information regarding all meals is a tedious and time consuming task and it is hard for people to adhere to this process for a long time. Moreover, it often needs a trained patient and as presented in [4], users’ log are prone to errors (users tend to underestimate its intake).

At the same time, classification methods improvement of the classification methods. Imagenet is dataset containing more than 1,2 million images dataset split into 1000 classes. Since 2010, the yearly challenges include localization, classification and detection. Numerous researchers, students, educators or tech companies are participated.

As described in figure 1.2 and using figures from [5], the mean error for each class for classification and localization has been greatly reduced between 2010 and 2014.

With the widespread use of smartphone, cameras or wearable devices, people can easily take pictures of a good quality and are already taking photos of their food and posting them on website such as Food Gawker, Instagram, Flickr or Yelp.

That’s why, it has recently been proposed to automate it and assist patient and their medical personnel. It extends the reach of care in a cost effective ways and counters some of the previous problem of manual report. It’s part of the rise of e-healthcare /

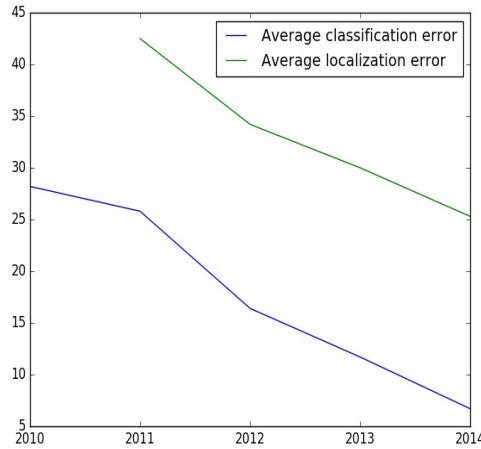


Figure 1.2: Average classification and localization error of the best results for different ImageNet challenges

m-healthcare [6, 7].

Users upload photographs of their daily meals to the application or website, and it constructs their food diary automatically. Using image processing, it determines the dietary composition of the meal and records the information for later viewing in formats such as a figure or a graphical representation.

Food recognition is a promising applications of image processing and machine learning. Its overall process is:

- Extract key characteristics
- Localise food items if the application allow it
- Recognise the food

extract characteristic (...)segmentate, classify, get calorie value or a simplified version (using for example the ... systems), keep log and being able to visualize it over the year

Feature description is essential to achieve good object detection and image categorisation. Preferably the method should be invariant of the luminosity, orientation or scale of

the picture.

In this thesis, we focus on the food recognition. It has already numerous challenges such as:

- **high intra-class variability** : we can have high variability between pictures for the same particular kind of food items, due to:

- the environmental condition: luminosity
- the way it is served
- variation of the way the picture is taken: numerous transformation can be applied to a same picture (scale, translation, rotation, skewness)

This is illustrated in figure 1.3 for pictures of laya toast.

- **low inter-class variability** : we can have low variability between different type of food such as between clear and miso soup as showed in 1.4.

The organization of this thesis is as follow. In section 2, previous work on food recognition and intake estimation is reviewed. In section 3, I describe the different image descriptors used, the classifiers in section 4 and in section 5. In the next section 7, we present and discuss our results. Finally, in section 8, the limitation and possible future work is discussed.



Figure 1.3: Examples of high intra-class variability for kaya toast. Pictures extracted from the UEC FOOD 256 dataset.



Figure 1.4: Examples of low inter-class variability for clear soup (left) and miso soup (right). Pictures extracted from the UEC FOOD 256 dataset.



# Chapter 2

## Previous work

Food localization process

circle detection if we make the assumption that the food is in a plate / bowl: 2.2 + 6.3  
+ [8]

color segmentation vs edge segmentation: 11.1 (very limited test)

DCNN: 3.2 + \* + 9.1

Food recognition

Using SVM:

Local using BOW: 3.3 global feature: Color and texture description: Spatial pyramid:

Mix of several features:

DCNN: \*

Estimate food intake:

Food log Food Cam



# **Chapter 3**

## **Feature descriptor**

For a computer, a picture is represented as a 2-D or 3-D array.

The color is one of the key components of a food item, thus it is widely applied for classification. Color statistics are commonly used, such as the first and second moment values for different channels. It can be computed for multiple color representations (RGB, HSV, gray, YCrCb or L\*a\*b\* space).

Another import feature of the food is the texture. Numerous texture feature can be used such as Gabor filters. As already presented, the local binary pattern is also applied.

### **3.1 Local binary pattern**

Local binary pattern is a visual descriptor for texture composition of an image, first presented in 2002 in [9] (although the concept of LBPs were introduced as early as 1993).

#### **3.1.1 Gray-scale LBP**

The figure 3.1 represents an example of the LBP in which the LBP code of the center pixel (in red color and value 20) is used as a local intensity threshold : the neighbour

pixels whose intensities are equal or higher than the center pixel's are labeled as "1"; otherwise as "0". Then, starting always from the same point, we can transform this binary string to decimal and is used to describe the central pixel. In this example we start at the top-right point and work our way clockwise accumulating the binary string as we go along and obtain the value 24.

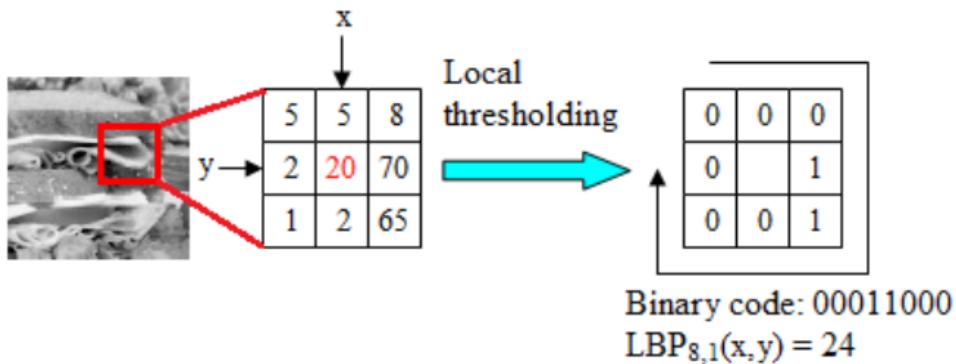


Figure 3.1: Illustration of the LBP descriptor's process

We adopt the following notation. Given a pixel  $c = (x_c, y_c)$ , the value of the *LBP* code of  $c$  is defined as:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p$$

where:

- $p$  is a neighbour pixel of  $c$  and the distance from  $p$  to  $c$  does not exceed  $R$ . Thus,  $R$  is the radius of a circle centered in  $c$  and  $P$  is the numbered of sampled points.
- $g_p$  and  $g_c$  are the gray values (intensities) of  $p$  and  $c$

- $s(x)$  is the function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

In Fig. 3.1,  $R$  and  $P$  are 1 and 8 respectively.

The number of histograms bins for  $LBP_{P,R}$  is  $2^P$ .

### 3.1.2 Uniform LBP

This algorithm has been enhanced to make it rotation invariant. Still in [9], the authors introduce the notion of uniform LBP. A LBP is considered to be uniform if it has at most two bitwise transitions (0 to 1 or 1 to 0 transitions in the binary word).

For example, the pattern *01000000* (2 transitions) and *11111110* (1 transition) are both considered to be uniform. For a  $LBP_{P,R}$ , there is  $p + 1$  possible uniforms.

Non-uniform LBP are considered as noise and are assigned the same constant value.

Thus, for uniform LBP, we use the formula:

$$LBP_{P,R}^{uni}(x_c, y_c) = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) 2^p & \text{if uniform} \\ P + 1 & \text{otherwise} \end{cases}$$

## 3.2 Color descriptor

### 3.2.1 Color histogram

HSV space is composed of:

- **Hue channel:** represents the dominant spectral component—color in its pure form,

as in green, red, or yellow

- **Saturation** channel: represents the white added to the pure color (the Hue)
- **Value** channel: represents the brightness of the color

Hue and Saturation corresponds to the chromaticity of the colour. For the joint histogram (2D histogram), the H and S channels are used as value is dependant of the condition where the picture were taken, thus is not interesting.

The coordinate system is cylindrical, and is often represented by a subspace defined by a six-sided inverted pyramid. The top of the pyramid corresponds to, with the “white” at the center. The hue is measured by the angle around the vertical axis, with red corresponding to 0. The saturation ranges from 0 at the center to 1 on the surface of the pyramid. An inverted cone is also used to denote the subspace instead of the pyramid.

### 3.2.2 Color moments

### 3.2.3 The first two moments

For a discrete random variable  $X$ , the first two moments are defined as:

- **Expected value:**

$$\mathbb{E}[X] = \mu = \sum_{i=1}^n p_i x_i$$

- **Variance:**

$$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \sum_{i=1}^n p_i(x_i - \mu)^2$$

### 3.2.4 Hu moments

#### Raw moments

For a two-dimensional continuous function  $f(x,y)$  the moment (sometimes called “raw moment”) of  $(p + q)$ th order is defined as:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y) dx dy$$

for  $p$  and  $q \in \mathbb{N}$ .

#### Central moments

And the central moments are :

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x,y) dx dy$$

with  $\bar{x} = \frac{M_{10}}{M_{00}}$  and  $\bar{y} = \frac{M_{01}}{M_{00}}$

#### Normalized central moments

The normalized central moments are:

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\gamma}}$$

where  $\gamma = 1 + \frac{i+j}{2}$  for  $i + j \geq 2$ .

### Definition of the Hu moments

On the base of those Moments, Hu in [10] introduced 7 Moments which are invariant for translation, rotation and resizing:

$$I_1 = \eta_{20} + \eta_{02}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]$$

$$+ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]$$

$$- (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

## 3.3 Bag-of-Words

### 3.3.1 Process

**Bag-of-Words** *BoW*, also called Bag of features, is a feature descriptor method inspired by information retrieval from textual documents.

As illustrated in Fig. 3.2, the main steps are:

- On each picture, keypoints are detected. In my case, I use a dense grid of evenly spaced points at a fixed scale and orientation.

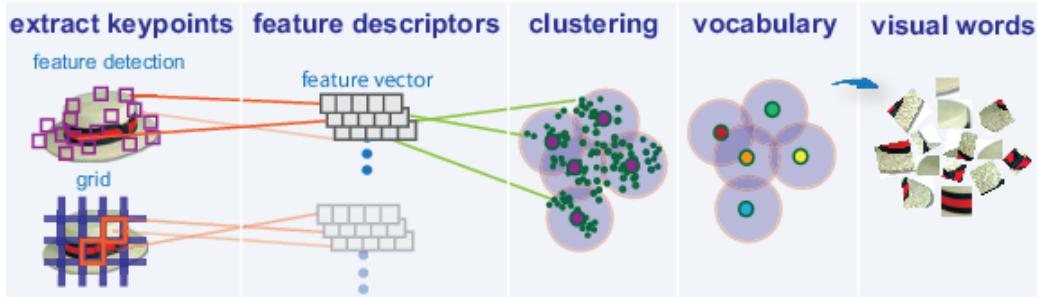


Figure 3.2: Illustration of the Bag-Of-Visual-Words model

- For every keypoint, a feature vector is generated. we describe it, SIFT (scale invariant feature transform).
- We generate the fix number of visual words that compose our codebook.
- We express each image as an histogram of these words' appearance.

The combination of a dense grid and SIFT is commonly called dense SIFT. It has been showed to have greater accuracy than using SIFT for keypoint detection and description.

### 3.3.2 SIFT descriptor

A SIFT descriptor of a local region (keypoint) is a 3-D spatial histogram of the image gradients as presented in the figure 3.3. The gradient at each pixel is regarded as a sample of a three-dimensional elementary feature vector, formed by the pixel location and the gradient orientation. Samples are weighed by the gradient norm and accumulated in a 3-D histogram  $h$ , which (up to normalization and clamping) forms the SIFT descriptor of the region. An additional Gaussian weighting function is applied to give less importance to gradients farther away from the keypoint center.

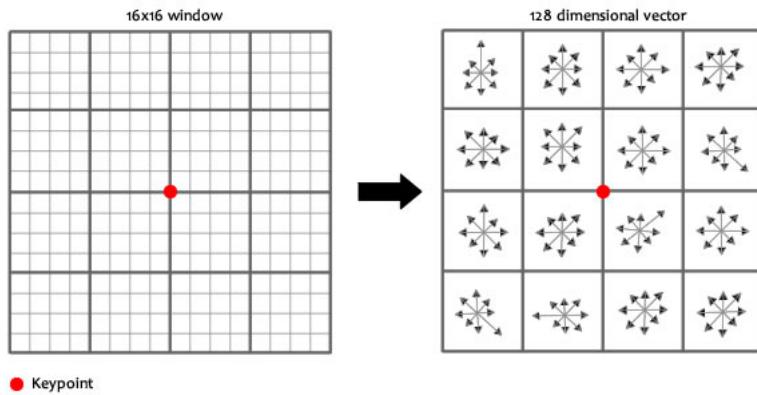


Figure 3.3: Illustration of SIFT as a local image descriptor

### 3.3.3 K-mean clustering

Given a set of observations  $(x_1, x_2, \dots, x_n)$ , where each observation is a d-dimensional real vector, k-means clustering aims to partition the  $n$  observations into  $k$  ( $k \leq n$ ) sets  $S = S_1, S_2, \dots, S_k$  so as to minimize the within-cluster sum of squares (sum of distance functions of each point in the cluster to the K center). In other words, its objective is to find:

$$\arg \min_S \sum_{i=1}^k \sum_{x \in S_i} \|x - \mu_i\|^2 \quad (3.2)$$

Problem, the exact solution is a NP hard problem. That's why, we can use Lloyd's heuristic algorithm to compute an estimation.

It is an iterative method that find a local minima of the Eq. 3.2:

1. A set of  $k$  initial “means” is chosen randomly within the data domain  $M = \{m_1, m_2, \dots, m_k\}$
2. Then,  $k$  clusters are created by associating every observation with the nearest mean.

$$\forall i \in \{1, \dots, k\}, \quad S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2 \quad \forall j \in \{1, \dots, k\}\}$$

3. The centroid of each of the  $k$  clusters becomes the new mean.

$$\forall i \in \{1, \dots, k\}, m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

4. Repeats step 2 and 3 until  $M$  not longer changes.

The centroid results and number of iterations are highly dependant of the initial centroid.

As a result, the computation is often done several times, with different initializations of the centroids. One method to help address this issue is the k-means++ initialization scheme, which has been described in [11]. This initializes the centroids to be (generally) distant from each other, leading to provably better results than random initialization.



# Chapter 4

## Classifier

k-nearest neighborhood

Naive bayesian

SGD classifier + loss function + regularization term

### 4.1 Decision tree and random forest

Decision tree is a simple learning method that can be used for classification or regression. The implementation used of decision tree is based on the CART (Classification and Regression Tree) algorithm.

A decision tree is recursively partitioning the space in a left  $P_{left}$  and right  $P_{right}$  partitions such that the samples with the same labels are grouped together, i.e. the generated sets with the smallest impurity.

It continues to split until the impurity can't be reduced or some pre-set stopping rules are met. Alternatively, the data are split as much as possible and then the tree is later pruned.

Since the set of splitting rules used to segment the predictor space can be summarized

in a tree, these types of approaches are known as decision tree methods. The figure 4.1 illustrate a toy example of decision tree.

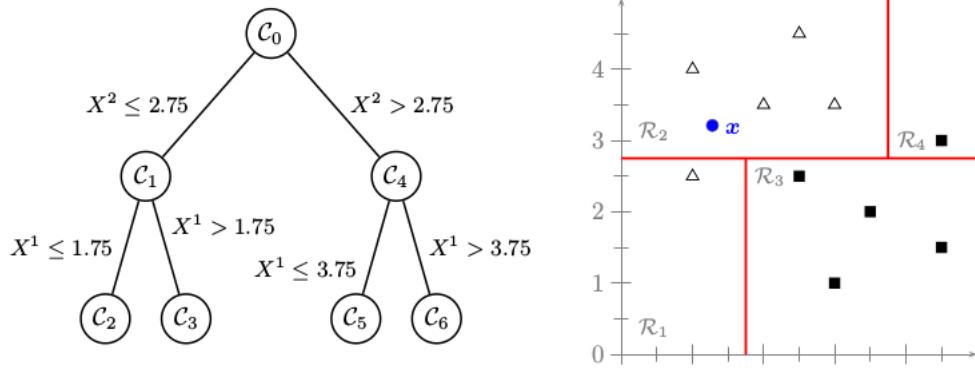


Figure 4.1: Decision tree of depth two for ten elements  $(X^1, X^2)$  belonging to the black square and white triangle classes

The most used impurity measure's functions are:

- **Gini:**

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk})$$

- **Cross-entropy:**

$$H(X_m) = - \sum_k p_{mk} \log(p_{mk})$$

To avoid overfitting, keep the decision tree as simple as possible.

**Random forest** or Decision forest is build from a number of decision trees. The prediction of the ensemble is given as the averaged prediction of the individual classifiers. Each tree is trained on a random subsets of the training data.

When building these decision trees, each time a split in a tree is considered, a random sample of  $m$  predictors is chosen as split candidates from the full set of  $n$  features. A typical value of  $m$  is  $m \approx \sqrt{n}$ .

## 4.2 Support Vector Machine

(binary case) + kernel trick + multi-class (one-versus-one or one-versus-all)

**Support Vector Machine SVM** is a method used for classification and regression.

### 4.2.1 Linear SVM

#### Hard margin

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

For a 2 classes (value represented as  $-1$  and  $1$ ), the hyperplane must verify:

$$\vec{x}_i \cdot \vec{w} + b \geq +1 \text{ for } y_i = +1 \quad (4.1)$$

$$\vec{x}_i \cdot \vec{w} + b \leq -1 \text{ for } y_i = -1 \quad (4.2)$$

where  $\vec{w}$  is the normal to the hyperplane

Combining equation 4.1 and 4.2, we obtain:

$$\forall i \in 0, \dots, n, \quad y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0$$

where  $y_i = f(\vec{x}_i) = -1, 1$

Gemetrically, the distance between the two hyperplane from 4.1 and 4.2 is  $\frac{2}{\|\vec{w}\|}$  (equal width to each side).

Thus, to obtain the hyperplane with the highest margin, we want to maximize:

$$\arg \max_{\vec{w}, b} \frac{2}{\|\vec{w}\|^2}$$

which is equivalent to minimize:

$$\arg \min_{\vec{w}, b} \frac{1}{2} \|\vec{w}\|^2$$

Thus, we obtain a constrained optimization problem.

## Soft Margin

For the case of non-separable training sets, we introduce a penalty parameter  $C$ ,  $C \leq 0$  and obtain:

$$\arg \min_{\vec{w}, b, \zeta} \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^n \zeta_i \text{ subject to } y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1 - \zeta_i, \zeta_i \geq 0, \forall i \in [1, \dots, n]$$

The decision function for new example is:

$$f(\vec{x}) = \text{sign} \left( \sum_{s_i \in \text{support vectors}} w_i \vec{s}_i \cdot \vec{x} + b \right)$$

where the support vectors selected sub-set of the training examples that define the boundary of the hyperplane separation and hence the classification boundary.

To generalize SVM to the case of multi-class, multiple approaches are possible:

- “one-versus-one”: train a separate classifier for each different pair of labels. This leads to  $\frac{N(N-1)}{2}$  classifiers
- “one-versus-all”: train a single classifier per class, with the samples of that class as

positive samples and all other samples as negatives

### 4.2.2 Non-linear SVM and kernel trick

The idea of the kernel trick is to transform the initial space to a higher dimensional space where a hyperplane can separate this data. Kernel trick: use kernel function to implicitly transform datasets to a higher-dimensional using no extra memory, and with a minimal effect on computation time: realise just a dot product.

To use the linear SVM for non-linear data: project the data in a new feature H space thanks to an application and then research for maximum margin hyperplane in H to make sure that the new problem has a unique solution, must satisfy the Mercer's condition or simply it must be a positive-definite matrix

- **Linear** :  $k(x, y) = \langle \vec{x}, \vec{y} \rangle + C = x^T y + C$
- **Polynomial**:  $k(x, y) = (\gamma \cdot \langle \vec{x}, \vec{y} \rangle + C)^d = (\gamma \times x^T y + C)^d$
- **Radial Basis Function (RBF)**:  $k(x, y) = \exp(-\gamma \|x - y\|^2)$
- **Chi-Square**:  $k(x, y) = 1 - \sum_{i=1}^n \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)}$

A modified version presented in [12] of this kernel is the **Additive Chi-Square**

$$\text{kernel} : k(x, y) = \sum_{i=1}^n \frac{2(x_i - y_i)}{x_i + y_i}$$

The adjustable parameters of these kernels are  $d, \gamma, C$  and must be chosen according to the problem.

For food classification, the chi square kernel is the most used kernel as it is often combined with histograms. !!CITE!!

### 4.3 Convolutional neural network

A **Convolutional Neural Network CNN** is a variant of a Neural Network, mainly used for machine learning on pictures. It is inspired by the neural system composed of different layers (made up of multiple neurons) and communication schemes.

Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity function. The whole network still expresses a single differentiable score function (linear or not): from the raw image pixels (the input layer) to class scores (output layer). Hidden layers separates these two layers, as described in 4.2.

A CNN (and more generally a NN) is trained by backpropagation, applying gradient descent that will update the weights.

It is a powerful, adaptive and noise resilient pattern recognition. The training phase is rather slow but querying it with an unseen example is fairly fast.

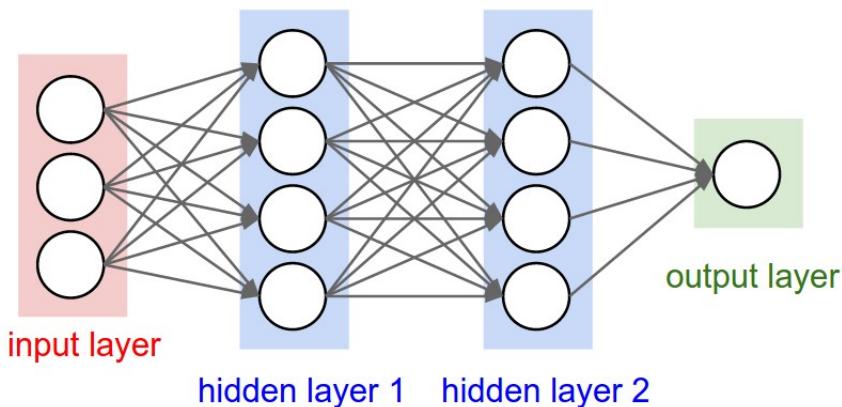


Figure 4.2: A regular 3-layer neural network

The figure 4.3 is a simple CNN based on the VGG-NET structure. It is composed of the 4 most popular layers that can be found in a CNN:

- **Convolutional** : layer giving the name for this type of neural network. It convolves the input image with a set of learnable filters, each producing one feature map in

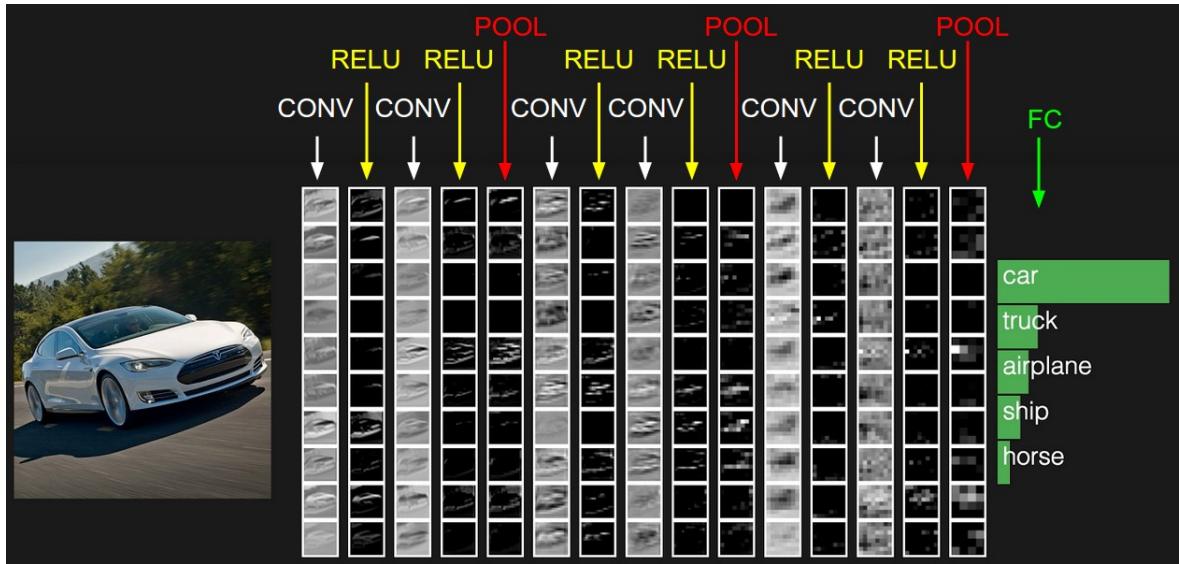


Figure 4.3: Example of a 16-layer deep convolutional neural network. The input layer is a whole picture, the output layer is the probability for each possible class. It used a succession of Convolutional, ReLU, Pooling layer with a final Fully connected one.

the output image, i.e. it computes a dot product on a neighborhhod of pixels:

$$y_{i,j} = b + \sum_{l=0}^{n-1} \sum_{m=0}^{n-1} w_{l,m} x_{j+l,k+m}$$

with:

- $x_{i,j}$  the input activation at position  $(x, y)$
  - $w_{l,m}$  the weights of the neuron
  - $n \times n$  is the size of the layer
  - $b$  is the bias value
  - $y_{i,j}$  the output values of the  $j, k$ th neuron
- **Activation layer:** element wise operation.

Example of function: the **Rectified Linear Unit *ReLU*** defines as:

$$f(x) = \max(0, x)$$

- **Pooling** or subsampling layer: down sampling of the input activation size. It reduces the number of values between the input and the output values of this layer to avoid overfitting the data and reduce the computation time of the neural network.

The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2 in figure 4.4.

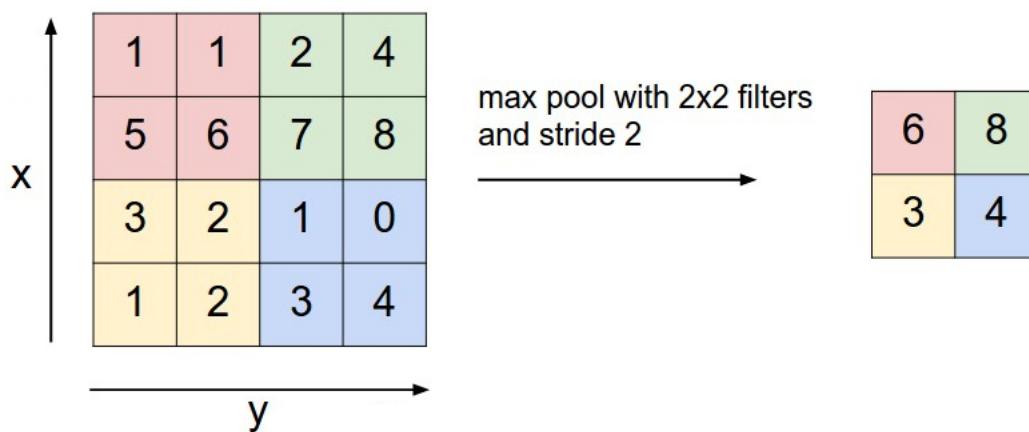


Figure 4.4: Illustration of a max pooling layer of stride 2, i.e. it selects the maximum value from a  $2 \times 2$  square

- **Fully connected**: compute the class scores. As the name implied, this neuron is connected to all activations from the previous values. For classification, it corresponds to a loss function, a common one is the sigmoid:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

A CNN can also be used as a feature descriptor if we use the output of the last layers.

# **Chapter 5**

## **Dataset**

Why do we use a dataset? - learning - some research make them freely available to test

Describe how it was build ?

### **5.1 Choice of the dataset**

Numerous datasets are already existing and have been made freely available. I could create my own dataset but it would have been very time consuming and I wouldn't be able to compare my results with previous scientific papers.

To choose, a couple of criteria were defined:

- Preferably, it should be a recent dataset
- It must have a decent number of pictures (a few thousand pictures)
- It must be composed of a general kind of food such as worldwide, Western or Asian
- It must contain pictures with multi-food items

As we can see in the table 5.1, UEC FOOD 256 is the dataset that best match our expectations.

Name	Re-lease date	Number of pictures	Type of food	Number of classes	Multiple food items
PFID [13]	2009	4545	American fast-food	101	No
UEC FOOD 100 [14]	2012	14361	Japanese	100	Yes
FIDS 30 [15]	2013	971	Fruit	30	No
ETHZ Food-101 [16]	2014	101 000	European	100	No
UPMC Food-101* [17]	2015	90 840	European	100	No
UNICT-FD889 [18]	2015	3 583	World	889	No
FooDD [19]	2015	3000	Fruit	23	Yes
<b>UEC FOOD 256</b> [20]	<b>2015</b>	<b>31395</b>	<b>World</b>	<b>256</b>	<b>Yes</b>

Table 5.1: Summary of some available food datasets according to the criteria.

\*UPMC FOOD 101 is including the recipe for most of the pictures

## 5.2 UEC FOOD-100 and UEC FOOD-256

**UEC FOOD-100** and **UEC FOOD-256** are datasets used for food localization and recognition.

The UEC FOOD-100 dataset can be found in <sup>1</sup>. It was created in 2012 and presented in [14].

It contains 100 types of food, mainly Japanese food. Each kind is represented by at least 100 samples.

As presented in figure 5.1, a photo can contain more than one food items. The dataset contains files to indicate bounding boxes marking the location of a food items.

UEC FOOD-256 can be found in <sup>2</sup>. It was presented in [20] in 2015. It contains the 100 types of food from UEC FOOD-100 plus 156 new ones. The pictures have been automatically extracted from the Internet and pre-processed.

The newly introduced food kinds are more international dishes with food from various

---

<sup>1</sup>Dataset can be found at <http://foodcam.mobi/dataset100.html>

<sup>2</sup>Dataset can be found at <http://foodcam.mobi/dataset256.html>

countries such as France, Italy, the USA, China, Thailand, Vietnam, Japan and Indonesia. As for FOOD 100, every food photo has a bounding box indicating the location of the food item.

The most represented category is miso soup with 728 and rice with 620 pictures.



Figure 5.1: Pictures with multiple food items from UEC FOOD 256

# Chapter 6

## Methodology

As illustrated in Fig. 6.1, we have our initial dataset that we split in :

- a **validation set** (10 % of the dataset) used for hyper-parameter optimization or model selection for localisation and classification
- a **train / test set** (remaining dataset) used for the localisation and classification. The train set is used to learn the parameters of a classifier that is then evaluated on the test set (using the same dataset for learning and testing would be a methodological mistake as it would overfit the dataset)

### 6.1 Hyperparameter optimization

There are numerous parameters that are part of the machine learning but are not learnt. Typical example include which kernel function used (if any) or the value of the penalty parameter  $C$  for SVM, the number of  $k$  of neighborhoods for kNN.

We use the exhaustive grid search method to select the parameters that have the highest performance score through 10 fold cross validation. It generates all the possible combi-

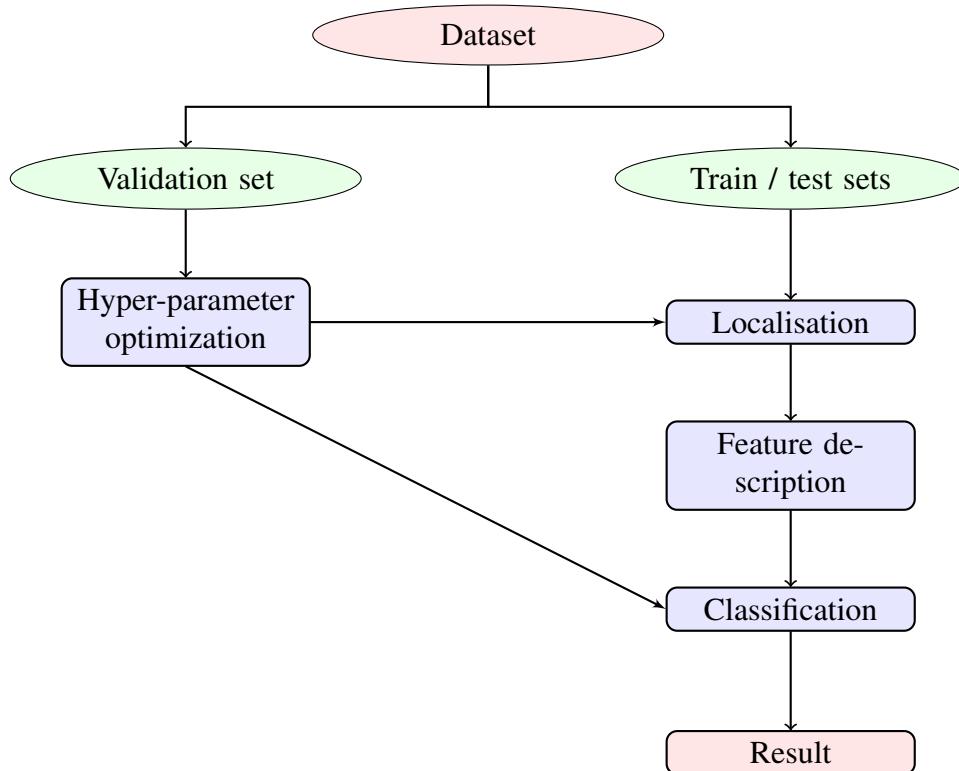


Figure 6.1: General process of the localisation and classification

nation of parameters value and train / test the classifier.

## 6.2 Localisation

For localisation, a different approach from the litterature has been used. The usual way is to detect area of food and non food in a picture. Yet, it was noticed that the food items of UEC FOOD 256 and 100 tends to be in the middle and stands out. Moreover, demanding the user to take pictures that follow these charastetcs is reasonable.

That's why a pre-trained CNN used for saliency detection has been used. It has been pre-trained in [21] on multiple datasets (Multi-Salient-Object, ILSVRC14). It is available

<sup>1</sup>.

---

<sup>1</sup><https://gist.github.com/jimmie33/339fd0a938ed026692267a60b44c0c58>

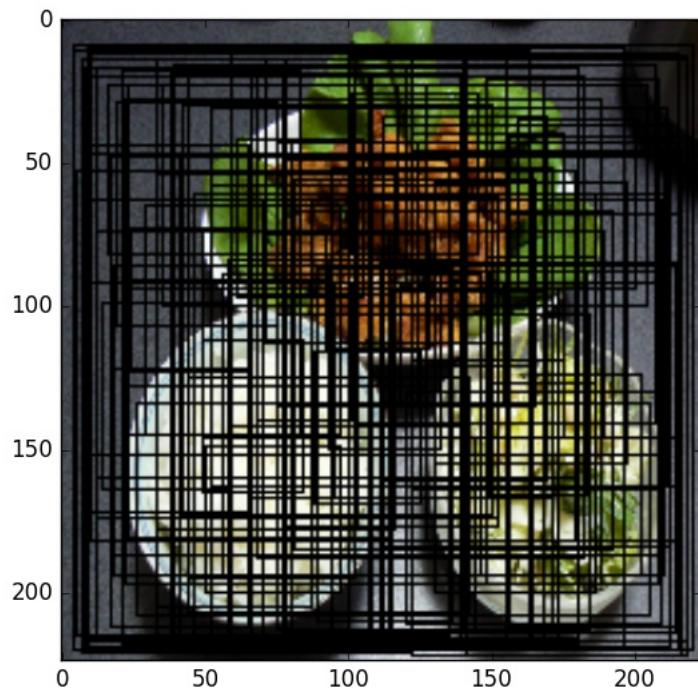


Figure 6.2: Picture of the 100 possible bounding boxes that the salient CNN will try to recognise

The CNN structure is a copy of “GoogleNet” model [22], i.e. it is composed of 22 layers, corresponding to a succession of convolutional, max pooling and activation layers, the last one being a sigmoid function.

The CNN has been pre-trained to detect the likelihood to belong to one of the 100 arbitrary bounding boxes as presented in Fig. 6.2.

## 6.3 Food recognition

### 6.3.1 Histograms and moments

The first feature descriptor used combined histograms of LBP and Color with color moments for each picture:

1. extract a 100-bin histogram of local binary pattern on the grayscale image
2. extract a 30-bin by 30-bin joint color histogram for the channel  $H$  and  $s$  of the HSV representation
3. extract the first two moments of the R, G, B, H, S and Gray channels
4. extract the 7 Hu moments

The feature vectors are then normalized to have all features centered around zero (mean equal to 0) and have unit variance (equal to 1).

Then, apply multiple classifiers:

- decision tree
- random forest (made up of 500 trees)
- SVM

Talk in result: show the best amelioration with hyperparameter (but in general it only improve it by one or two percents)

### 6.3.2 Bag of words

1. detection of keypoints using a dense grid (4 spaces)

2. descriptors: Root SIFT. Root SIFT is a simple variant of SIFT, presented in [23].

When the SIFT descriptors as been computed for each keypoints, we apply an element wise square root of the L1 normalized SIFT vectors

Then these feature vectors are clustered using the k-means algorithm to obtain a 1000-word codebook.

For each picture, we compute the histogram of occurence counts of visual words.

This descriptor is used with the SVM classifier and additive  $\chi$ -squared kernel

### 6.3.3 CNN as a Descriptor

As described in section 4, a CNN can be used as a feature descriptor. The pre-trained CNN was used for image recognition on ImageNet Challenge 2014 and presented in [24]. It is available<sup>2</sup>.

The model is an improved version of the 19-layer model used by the VGG team in the ILSVRC-2014 competition. As the CNN used for segmentation, it takes a  $224 \times 224$  RGB picture.

The output of the layer just before the FC is used.

## 6.4 Code

The code is freely available on Github<sup>3</sup>.

I'm using python 3.5.2 and its scientific stack bas on Scipy [25]:

- Numpy [26] for N-dimensional array
- Pandas [27] for the data structure

---

<sup>2</sup><https://gist.github.com/ksimonyan/3785162f95cd2d5fee77/>

<sup>3</sup>[https://github.com/bnogaret/food\\_log](https://github.com/bnogaret/food_log)

- Scikit-image [28] and opencv 3 [29] for some of the image processing algortihms
- Scikit-learn [30] for most of the machine learning and Caffe [31] for the CNN
- Matplotlib [32] for 2D graph generation
- Sphinx for the documentation

# **Chapter 7**

## **Evaluation**

### **7.1 Environment**

All the code has been run on the “Astral” high performance computer of Cranfield’s university. The operating system is SUSE Linux Enterprise Server 11 (64 bits architecture), with a Linux 3 kernel.

The system is separated in login nodes and compute nodes. There are two “front-end” login nodes and they contain two Intel E5-2660 (Sandy Bridge - 8 cores) CPUs giving 16 CPU cores and have a total of 192 GB of shared memory. The login nodes enable the user to connect to the system and compile one’s program. There are 80 compute nodes, each node having two Intel E5-2660 (Sandy Bridge - 8 cores) CPUs. This is giving a total of 1280 available cores. Each compute node have at least accessed to 64 GB shared memory. Nodes are connected with Infiniband<sup>TM</sup> low-latency interconnect.

## 7.2 Segmentation metrics

To measure the precision of the localization / segmentation algorithm, we use the metrics as defined in [33]<sup>1</sup>.

To be considered a correct detection, the **Intersection over Union** *IoU* between the predicted bounding box  $B_p$  and ground truth bounding box  $B_{gt}$  must exceed 50% by the formula:

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

To simplify the calculation, this formula can be rewritten as:

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p) + area(B_{gt}) - area(B_p \cap B_{gt})}$$

Using this metric, we can compute the precision  $P$ , the recall  $R$  and the accuracy  $A$  given by:

$$P = \frac{T_p}{T_p + F_p}$$

$$R = \frac{T_p}{T_p + F_n}$$

$$A = \frac{T_p}{T_p + F_n + F_p}$$

with:

- $T_p$  the number of true positives (the bounding boxes correctly localized)
- $F_p$  the number of false positives (the predicted bounding boxes incorrectly local-

---

<sup>1</sup>Information on the evaluation system can be found at [http://host.robots.ox.ac.uk/pascal/VOC/voc2012/devkit\\_doc.pdf](http://host.robots.ox.ac.uk/pascal/VOC/voc2012/devkit_doc.pdf)

ized)

- $F_n$  the number of false negative (the ground truth bounding boxes not localized)

Note that given the convention from [33], if more than one predicted bounding box overlaps the same ground truth bounding box, only one will be considered as  $T_P$ , the rest will be  $F_P$ s.

## 7.3 Cross validation

Cross validation is a technique used to assert the generalization to a new dataset of the different metrics used.

A common type of cross validation is the k-fold cross validation. In this method, the original sample is randomly split into  $k$  partitions of equal sized. Of these generated subsamples, a single split is used for test set, the remaining are used as training data. This last task is repeated  $k$  times, each of the  $k$  partitions being used only once for testing. The  $k$  results can then be averaged to produce a single estimation (illustrated in figure 7.1)

The advantage of this method over repeated random sub-sampling is that all observations are used for both training and testing, each observation being used for testing exactly once.

10-fold cross-validation were used for all the presented results (the most common fold value that maximises the training set size).

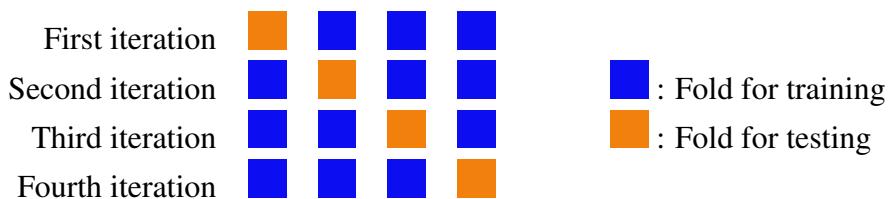


Figure 7.1: Illustration of 4-fold cross validation

## 7.4 Results

First, the localisation and classification processes were run independantly (using the ground truth bounding box for classification).

### 7.4.1 Food localisation

Metric (average)	My method	DCNN from [34]
Accuracy	<b>73 %</b>	60 %
Recall	<b>74 %</b>	80 %
Precision	<b>79 %</b>	70 %

Table 7.1: Average localisation accuracy result for UEC FOOD 256

The table 7.1 gathers the average accuracy, recall, precision of my localisation method using a DCNN pre-trained on salient object detection. In [34], the authors use fine-tuned pre-trained Deep Neural Network and obtain around 60 % of accuracy (using the same IoU over 50 %).

Compare to the found litterature, my method lead to a higher accuracy. It seems that the assumptions made to switch from a DCNN trained to detect food / non-food detection to salient object detection is founded.

For the result of the table 7.1, we use an IoU of 50 %. In Fig 7.2, we can see that the metrics' values are greatly influenced by the threshold choose for correctness (from 73 % of average accuracy with a threshold at 50 % to 0 % of accuracy for a thresholf of 100 %).

### 7.4.2 Classification

In [34], the authors use fine-tuned pre-trained Deep Neural Network and obtain 63 % accuracyon UEC FOOD-256.

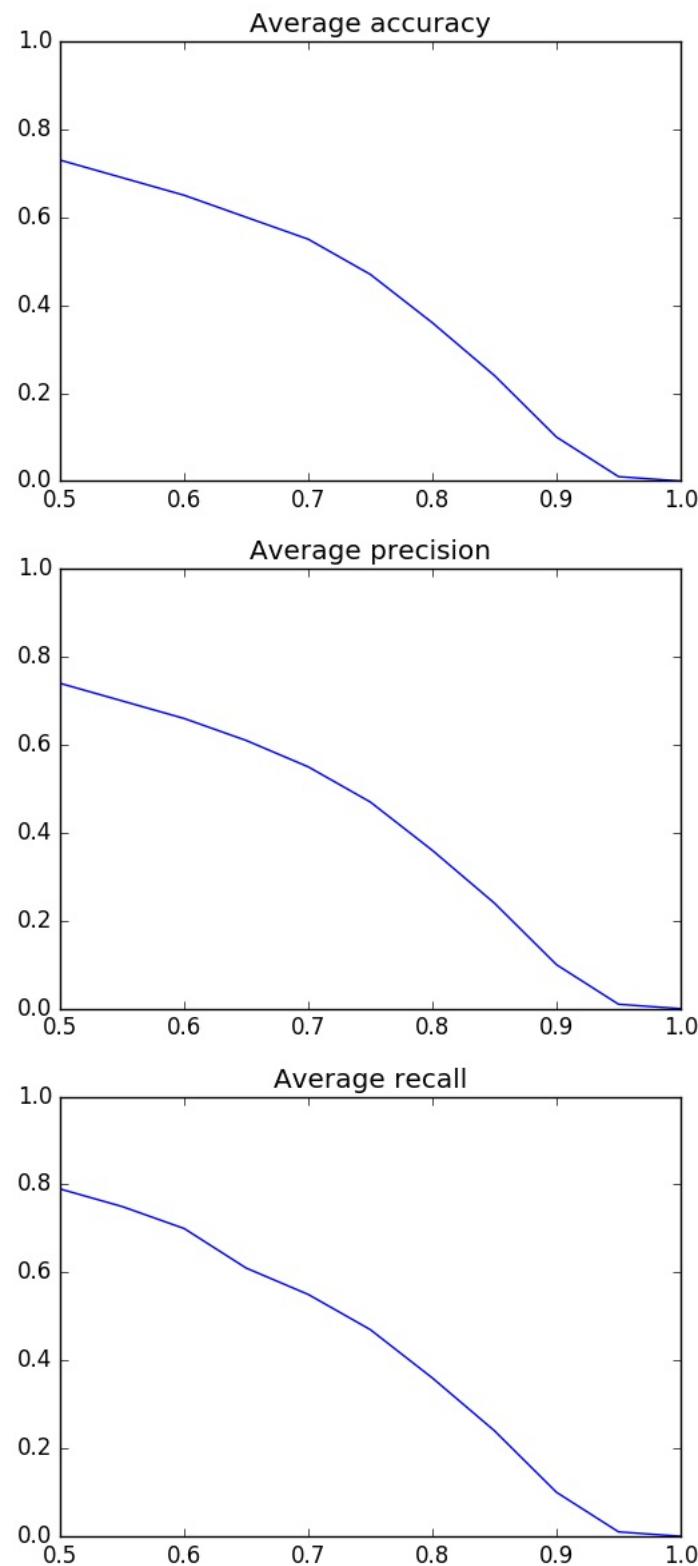


Figure 7.2: Curves of Accuracy over IoU (top), Precision over IoU (centre) and Recall over IoU (bottom)

Method	Average accuracy
CNN as descriptor + RF	<b>40 %</b>
BoW (1000 words)+ SVM with $\chi^2$	10 %
LBP + color histograms and moments + Decision tree	5 %
LBP + color histograms and moments + SVM	11 %
LBP + color histograms and moments + RF	16 %
DCNN from [34]	63 %
DCNN from [35]	67 %

Table 7.2: Average classification accuracy result for UEC FOOD 256

Process	My method	DCNN from [34]
Overall	<b>28 %</b>	37 %
Localisation	<b>74 %</b>	60 %
Classification	<b>38 %</b>	60 %

Table 7.3: Average accuracy result for UEC FOOD 256

In [35], the authors use fine-tuned pre-trained Deep Neural Network and obtain 67 % accuracy on UEC FOOD-256.

### 7.4.3 Segmentation followed by classification

My classification is slightly less accurate as the bounding box is not as precise as the ground truth.

Using the segmentation and classification method with the highest accuracy, i.e. CNN Segmente + CNN feature descriptor + RF classifier

As can be seen in Fig 7.3 and 7.4, the best performing class is rice and the least one is tanmen. The possible explanations are:

- rice is the most represented food items in the dataset, maximising the size of the training sets (same for miso soup)
- rice has a specific texture and color that is relatively invariant to the condition



Figure 7.3: The five classes having the highest accuracy with (from the left to the right, starting from the highest accuracy) rice (98 %), miso soup (95 %), grilles pacific saury (94 %), hamburger (93 %), roll bread (90 %)



Figure 7.4: The five classes having the lowest accuracy with (from the left to the right, starting from the lowest accuracy) tanmen (0 %), Pork with lemon (0 %), clear soup (1 %), yellow curry (1 %), grilles eggplant (1 %)

- tanmen is a soup containing noodle and various vegetables. Thus, it can occur in different color, shape and size.
- there are numerous soups in the dataset and tanmen is often confused with them.

Process	My method	DCNN from [36]	DCNN from [37]
Overall	<b>33 %</b>	-	-
Localisation	<b>67 %</b>	60 %	-
Classification	<b>50 %</b>	-	72 %

Table 7.4: Average accuracy result for UEC FOOD 100



Figure 7.5: The six most confused classes (with from the left to the right, starting from the lowest accuracy) clear soup and miso soup (83 %), chicken rice and fried rice (54 %)

# **Chapter 8**

## **Future work**

One of the possible future area of work is using a more accurate feature descriptor and / or classifier. Compared to the litterature, my food recognition is rather low. Using a pre-trained DCNN for food recognition seems a promising tool.

It could be also interesting to use multiple segmentation method to combine them.

Then, it can be added the estimation part that include an calorie estimation or a simplified version based on MyPyramid or MyPlate and an application to take picture and visualize user's record.



# Appendix A

## Appendix

### A.1 RGB to HSV

Assuming the RGB values have been normalised to be in  $[0, 1]$ , we have:

$$M = \max(R, G, B) \quad m = \min(R, G, B) \quad C = M - m$$

$$H = \begin{cases} 0 & \text{if } C = 0 \\ 60 \times \left[ \frac{G-B}{C} \mod 6 \right] & \text{if } M = R \\ 60 \times \left[ \frac{B-R}{C} + 2 \right] & \text{if } M = G \\ 60 \times \left[ \frac{R-G}{C} + 4 \right] & \text{if } M = B \end{cases}$$

$$S = \begin{cases} 0 & \text{if } M = 0 \\ \frac{C}{M} & \text{otherwise} \end{cases}$$

$$V = M$$

## A.2 HSV to RGB

The obtained R, G and B values are in  $[0, 1]$  and calculated as such:

$$C = V \times S \quad X = C \times \left(1 - \left|\frac{H}{60} \bmod 2 - 1\right|\right) \quad m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & 0 \leq H \leq 60 \\ (X, C, 0) & 60 \leq H \leq 120 \\ (0, C, X) & 120 \leq H \leq 180 \\ (0, X, C) & 180 \leq H \leq 240 \\ (X, 0, C) & 240 \leq H \leq 300 \\ (C, 0, X) & 300 \leq H \leq 360 \end{cases}$$

$$(R, G, B) = (R' + m, G' + m, B' + m)$$

# Bibliography

- [1] Ali H Mokdad et al. “Prevalence of obesity, diabetes, and obesity-related health risk factors.” In: *JAMA : the journal of the American Medical Association* 289.1 (2003), pp. 76–9. ISSN: 0098-7484. DOI: 10.1001/jama.289.1.76..
- [2] Ping Zhang et al. “Global healthcare expenditure on diabetes for 2010 and 2030”. In: *Diabetes Research and Clinical Practice* 87.3 (2010), pp. 293–301. ISSN: 01688227. DOI: 10 . 1016 / j . diabres . 2010 . 01 . 026. URL: <http://dx.doi.org/10.1016/j.diabres.2010.01.026>.
- [3] Lora E. Burke, Jing Wang, and Mary Ann Sevick. “Self-Monitoring in Weight Loss: A Systematic Review of the Literature”. In: *Journal of the American Dietetic Association* 111.1 (2011), pp. 92–102. ISSN: 00028223. DOI: 10 . 1016 / j . jada . 2010 . 10 . 008. URL: <http://dx.doi.org/10.1016/j.jada.2010.10.008>.
- [4] S W Lichtman et al. “Discrepancy between self-reported and actual caloric intake and exercise in obese subjects.” In: *The New England Journal of Medicine* 327.27 (1992), pp. 1893–1898. ISSN: 0028-4793. DOI: 10.1056/NEJM199212313272701. arXiv: [arXiv:1011.1669v3](https://arxiv.org/abs/1011.1669v3).
- [5] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252. ISSN: 15731405. DOI: 10.1007/s11263-015-0816-y. arXiv: [1409.0575](https://arxiv.org/abs/1409.0575).

- [6] Richard Hillestad et al. “Can electronic medical record systems transform health care? Potential health benefits, savings, and costs.” In: *Health affairs (Project Hope)* 24.5 (2005), pp. 1103–17. ISSN: 0278-2715. DOI: 10.1377/hlthaff.24.5.1103. URL: <http://www.ncbi.nlm.nih.gov/pubmed/16162551>.
- [7] Nir Menachemi and Taleah H. Collum. “Benefits and drawbacks of electronic health record systems”. In: *Risk Management and Healthcare Policy* 4 (2011), pp. 47–55. ISSN: 11791594. DOI: 10.2147/RMHP.S12985. arXiv: 0710.4428v1.
- [8] Joachim Dehais, Marios Anthimopoulos, and Stavroula Mougiakakou. “Dish Detection and Segmentation for Dietary Assessment on Smartphones”. In: *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops* 9281 (2015), pp. 433–440. ISSN: 16113349. DOI: 10.1007/978-3-319-23222-5. URL: [http://link.springer.com/chapter/10.1007/978-3-319-23222-5%7B%5C\\_%7D53](http://link.springer.com/chapter/10.1007/978-3-319-23222-5%7B%5C_%7D53).
- [9] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), pp. 971–987. ISSN: 01628828. DOI: 10.1109/TPAMI.2002.1017623.
- [10] Ming-Kuei Hu. “Visual pattern recognition by moment invariants”. In: *IRE Transactions on Information Theory* 8 (1962), pp. 179–187. ISSN: 0096-1000. DOI: 10.1109/TIT.1962.1057692.
- [11] David Arthur and Sergei Vassilvitskii. “k-means++: The Advantages of Careful Seeding”. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* 8 (2007), pp. 1027–1035. URL: <http://portal.acm.org/citation.cfm?id=1283494>.

- [12] A Vedaldi and A Zisserman. “Efficient Additive Kernels via Explicit Feature Maps”. In: *{IEEE} Int. Conf. on Computer Vision and Pattern Recognition XX.Xx* (2010), pp. 3539–3546.
- [13] Mei Chen et al. “PFID: Pittsburgh Fast-food Image Dataset”. In: *Proceedings - International Conference on Image Processing, ICIP* (2009), pp. 289–292. ISSN: 15224880. DOI: 10.1109/ICIP.2009.5413511.
- [14] Yuji Matsuda, Hajime Hoashi, and Keiji Yanai. “Recognition of multiple-food images by detecting candidate regions”. In: *Proceedings - IEEE International Conference on Multimedia and Expo*. IEEE, July 2012, pp. 25–30. ISBN: 978-1-4673-1659-0. DOI: 10.1109/ICME.2012.157. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6298369>.
- [15] Škrjanec Marko. “Automatic fruit recognition using computer vision”. Mentor: Matej Kristan. Bsc thesis. Faculty of Computer and Information Science, University of Ljubljana, 2013.
- [16] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. “Food-101 - Mining discriminative components with random forests”. In: *Lecture Notes in Computer Science*. Vol. 8694 LNCS. PART 6. 2014, pp. 446–461. ISBN: 9783319105987. DOI: 10.1007/978-3-319-10599-4\_29. arXiv: 978-3-319-10599-4{\\_}29 [10.1007]. URL: [http://link.springer.com/chapter/10.1007/978-3-319-10599-4%7B%5C\\_%7D29](http://link.springer.com/chapter/10.1007/978-3-319-10599-4%7B%5C_%7D29).
- [17] Xin Wang et al. “Recipe recognition with large multimodal food dataset”. In: *2015 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2015*. IEEE, June 2015, pp. 1–6. ISBN: 9781479970797. DOI: 10.1109/ICMEW.2015.7169757. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7169757>.

- [18] Giovanni Maria Farinella, Dario Allegra, and Filippo Stanco. “A benchmark dataset to study the representation of food images”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8927 (2015), pp. 584–599. ISSN: 16113349. DOI: 10.1007/978-3-319-16199-0\_41. arXiv: 1410.2488.
- [19] Parisa Pouladzadeh Abdulsalam Yassine and Shervin Shirmohammadi. “FooDD: Food Detection Dataset for Calorie Measurement Using Food Images”. In: *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops* 9281 (2015), pp. 441–448. ISSN: 16113349. DOI: 10.1007/978-3-319-23222-5. URL: [http://link.springer.com/chapter/10.1007/978-3-319-23222-5%7B%5C\\_%7D54](http://link.springer.com/chapter/10.1007/978-3-319-23222-5%7B%5C_%7D54).
- [20] Yoshiyuki Kawano and Keiji Yanai. “Automatic expansion of a food image dataset leveraging existing categories with domain adaptation”. In: *Lecture Notes in Computer Science* 8927 (2015), pp. 3–17. ISSN: 16113349. DOI: 10.1007/978-3-319-16199-0\_1.
- [21] Jianming Zhang et al. “Unconstrained Salient Object Detection via Proposal Sub-set Optimization”. In: *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)* (2016). URL: <http://cs-people.bu.edu/jmzhang/sod.html>.
- [22] C Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1–9. DOI: 10.1109/CVPR.2015.7298594. URL: /citations?view%7B%5C\_%7Dop=view%7B%5C\_%7Dcitation%7B%5C&%7Dcontinue=/scholar?hl=ja%7B%5C&%7Das%7B%5C\_%7Dsdt=0,5%7B%5C&%7Dscilib=1%7B%5C&%7Dcitilm=1%7B%5C&%7Dcitation%7B%5C\_%7Dfor%7B%5C\_%7Dview=KtmM-dAAAAAJ:JV2RwH3%7B%5C\_%7DSTOC%7B%5C&%7Dhl=ja%7B%5C&%7Doi=p.

- [23] Relja Arandjelovic and Andrew Zisserman. “Three things everyone should know to improve object retrieval c”. In: *IEEE Conference on computer vision and Pattern Recognition* April (2012), pp. 2911–2918. ISSN: 9781467312288. DOI: 10.1109/CVPR.2012.6248018.
- [24] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *ImageNet Challenge* (2014), pp. 1–10. ISSN: 09505849. DOI: 10.1016/j.infsof.2008.09.005. arXiv: 1409.1556. URL: <http://arxiv.org/abs/1409.1556>.
- [25] Travis E Oliphant. “SciPy: Open source scientific tools for Python”. In: *Computing in Science and Engineering* 9 (2007), pp. 10–20. ISSN: 1521-9615. URL: <http://www.scipy.org/>.
- [26] Stéfan Van Der Walt, S. Chris Colbert, and Gaël Varoquaux. “The NumPy array: A structure for efficient numerical computation”. In: *Computing in Science and Engineering* 13.2 (2011), pp. 22–30. ISSN: 15219615. DOI: 10.1109/MCSE.2011.37. arXiv: 1102.1523.
- [27] Wes McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference* (2010), pp. 51–56. URL: <http://conference.scipy.org/proceedings/scipy2010/mckinney.html>.
- [28] Stéfan van der Walt et al. “Scikit-image: image processing in Python”. In: *PeerJ* 2 (2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. arXiv: 1407.6245. URL: <https://peerj.com/articles/453>.
- [29] G Bradski. “The OpenCV Library”. In: *Dr Dobbs Journal of Software Tools* 25 (2000), pp. 120–125. ISSN: 1044-789X. DOI: 10.1111/0023-8333.50.s1.10. URL: <http://opencv.org/>.

- [30] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *... of Machine Learning ...* 12 (2012), pp. 2825–2830. ISSN: 15324435. DOI: 10.1007/s13398-014-0173-7. 2. arXiv: 1201.0490. URL: <http://scikit-learn.org/stable/>.
- [31] Yangqing Jia et al. “Caffe: Convolutional Architecture for Fast Feature Embedding”. In: *Proceedings of the ACM International Conference on Multimedia* (2014), pp. 675–678. ISSN: 10636919. DOI: 10.1145/2647868.2654889. arXiv: 1408.5093. URL: <http://arxiv.org/abs/1408.5093>.
- [32] John D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science and Engineering* 9.3 (2007), pp. 99–104. ISSN: 15219615. DOI: 10.1109/MCSE.2007.55.
- [33] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. URL: <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [34] Marc Bolaños and Petia Radeva. “Simultaneous Food Localization and Recognition”. In: (2016), pp. 2–7. arXiv: 1604.07953. URL: <http://arxiv.org/abs/1604.07953>.
- [35] Keiji Yanai and Yoshiyuki Kawano. “Food image recognition using deep convolutional network with pre-training and fine-tuning”. In: *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, June 2015, pp. 1–6. ISBN: 978-1-4799-7079-7. DOI: 10.1109/ICMEW.2015.7169816. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7169816>.
- [36] Wataru Shimoda and Keiji Yanai. “CNN-based food image segmentation without pixel-wise annotation”. In: *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*. Vol. 9281. 2015, pp. 449–457. ISBN: 9783319232218. DOI: 10.

- 1007/978-3-319-23222-5\_55. URL: [http://link.springer.com/chapter/10.1007/978-3-319-23222-5%7B%5C\\_%7D55](http://link.springer.com/chapter/10.1007/978-3-319-23222-5%7B%5C_%7D55).
- [37] Yoshiyuki Kawano and Keiji Yanai. “Food Image Recognition with Deep Convolutional Features”. In: *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)* (2014), pp. 589–593. DOI: 10.1145/2638728.2641339. URL: <http://dx.doi.org/10.1145/2638728.2641339>.