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 13, 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 13, 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

Food log; Localisation; Classification; Convolutional neural network

Contents

List of Figures

List of Tables

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.

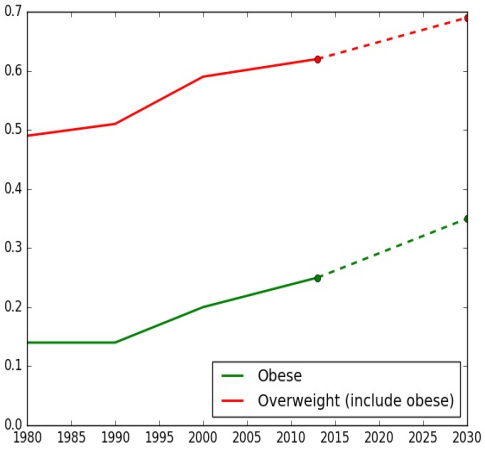

Figure 1.1: Obesity and overweight rate of the adult population in the uk between 1980 and 2030

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 foos 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 localisation, 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 classification error for each class and localisation 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 (nutritionist, psychologist) to understand the patient's behaviour and habits. It extends the reach of care in a cost effective ways and counters some of the

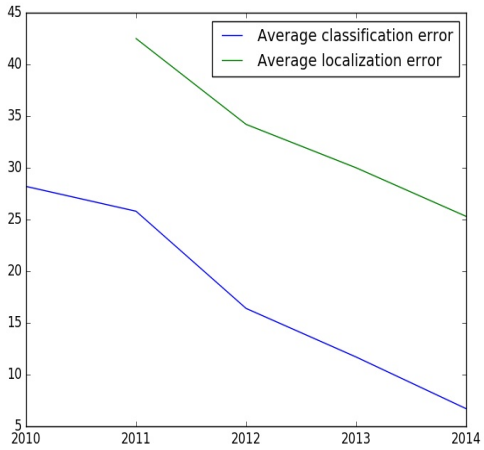

Figure 1.2: Average classification and localisation error of the best results for different ImageNet challenges

previous problem of manual report. It's part of the rise of e-healthcare / m-healthcare [6, 7].

Users upload pictures of their daily meals to the application or website that constructs their food diary automatically. Using image processing, it estimates the dietary composition of the meal and records the information for later viewing in formats such as tables or graphical representations.

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 multiple food items
- Recognise the food

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:
  - environmental conditions (e.g. luminosity, quality of the camera)
  - 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 kaya 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.

This makes localisation, classification and retrieval of food images a difficult task for current state-of-the-art techniques, and hence a compelling challenge for image processing and machine learning researchers.

The organization of this thesis is as follow. In section 2, previous work on food localisation, recognition and intake estimation is reviewed. Section 3 introduces the different image descriptors used, and in section 4 the classifiers are presented. In section 5, the dataset is introduced. Section 7 reports the experimental settings and results. Finally, in section 8, we draw the conclusion and state the limitation and possible future work.

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

2.1 Food localization

A way to localize food is based on edge detection and colour segmentation.

In [8], the authors describes and compare these two methods to localise an orange in a picture. It applied these methods on a small dataset of 20 orange images (only one orange per image), with different lighting conditions and backgrounds (pictures are taken from the Internet). In more details, the edge-based segmentation apply the canny-edge segmentation, then apply non-maximum suppression to eliminate noises. Then, each pixels are classified. The colour-based segmentation normalised the lightning condition with a Gaussian low-pass filter, convert the RGB image into a $L*a*b$

1. Gaussian low pass filter to normalize the lightning condition
2. convert the image from RGB representation to $L*a*b$
3. use the a channel to classify each pixel as “fruit” or “non-fruit”
4. remove small object

5. fill the binary image regions and holes

For orange detection, the colour segmentation has an higher accuracy. Yet, it is very hard to generalise this method.

An other method for food detection relies on circle detection. Indeed, food items are often served in a round shape container such as a bowl, pan or plate.

In [9], the authors describe their use of the Hough transformation to detect circle (they constitutes the food region).

A more recent development is the used of convolutional neural network.

In [10], the authors presents their segmentation process based on a pre-trained deep CNN.

The proposed pipeline is composed of 6 main steps:

1. detect all the possible bounding box (maximum 2000 per image) using selective search

2. cluster the bounding box, using the ration of intersection over union (IOU, also call overlap ratio) to obtain 20 at most.

3. a Deep CNN for all the selected bounding box to get a saliency map. The DCNN is modelled on AlexNet CNN, was pre-trained on the Salient Object Subitizing (14 000 everyday pictures) dataset and fine-tuned on UEC FOOD 100.

4. use the GrabCut algorithm to extract the foreground region from the food area.

5. In case of overlapped bounding box, the authors proposed to apply the non-maximum suppression (NMS) algorithm.

The authors apply this process on the UEC-FOOD 100 dataset and PASCAL VOC 2007. The latter is used for object detection and recognition of 20 common classes (train,

tv, cat, human ...)). These two datasets use bounding box to spot items. A segmentation is correct if the overlap ratio exceeds 50% between the predicted and the ground truth bounding box.

For UEC-FOOD 100, the authors obtain 49.9 % mean average accuracy and 58.7 % for PASCAL VOC 2007.

2.2 Food recognition

Food recognition

Using SVM:

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

Mix of several features:

More recently, people have started to heavily use Convolutional Neural Networks *CNN* with great results.

In [11], the authors use a pre-trained Deep CNN *DCNN* for feature extraction. The DCNN, called OverFeat, [1] was trained on ImageNet and is composed of 19 layers. The authors add more conventional image features to obtain feature vectors composed of:

1. a variant of the Histogram of Oriented Gradients *HOG* called "RootHog" that is an element-wise square root of the L1 normalized HOG

2. mean and variance values of each channel of the RGB representation value of pixels from each of 2*2 block

3. the last two layers of the DCNN

[1] Can be found `http://cilvr.nyu.edu/doku.php?id=code:start`

The three descriptors are then encoded in a fisher vector. Using SVM, the authors obtain 72% of accuracy for UEC-FOOD 100.

[12] use a fine-tuned DCNN, pre-trained on 2000 classes (including 1000 food classes) from ImageNet. The authors obtain 78 % accuracy on UEC-FOOD 100 and 68 % for UEC-FOOD 256. This dataset, presented in [13], is an extension of 256 classes of UEC-FOOD 100 using a so-called "foodness classifier" and transfer learning on images coming from crowdsourcing. In [14], the authors use a pre-trained DCNN on UEC-FOOD 256.

2.3 Food intake estimation

FoodLog [2] is a website that enables the user to upload pictures of its daily meals to be archived and processed. The goal of this application is to assist the user to keep notes of their meals and balance the nutritional values coming from different kinds of food.

In [15], the images containing food items are identified by exploiting features related to the HSV and RGB colour domains, as well as the shape of the plate. A SVM classifier is trained to detect food images. More specifically, the images are divided in 300 blocks and each block is classified as "non-food" (discarded block) or one of the nutritional categories described in the "MyPyramid" model [3].

MyPyramid [16] was designed by the United State Department of Agriculture *USDA* in 2005 and was replaced in 2011 by "MyPlate" [4] [17]. This dietary model is composed of 5 kinds of food: grains, vegetable, meals and beans, milk and fruit. For each group, a recommended intake per day is associated, Fig. 2.1. Quantity is categorized by "servings" *SV*, making it simpler to compute and keep log.

In [18] the Support Vector Machine is replaced by a Bayesian Framework *BF*. The

[2] http://www.foodlog.jp
[3] http://www.mypyramid.gov
[4] http://www.choosemyplate.gov

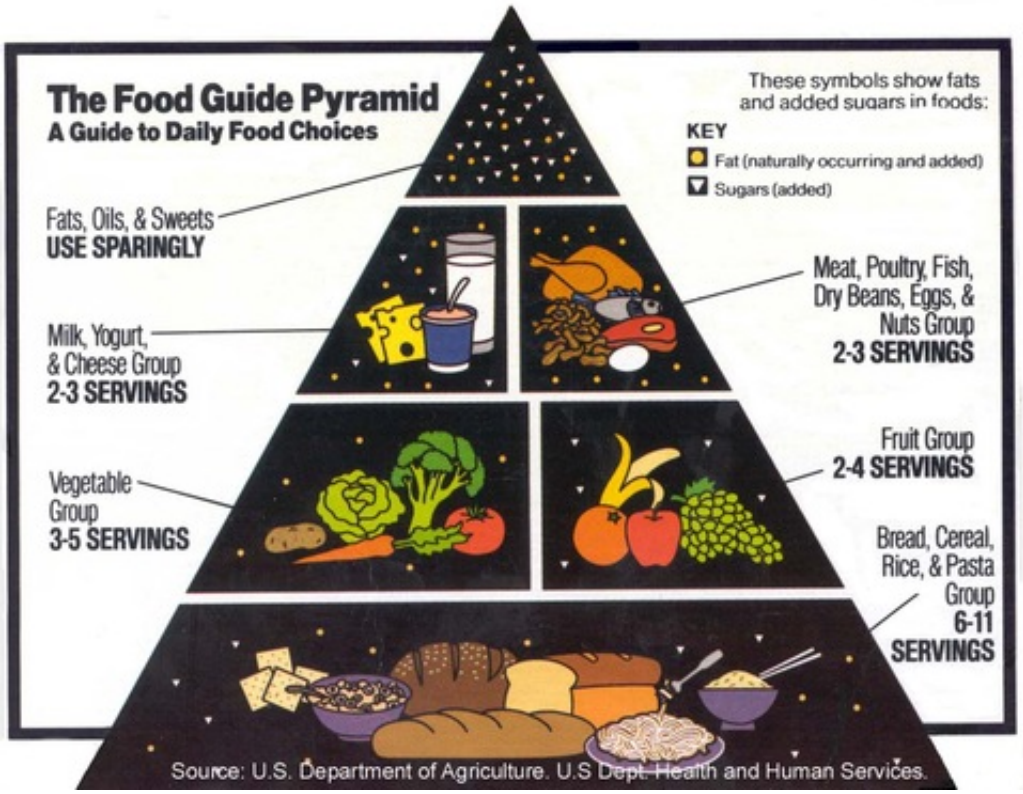

Figure 2.1: USDA MyPyramid original logo

BF is based on the Gaussian Naive Bayesian (suppose independence between every pair of features and the distribution of each feature is assumed to be Gaussian). The BF takes into account the estimation using colour moments and Bag-Of-Feature of SIFT, the prior distribution and the mealtime category (breakfast, lunch and dinner).

In [19], the authors use a Convolutional Neural Network [CNN] to detect and classify food from a small subset of image loaded in the FoodLog system. Compared to the other conventional methods (use of a feature descriptor such as Bag-of-Words with a classifier, e.g. SVM) described previously, the CNN showed a significantly higher accuracy.

An other method to estimate the food intake is to evaluate the food volume.

In [20], the authors presents a method that use the depth information of the picture. Once the food has been classified, the area of the food container (bowl, plate) and the depth value of the contained food is computed to obtain the food volume. Yet, this technique is still limited as it can only be used for non-transparent food, i.e it can't detect some food item such as water or cooked rice, and force the user to have a depth camera (such as Kinect).

In [21], the authors presents a novel food recognition system that is able to estimate of the nutrition intake. Moreover, they develop a mobile application to easily take pictures and keep track of the user's diet. To measure the food intake, authors compare before and after eating pictures and use the thumb as the calibration system (it supposes a one-time calibration to know the size of the thumb of the user). The process to show the intake is:

1. the user takes food pictures

2. get the contour of each picture

3. recognition of the food using colour, shape and size features with SVM.

4. volume calculation, that is computed in two steps:

(a) user takes a picture from above. Then, the food shape is divided into known shape (rectangle, circle, triangle ...) to compute the area.

(b) user takes a second picture from the side. This is used to compute the height of the food and calculate the overall volume.

The system assumes that the plate is white and round.

5. use a nutrition database to obtain the average calories

If the user has not eaten everything, the entire must be repeated. The drawbacks of this method is the user have to take several pictures, with one's thumb each time and it has been tested with a limited set of simple food types.

In [22], the authors develop a mobile application to keep food records of a user that is taking pictures of one's meal. Their method can detect multiple food items in one picture. They use a colour marker (color chequerboard) as an illumination and size indicator. As in [21], images obtained before and after foods are eaten are used to estimate the amount of food consumed.

When the user upload a picture, it is segmented, then classified by a back-end server. The estimation (labelled image with food type and volume) are sent back to the user for confirmation.

For segmentation, the authors use connected component analysis, active contours, and normalized cuts. Then, colour and texture features are extracted to feed a SVM classifier. The authors use:

- Gabor filters. Gabor filters describe properties related to the local power spectrum of a signal and have been used for texture analysis
- 2-D colour histograms of the a* and b* channels of the CIELaB representation. Values are corrected using the colour marker

For the volume estimation, the authors use a 3-D volume reconstruction process. The food area is partitioned and assigned to "geometric classes", each with their own sets of parameters.

They evaluate their segmentation and classification methods on a very small dataset composed of 63 images and 19 classes. The authors obtain an average accuracy of 89 %.

In [23], their method is named "multiple hypotheses segmentation and classification" *MHSC*. It is an iterative algorithm composed of a segmentation, description (extraction of features) and classification steps.

For segmentation, the authors first detect salient region, using Canny edge and colour distribution to reject background. Then, they apply a multi-scale segmentation using normalized cut. Small segmented regions are discarded.

On the selected region, the authors used a mixed of global descriptors (first and second moment of each channel for RGB, YCbCr, L*a*b*, and HSV colour spaces, first and second moment of the entropy in RGB, predominant colour descriptor, entropy and two first moments of the Gradient Orientation Spatial-Dependence Matrix, entropy categorization and fractal dimension estimation and estimation of the fractal dimension of the response of different Gabor filter) with local feature (multi Bag-Of-Words using SIFT for RGB, SURF for RGB, SIFT for each channel of the RGB representation and steerable filters).

Each of the 12 descriptor, global and local, is classified independently and assigned a confidence score. A late fusion function (either maximum confidence score or majority vote) is used to decide the final class. For classification, the authors use K-NN and SVM.

If the total score is inferior to a certain threshold, the overall process is repeated. The confidence score of the previous step is used to improve the segmentation.

Applied on a dataset composed of 83 labels (79 food classes plus "utensils", "glasses", "plates", and "plastic cups" classes), each class having at least 30 images, they obtain a top-8 accuracy of 75 %, using K-NN with the maximum confidence score.

In [24], the authors propose a food recognition system named **FoodCam** to identify food items of a picture. The presented process is used on a mobile application, the user taking a picture that is transferred to a sever, processed and results are displayed.

The first step is to detect potential region with multiple object detection algorithms. Then, for these regions, several feeatures are extracted and used to feed SVM with Multiple Kernel Learning *MKL* method. To detect candidate regions, the authors use:

- Felzenszwalb's deformable part model (DPM), based on Histogram of Oriented Gradients (HOG).
- a circle detector: the image is converted to a gray-scale, contour are extracted using the Canny Edge Detector and circles detected by the Hough Transform
- JSEG region segmentation: segment region based on colour. It only keeps circular regions.
- whole image, for picture with one large dish

Then, it aggregates all the candidate regions to get the bounding box of each food item.

For each region, it extracts multiple common features:

- Bag of Feature of SIFT and C-SIFT (sift with colour invariant characteristics)
- Spatial pyramid representation: object regions are divided by hierarchical grids. In this paper, the three level pyramid is used: 1×1, 2×2, 3×3. For each grid, a BoF vector is extracted
- Histogram of Oriented Gradient (HOG)
- Gabor texture

After extraction of the feature vectors from each candidate region, a linear SVM trained by MKL is used (χ^2 kernel).

For evaluation, the authors build a new dataset composed of 100 categories of traditional Japanese dishes with their associated bounding boxes for a total of 9060 images named **UEC-FOOD 100**. For multiple food item images, they obtain 55.8 % classification rate and 68.9 % for single food item pictures.

In [25], the authors develop a mobile real-time food recognition system for calorie and nutrition estimation. Contrary to the previous paper, all the calculation are realised on the user smartphone. The recognition takes less than 1 second thanks to the multi-core architecture of modern smartphones. The user takes a picture and draws bounding boxes around food items. Then, the system refine the segmentation based on the users' rough demarcation using Grabcut, an iterative method using graph cuts to extract foreground. For each item, it extracts image features and classify the image among the one hundred food classes using a linear SVM. Then, the top five food candidates are shown and the user can select one of the proposition. This recognition is updated every one second, the direction arrow as presented in figure 2.2 being displayed to help the user improve the result by changing the camera position and direction. To estimate the most suitable direction, the authors use the Efficient Sub-window Search method, a recent and powerful window search algorithm used in object detection. The mobile application keep records of all the pictures and their approved classification and labeled with the volume estimation. Food intake is estimated thanks to a slider on the bottom-left of the screen.

Two different descriptors are used:

- bag-of-feature, SURF for detection and description, and colour histogram with the chi^2 kernel feature map

- HOG and a colour patch descriptor (mean and variance of RGB values on a $2 \times$

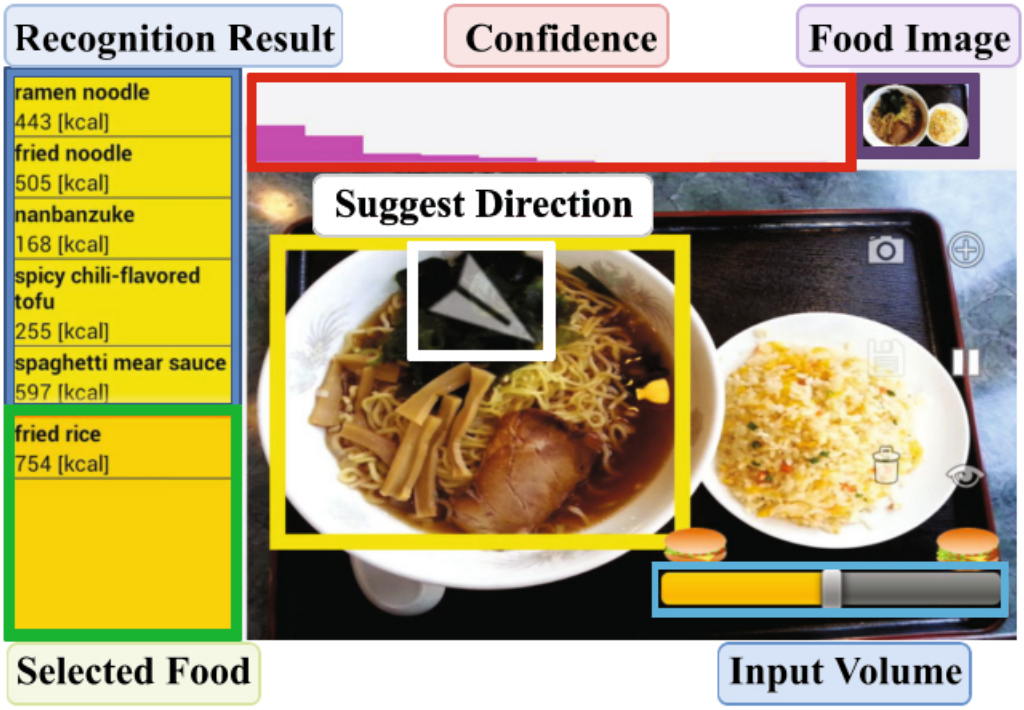

Figure 2.2: Annotated screenshot of the FoodCam application

2 blocks of pixel) encoded using Fisher Victor, a patch encoding strategy using Gaussian mixture models.

The authors evaluate these two methods on UEC-FOOD 100. Taking the top 5 classes, they obtain 79 % classification accuracy for colour patches and 68 % for the other.

Chapter 3

Feature descriptor

For a computer, a picture is represented as a 2-D or 3-D array. To facilitate the classification, feature descriptor extract derived values (the features), calculated to be more informative and invariant to some common picture transformations.

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

Another import feature of food is the texturee. Numerous texture descriptors can be used such as Gabor filters or Local Binary Pattern.

3.1 Local binary pattern

Local binary pattern is a visual descriptor for texture composition of an image, first presented in 2002 in [26] (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 centre 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 centre pixel's are labelled 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.

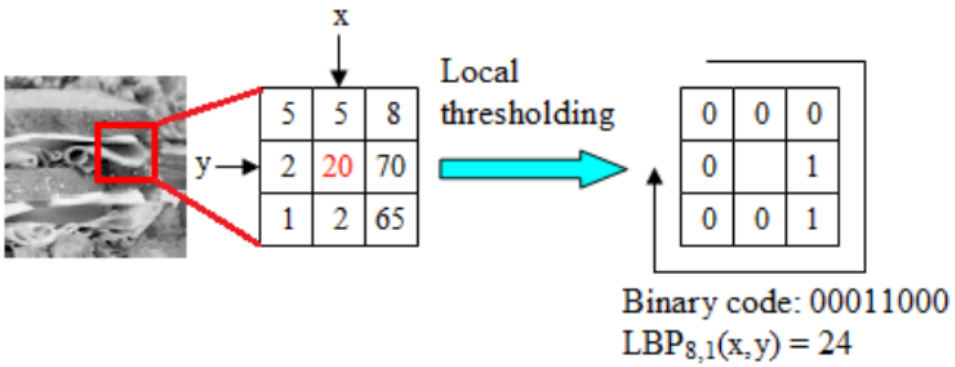

Figure 3.1: Illustration of the LBP descriptor's process

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 centred in c and P is the numbered of sampled points.
- g_p and g_c are the grey values (intensities) of p and c

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

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

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 [26], 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:

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

3.2 Color descriptor

3.2.1 Color histogram

HSV space is composed of:

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

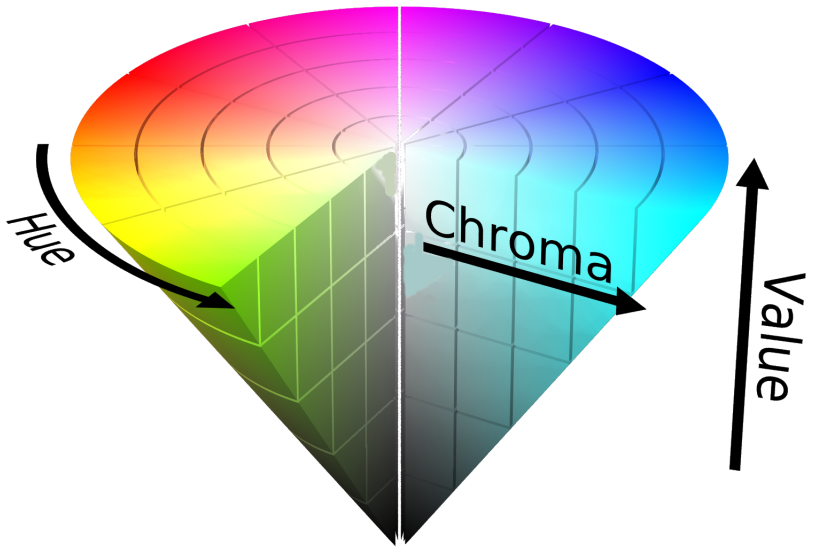

Figure 3.2: Pyramid representation of the HSV channels

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 colour

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 six-sided inverted pyramid (see figure 3.2).

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**:

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

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 [27] introduced 7 Moments which are invariant for translation, rotation and resizing:

$$\begin{aligned}
I_1 =& \eta_{20} + \eta_{02} \\
I_2 =& (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 =& (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 =& (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 =& (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
& + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
I_6 =& (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 =& (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
& - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned}$$

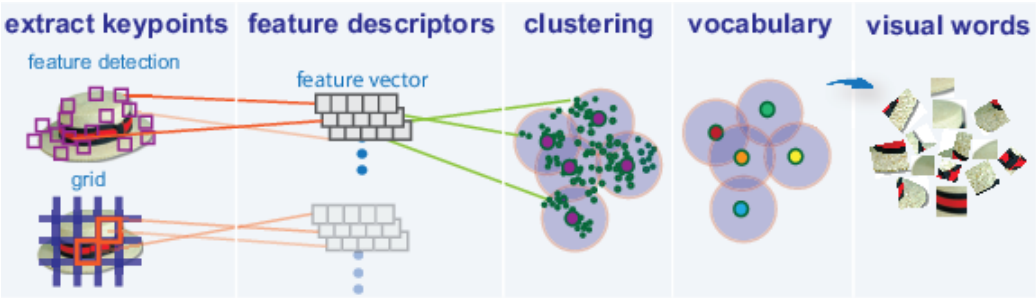

Figure 3.3: Illustration of the Bag-Of-Visual-Words model

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. 3.3, the main steps are:

- detecting keypoints on each picture. In my case, I use a dense grid of evenly spaced points at a fixed scale and orientation.
- describing each keypoints, i.e. extract a feature vector on a neighbourhood of pixels. SIFT, HOG and SURF are common descriptors.
- Generating a 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 and SURF

Scale-Invariant Feature Transform *SIFT* is an algorithm used for detection and description of local feature [28].

The major stages of the algorithm are:

1. Scale-space extrema detection: The scale space of an image $L(x,y,\sigma)$ is defined as the product of the convolution of a Gaussian filter $G(x,y,\sigma)$ and an image $I(x,y)$:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

where $*$ is the convolution at (x,y) and $G(x,y,\sigma) = \frac{1}{2\pi\sigma^2}\exp(-(x^2+y^2)/2\sigma^2)$.

Laplacian of Gaussian $\sigma^2\nabla^2 G$ produced the most stable image features but are expensive to compute, thus it is approximated as an Difference of Gaussian (scale-normalized LoG for scale-invariance)

$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G$$

As presented in figure 3.4, pyramid of DoG for each octave of scale space is computed: the initial image is repeatedly convolved with Gaussian filters for different values of σ to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian.

2. Keypoint localisation: In order to detect the local maxima and minima of $D(x,y,\sigma)$, each sample point is compared to its eight neighbours in the current image and nine neighbours in the scale above and below. Low contrast and edge keypoints are filtered.

3. Orientation and magnitude assignment: by assigning a consistent orientation to

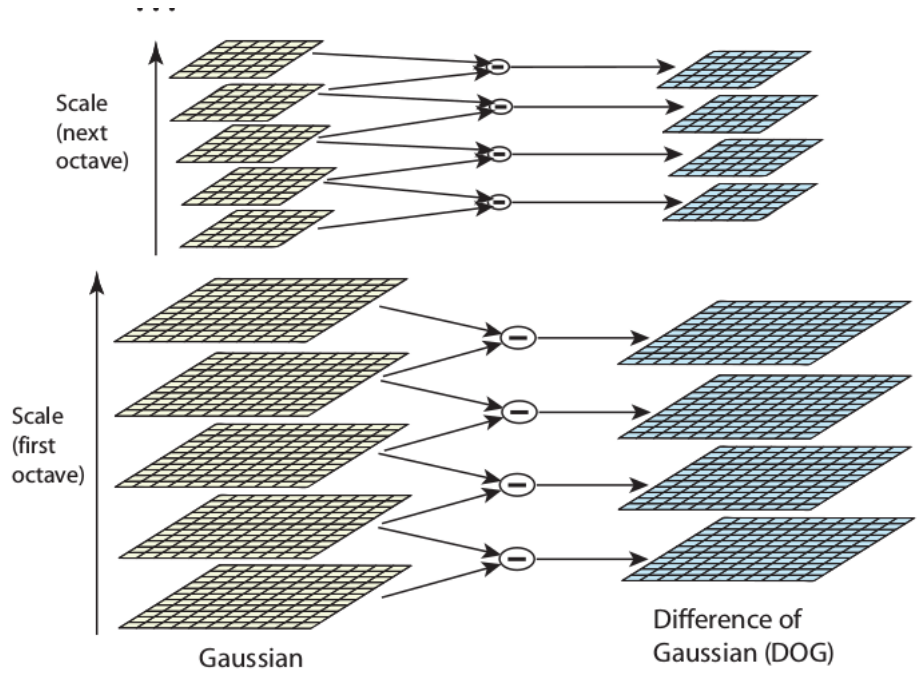

Figure 3.4: Illustration of the difference of Gaussian over multiple octave

each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation.

The magnitude and orientation are defined as:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)})$$

4. Keypoint descriptor: the local image gradients of a keypoint as presented in figure 3.5 are computed and accumulated in a histogram. An additional Gaussian weighting function is applied to give less importance to gradients farther away from the keypoint centrer. Once a keypoint candidate has been found by comparing a pixel to its neighbours, the next step is to perform a detailed fit to the nearby data for

location, scale, and ratio of principal curvatures. This information allows points to be rejected that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

Usually, SIFT is evaluated at 8 orientation planes over a 4×4 neighbourhood giving a 128-dimension feature vector

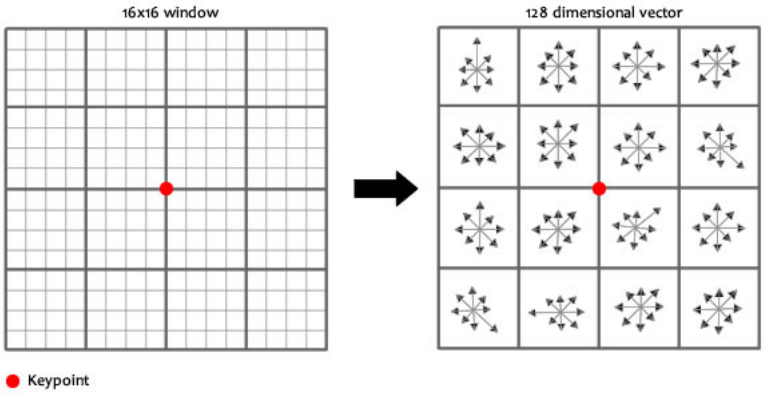

Figure 3.5: Illustration of SIFT as a local image descriptor

As proven in [28], this method is invariant to translation, scaling and rotation of the picture and is robust to illumination changes, addition of noise, change in the 3-D viewpoint and local geometric distortion.

Multiple variant of SIFT exists. Colour SIFT computes the SIFT in the same manner than the grey scale, except that it does it for each channel independently. Root SIFT is a simple variant of SIFT, presented in [29]. When the SIFT descriptors as been computed for each keypoints, we apply an element wise square root of the L1 normalized SIFT vectors

However the SIFT algorithm is quite slow method. That's why in [30], the authors present a faster algorithm base on the SIFT approach - **Speeded Up Robust Features** *SURF*. The key is to approximate the LoG with the Box Filter which is the other approximation but of DoG. Using them, the integral image can be constructed and used, which

speeds up the whole process since there is no need to iteratively apply those filters one by one (it can be even parallelized). This approach is eagerly applied in the real-time object recognition tasks.

3.3.3 K-mean clustering

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

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{3.2}$$

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 $M = \{m_1, m_2, \ldots, m_k\}$

2. Then, k clusters are created by associating every observation with the nearest mean.

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

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

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

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 initialisations.

To help to overcome this issue, the “kmeans++” initialisation scheme is often used, which has been described in [31]. This method initializes the centroids to be (generally) distant from each other, leading to provably better results than random initialization.

Chapter 4

Classifier

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.

The most used impurity measure's functions are:

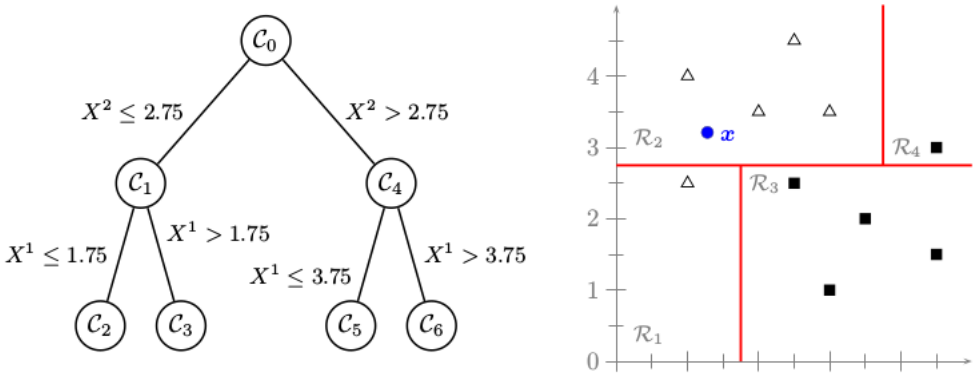

Figure 4.1: Decision tree of depth two for ten elements (X^1, X^2) belonging to the black square and white triangle classes

- **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

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

4.2.1 Linear SVM

Hard margin

A support vector machine constructs a hyper-plane or a 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:

$$\vec{x_i} \cdot \vec{w} + b \geq +1 \text{ for } y_i = +1 \tag{4.1}$$

$$\vec{x_i} \cdot \vec{w} + b \leq -1 \text{ for } y_i = -1 \tag{4.2}$$

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

Combining equation 4.1 and 4.2, we obtain:

$$\forall i \in 0, \ldots, n, \;\; y_i(\vec{x_i} \cdot \vec{w} + b) - 1 \geq 0$$

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

Geometrically, the distance between the two hyperplanes 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:

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

which is equivalent to minimize:

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

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:

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

The decision function for new example is:

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

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 others 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)**: $k(x,y) = \exp\left(-\gamma \|x-y\|^2\right)$
- **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 [32] of this kernel is the **Additive Chi-Square** 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, γ, C and must be choosen 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 scheme.

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 backward propagation of the errors (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.

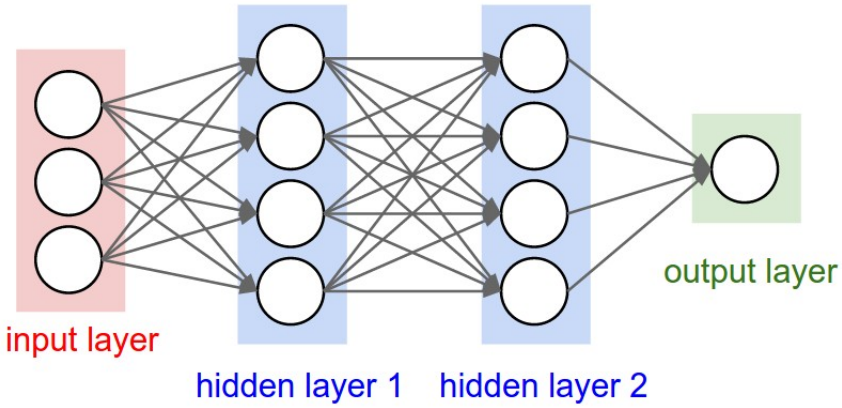

Figure 4.2: A regular 3-layer neural network

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

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

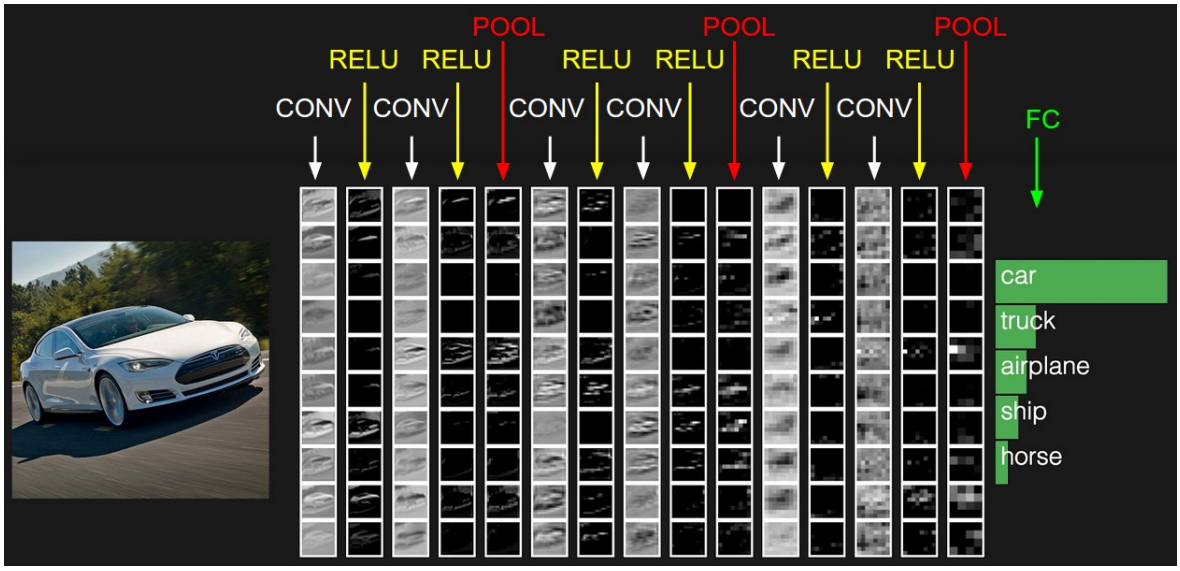

Figure 4.3: Example of a 16-layer deep convolutional neural network. The input layer is a whole picture, the output layer is the probability for each possible class. It used a succession of Convolutional, ReLU, Pooling layer with a final Fully connected one.

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, kth 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 re-

duces 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.

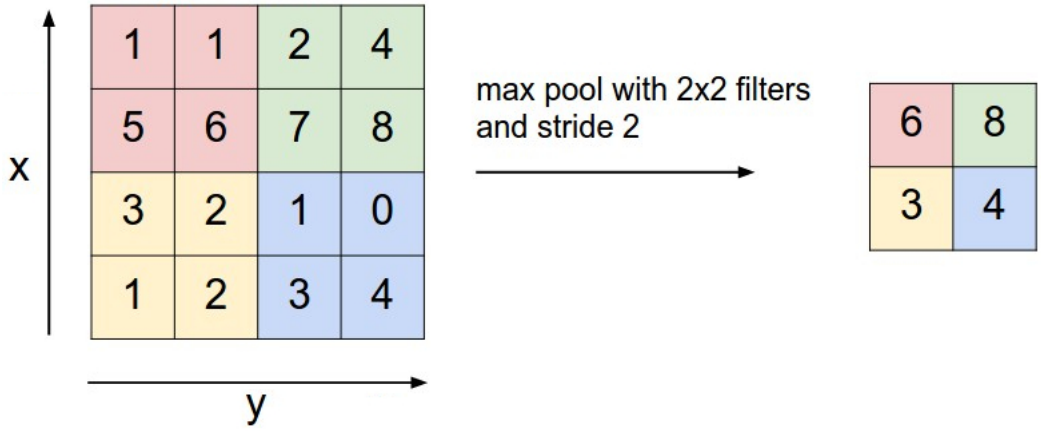

Figure 4.4: Illustration of a max pooling layer of stride 2, i.e. it selects the maximum value from a 2×2 square

- **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 datatset

Numerous datasets are already existing and have been made freely available. Creating one's own dataset was an option but it would have been very time consuming and our method's result could not be compared to 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 images)
- 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	Release date	Number of pictures	Type of food	Number of classes	Multiple food items
PFID [33]	2009	4545	American fast-food	101	No
UEC FOOD 100 [24]	2012	14361	Japanese	100	Yes
FIDS 30 [34]	2013	971	Fruit	30	No
ETHZ Food-101 [35]	2014	101 000	European	100	No
UPMC Food-101* [36]	2015	90 840	European	100	No
UNICT-FD889 [37]	2015	3 583	World	889	No
FooDD [38]	2015	3000	Fruit	23	Yes
UEC FOOD 256 [13]	**2015**	**31395**	**World**	**256**	**Yes**

Table 5.1: Summary of some available food datasets according to the criteria.
*UPMC FOOD 101 is also 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 [1]. It was created in 2012 and presented in [24].

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 [2]. It was presented in [13] 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

[1]Dataset can be found at `http://foodcam.mobi/dataset100.html`
[2]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 lead to overfit the dataset and will not represent the capacity of the method to recognise new unknown element)

6.1 Hyperparameter optimization

There are numerous parameters that are part of the machine learning algorithm 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 neighbourhoods for K-NN.

We use the exhaustive grid search method to select the parameters that have the highest

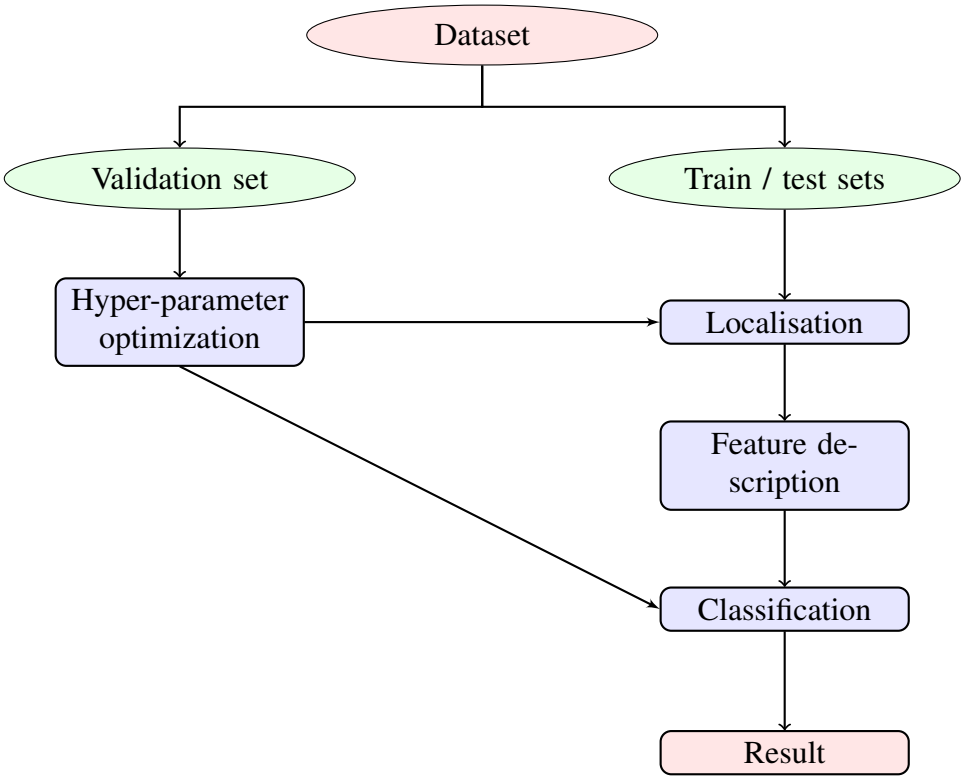

Figure 6.1: General process of the localisation and classification

performance score through 10 fold cross validation. It generates all the possible combination of parameters value and train / test the classifier.

6.2 Localisation

For localisation, a different approach from the literature 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, requesting the user to take pictures that follow these characteristics is reasonable.

That's why a pre-trained CNN used for saliency detection has been used. It has been pre-trained in [39] on multiple datasets (Multi-Salient-Object, ILSVRC14). It is available

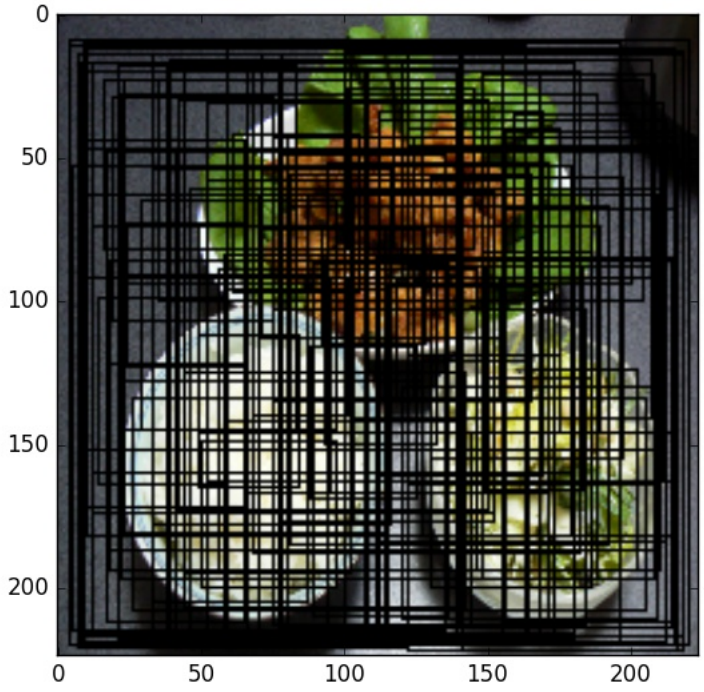

Figure 6.2: Picture of the 100 possible bounding boxes that the salient CNN will try to recognise

on this gist [1].

The CNN structure is a copy of "GoogleNet" model [40], 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.

Bounding boxes with a probability higher than a threshold T are selected as candidate (if no box meet this limit, the bounding box with the maximum value is selected). As can be seen in figure 6.3, it generates a lot of overlapping copies. That's why, the final step of the localisation process is to discard small bounding boxes and overlapping ones (overlap

[1] `https://gist.github.com/jimmie33/339fd0a938ed026692267a60b44c0c58`

higher than 30 %), keeping the ones with highest probabilities.

6.3 Food recognition

6.3.1 Histograms and moments

The first feature descriptor used a combination of LBP histogram with colour moments and histogram for each picture:

1. extract a 100-bin histogram of local binary pattern on the greyscale image
2. extract a 30-by-30-bin joint colour 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 centred around zero (mean equal to 0) and have unit variance (equal to 1).

Then, multiple classifiers are applied :

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

6.3.2 Bag of words

The usual process of Bag-of-Features is used:

1. detection of keypoints using a dense grid

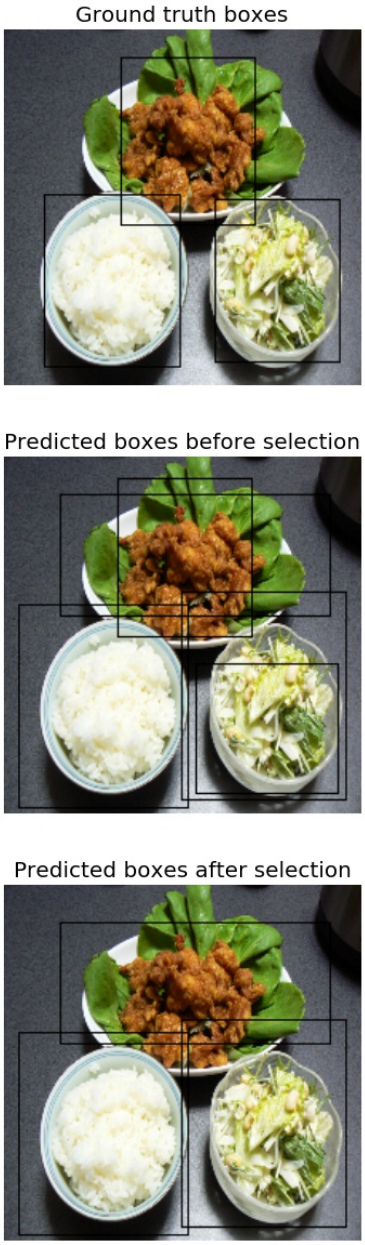

Figure 6.3: Segmentation process with: the bounding boxes (top), the candidate bounding boxes (middle), the proposed bounding boxes after overlapping suppression (bottom)

2. Root SIFT description

3. clustering using the k-means algorithm to obtain a 1000-word codebook.

Then, for each picture, we compute the histogram of occurrence counts of visual words. This descriptor is used with the SVM classifier and additive χ-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 [41]. It is available on gist [2] .

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 224×224 RGB picture as input.

The output of the layer just before the FC is used as a descriptor. Thus, each picture is described by a 4096 feature vectors.

6.4 Code

The code is freely available on Github[3].

I'm using python 3.5.2 and its scientific stack based on Scipy [42]:

- Numpy [43] for N-dimensional array
- Pandas [44] for the data structure
- Scikit-image [45] and OpenCV 3 [46] for some of the image processing algorithms

[2] `https://gist.github.com/ksimonyan/3785162f95cd2d5fee77/`
[3] `https://github.com/bnogaret/food_log`

- Scikit-learn [47] for most of the machine learning and Caffe [48] for the CNN
- Matplotlib [49] 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^{TM} low-latency interconnect.

7.2 Segmentation metrics

To measure the precision of the localization / segmentation algorithm, we use the metrics as defined in [50] [1].

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-

[1] Information on the evaluation system can be found at `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 [50], 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_Ps.

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).

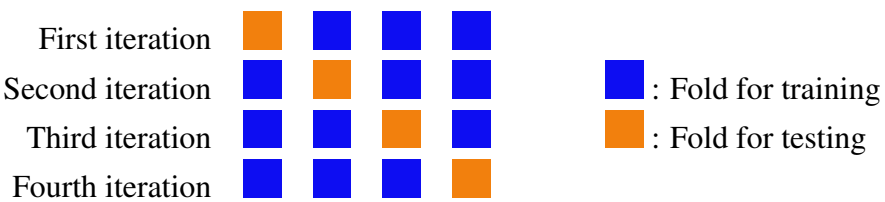

Figure 7.1: Illustration of 4-fold cross validation

7.4 Results

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

7.4.1 Food localisation

Metric (average)	My method	DCNN from [14]
Accuracy	**73 %**	60 %
Recall	**74 %**	80 %
Precision	**79 %**	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 [14], 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 literature, 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 threshold of 100 %).

7.4.2 Classification

In [14], the authors use a fine-tuned pre-trained Deep Neural Network and obtain 63 % accuracy on UEC FOOD-256.

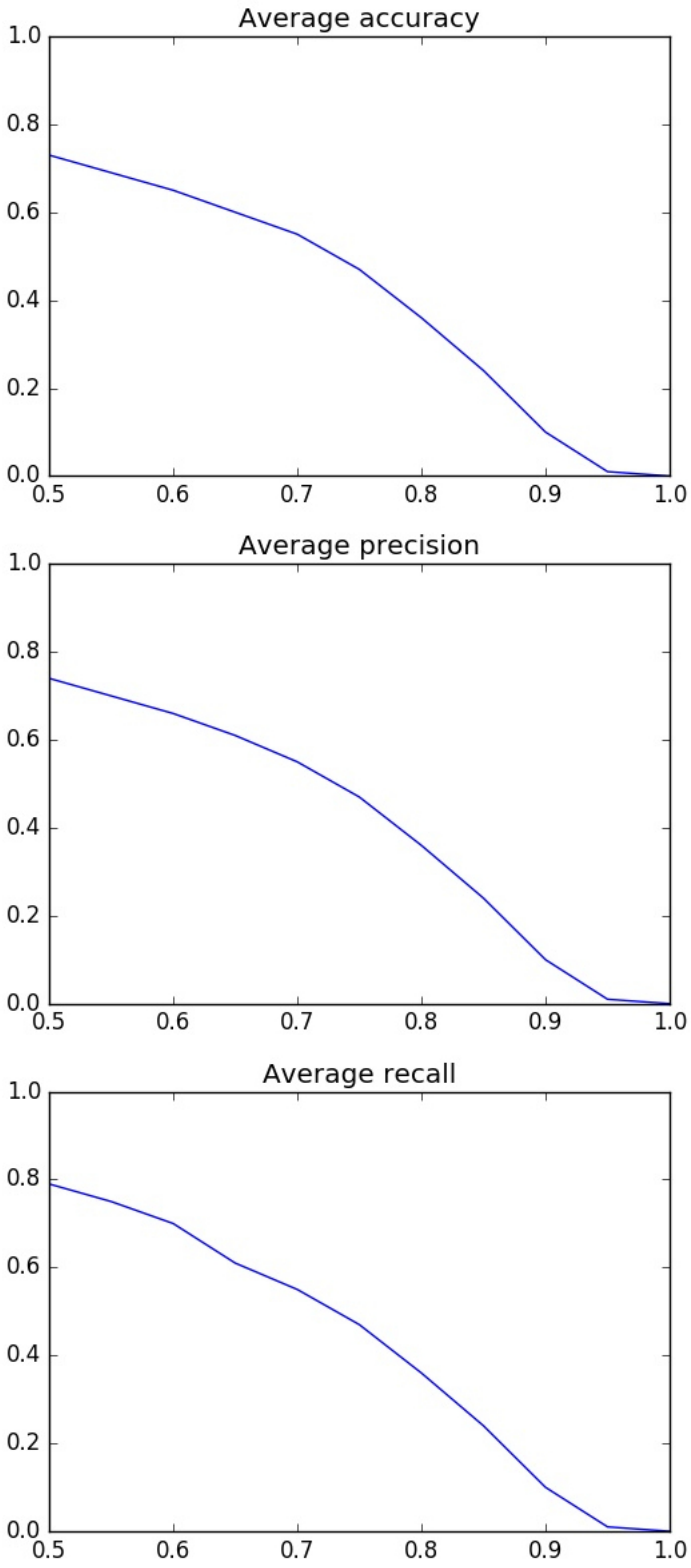

Figure 7.2: Curves of Accuracy over IoU (top), Precision over IoU (centre) and Recall over IoU (bottom)

Method	Average accuracy
CNN as descriptor + RF	**40 %**
BoW (1000 words)+ SVM with χ^2	10 %
LBP + CHM + Decision tree	5 %
LBP + CHM + SVM	11 %
LBP + CHM + RF	16 %
DCNN from [14]	63 %
DCNN from [12]	67 %

Table 7.2: Average classification accuracy result for UEC FOOD 256. CHM stands for colour histograms and moments

Process	My method	DCNN from [14]
Overall	**28 %**	37 %
Localisation	**74 %**	60 %
Classification	**38 %**	60 %

Table 7.3: Average accuracy result for UEC FOOD 256

In [12], the authors use a fine-tuned pre-trained Deep Neural Network and obtain 67 % accuracy on UEC FOOD-256.

7.4.3 Segmentation followed by classification

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

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 %)

Process	My method	DCNN from [10]	DCNN from [11]
Overall	**33 %**	-	-
Localisation	**67 %**	60 %	-
Classification	**50 %**	-	72 %

Table 7.4: Average accuracy result for UEC FOOD 100

As can been 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 colour that is relatively invariant to the condition
- tanmen is a soup containing noodle and various vegetables. Thus, it can occur in different colour, shape and size.
- there are numerous soups in the dataset and tanmen is often confused with them.

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

In this thesis, a localisation and classification method was proposed and it can be used to detect food items of a picture. The localisation process use a novel approach with a pre-trained convolutional neural network to detect salient objects and it currently outperforms the previous works.

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

Then, it can be added the estimation part that includes a calorie evaluation or a simplified version based on MyPyramid or MyPlate and an application to take pictures 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:

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

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

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

$$V = M$$

A.2 HSV to RGB

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

$$C = V \times S \qquad X = C \times (1 - |\frac{H}{60} \mod 2 - 1|) \qquad m = V - C$$

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

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

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.

[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.

[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] R. Thendral, A. Suhasini, and N. Senthil. “A comparative analysis of edge and color based segmentation for orange fruit recognition”. In: *International Conference on Communication and Signal Processing, ICCSP 2014 - Proceedings* (2014), pp. 463–466. DOI: 10.1109/ICCSP.2014.6949884. URL: http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=6949884.

[9] Minami Wazumi et al. “Auto-Recognition of Food Images Using SPIN Feature for Food-Log System”. In: *Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on* (2011), pp. 874–877. URL: http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=6316741.

[10] 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.

[11] Yoshiyuki Kawano and Keiji Yanai. “Food Image Recognition with Deep Convolutional Features”. In: *ACM International Joint Conference on Pervasive and Ubiq-*

uitous Computing (UbiComp) (2014), pp. 589–593. DOI: 10.1145/2638728.2641339. URL: http://dx.doi.org/10.1145/2638728.2641339.

[12] K Yanai and Y Kawano. "Food image recognition using deep convolutional network with pre-training and fine-tuning". In: *Multimedia Expo Workshops (ICMEW), 2015 IEEE International Conference on*. 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.

[13] 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.

[14] 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.

[15] Keigo Kitamura, Toshihiko Yamasaki, and Kiyoharu Aizawa. "Food log by analyzing food images". In: *ACM international conference on Multimedia* (2008), p. 999. DOI: 10.1145/1459359.1459548. URL: http://portal.acm.org/citation.cfm?doid=1459359.1459548.

[16] United States Department of Agriculture. *mypyramid.gov, steps to a healthier you*. 2005. URL: http://www.mypyramid.gov/.

[17] United States Department of Agriculture. *MyPlate*. 2005. URL: http://www.choosemyplate.gov/ (visited on 03/05/2016).

[18] Kiyoharu Aizawa et al. "Food balance estimation by using personal dietary tendencies in a multimedia food log". In: *IEEE Transactions on Multimedia* 15.8 (Dec.

2013), pp. 2176–2185. ISSN: 15209210. DOI: 10.1109/TMM.2013.2271474. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6548059.

[19] Hokuto Kagaya, Kiyoharu Aizawa, and Makoto Ogawa. "Food Detection and Recognition Using Convolutional Neural Network". In: *ACM Multimedia*. 2. 2014, pp. 1085–1088. ISBN: 9781450330633. DOI: 10.1145/2647868.2654970. URL: http://dl.acm.org/citation.cfm?doid=2647868.2654970.

[20] Mei-Yun Chen et al. "Automatic Chinese food identification and quantity estimation". In: *SIGGRAPH Asia* (2012), pp. 1–4. DOI: 10.1145/2407746.2407775. URL: http://dl.acm.org/citation.cfm?doid=2407746.2407775.

[21] Rana Almaghrabi et al. "A novel method for measuring nutrition intake based on food image". In: *2012 Ieee I2Mtc* (2012), pp. 366–370. ISSN: 1091-5281. DOI: 10.1109/I2MTC.2012.6229581. URL: http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=6229581.

[22] F Zhu et al. "The Use of Mobile Devices in Aiding Dietary Assessment and Evaluation". In: *IEEE Journal of Selected Topics in Signal Processing* 4.4 (2010), pp. 756–766. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2010.2051471.

[23] Fengqing Zhu et al. "Multiple hypotheses image segmentation and classification with application to dietary assessment". In: *IEEE Journal of Biomedical and Health Informatics* 19.1 (Jan. 2015), pp. 377–388. ISSN: 21682194. DOI: 10.1109/JBHI.2014.2304925. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6733271.

[24] 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.

[25] Yoshiyuki Kawano and Keiji Yanai. "FoodCam: A real-time food recognition system on a smartphone". In: *Multimedia Tools and Applications* (2014), pp. 5263–5287. ISSN: 13807501. DOI: 10.1007/s11042-014-2000-8.

[26] 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.

[27] 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.

[28] David G Lowe. "Distinctive image features from scale invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. ISSN: 0920-5691. DOI: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94. arXiv: 0112017 [cs]. URL: http://portal.acm.org/citation.cfm?id=996342.

[29] 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.

[30] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded up robust features". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 3951 LNCS. 2006, pp. 404–417. ISBN: 3540338322. DOI: 10.1007/11744023_32.

[31] 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.

[32] 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.

[33] 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.

[34] Škrjanec Marko. “Automatic fruit recognition using computer vision”. Mentor: Matej Kristan. Bsc thesis. Faculty of Computer and Information Science, University of Ljubljana, 2013.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] Jianming Zhang et al. "Unconstrained Salient Object Detection via Proposal Subset Optimization". In: *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)* (2016). URL: http://cs-people.bu.edu/jmzhang/sod.html.

[40] 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_%7DST0C%7B%5C&%7Dhl=ja%7B%5C&%7Doi=p.

[41] 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.

[42] 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/.

[43] 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.

[44] 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.

[45] 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.

[46] 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/.

[47] 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/.

[48] 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.

[49] 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.

[50] 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.