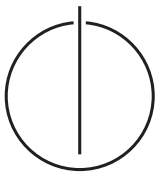
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Evaluation and Implementation of Computer Vision Concepts for Image Recognition of Food Items Using Deep Learning and SVMs

Felix Schober



DEPARTMENT OF INFORMATICS

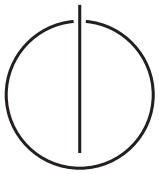
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Evaluation and Implementation of Computer Vision Concepts for Image Recognition of Food Items Using Deep Learning and SVMs

Evaluierung und Implementierung von Computer Vision Konzepten für Bilderkennung von Nahrungsmitteln mit Hilfe von Deep Learning und SVMs

Author:	Felix Schober
Supervisor:	PD Dr. Georg Groh
Advisor:	Hanna Schäfer, M.Sc.
Submission Date:	15.04.2016



I confirm that this bachelor's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 15.04.2016 Felix Schober

Acknowledgments

I would like to thank all the people that helped and contributed to this thesis. I want to express my gratitude to Hanna Schäfer and Georg Groh. They gave me the opportunity to work on this amazing topic and write my bachelor's thesis at the Chair of Applied Informatics. The weekly meetings gave me the motivation, feedback and advice needed to complete this thesis. Thanks for providing me with two servers for my experiments and all the fulfilled requests that I constantly issued.

Abstract

As obesity becomes more and more of a problem in developed countries, food logging is frequently used to help overweight people to balance their energy intake. Unfortunately, food logging is a tedious and inaccurate process. Computer vision and machine learning can help the user with this process. By taking an image of the meal, algorithms are able to make automated food intake assessments by detecting food items and their size. The goal of this thesis is the evaluation and implementation of a proof of concept application that can be used to facilitate and extend future food logging applications.
For the task of food classification seven approaches were evaluated including feature classifiers like SIFT and SURF and convolutional neural networks. To enable the classification, segmentation, training and evaluation of classifiers, an extensive application was implemented using Python. The application supports data preprocessing and is designed so that it can be extended with additional image recognition concepts.
With the aforementioned algorithms it was shown that algorithms are able to achieve a classification accuracy of 75% on 50 different food item classes if the algorithm suggests five possible candidates.

Zusammenfassung

Fettleibigkeit wird zunehmend zu einem Problem in Industrieländern. Durch das Führen eines Ernährungstagebuchs kann übergewichtigen Personen dabei geholfen werden, ihren Energiehaushalt auszugleichen. Ernährungstagebücher sind jedoch oft ungenau und das detaillierte Führen eines solchen Tagebuchs ist aufwendig. Durch Computer Vision und Machine Learning Konzepte kann dieser Prozess erleichtert werden. Nutzer können ein Bild ihrer Mahlzeit aufnehmen, welches durch Algorithmen analysiert wird. Diese können dann automatisiert den Energiewert berechnen und in das Tagebuch eintragen. Das Ziel dieser Arbeit ist die Evaluierung und Implementierung eines Machbarkeitsnachweises, welcher für zukünftige Ernährungstagebücher benutzt und erweitert werden kann.
Für die Klassifizierung der Nahrungsmittel wurden sieben Ansätze getestet. Unter anderem wurden SIFT, SURF und Convolutional Neural Networks geprüft. Dafür entstand ein umfassendes Programm, welches Klassifizierung, Segmentierung und die Evaluierung von verschiedenen Bilderkennungsalgorithmen ermöglicht. Das Programm unterstützt zudem diverse Algorithmen zur Datenvorverarbeitung und ist so konzipiert, dass es einfach mit weiteren Konzepten zur Bilderkennung erweitert werden kann.
Mit den genannten Algorithmen konnte gezeigt werden, dass sich eine Genauigkeit von 75% erreichen lässt, wenn der Algorithmus die Möglichkeit erhält, eine Liste mit 5 Vorschlägen als Antwort zu geben.

Contents

1 Introduction

1.1 Motivation

Obesity is a huge, growing but preventable problem among children, adolescents and adults in developed countries. Obesity can cause many major diseases like cardiovascular diseases, diabetes or even cancer [64, 85].
One method to prevent or at least reduce obesity is food logging because it can help to balance the energy intake. Many publications have shown that the action of food logging can be correlated to weight loss [35, 16, 78]. However, the accuracy of food intake in food diaries is prone to underreporting over all age groups [57]. Especially underreporting is a problem if the goal is to loose weight by balancing the energy intake and activity level. Some approaches relied on expert nutritionists to analyze food images manually or used crowd sourcing to extract nutritional information. Both methods are either costly, inaccurate or slow [62].
It has been proven that taking pictures of food raises a better awareness of the nutritional values of a meal compared to written entries [86] and is also a highly motivating factor for logging food [46]. Despite the raising trend of sharing pictures of food on social media it is still tedious to actually take pictures of every meal and annotate these images with information about the consumed food items manually.
A possible solution is to extract the nutritional information automatically by using computer vision algorithms and machine learning concepts. In the last few years image recognition and classification made a huge leap forward due to advances in deep learning and increasing computational resources [79]. Computer vision algorithms have come very close to human classification performance and can in some places even surpass the human brain.
Food logging is a perfect example of useful computer vision application because it can potentially increase the accuracy of meal logging and provide a much faster and less cumbersome way of logging food. It allows users to add multiple meal images together so that the diary does not need be opened at every meal. In addition, image recognition enables applications to provide information beyond the knowledge of the

user and makes it possible to track the intake of fat or sugar and even warn the user of possible harmful allergens.
This thesis provides the foundation for future food logging applications by facilitating the training, segmentation and classification of food images using modern computer vision algorithms and machine learning models.

1.2 Challenges of Food Recognition

Image recognition has advanced tremendously over the last years. The latest winner of the ImageNet challenge 2015 [75] achieved a top-5 localization error of only 8.9% on 1000 different classes with a 150 layer deep neural network [33]. Nevertheless, there is still not a single comprehensive, publicly available food classification tool. The best approach so far only achieved a mere 79% accuracy on a 101 class food dataset [62].
The following section will provide reasons why food recognition is indeed much more challenging than normal object recognition.
The information a useful computer vision aided food intake diary has to extract can be structured into two categories which both pose a challenge for any potential application:

Classification

Quantity of Food Types

The goal of the classification task is to predict the food category as accurately as possible with a preferably complete set of image categories which poses the first problem. Building a classifier for all plane types is relatively easy as the number of possible plane types is limited. The number of different food items, however, is nearly unlimited. There are so many variations of food that it is not possible to train a classifier on all items but for accurate nutrition information extraction it is extremely important to be able to differentiate between different sorts of food items. The nutritional values of a slice of brown bread are different than those of a slice of white bread.

Food Parameters

Another problem for food image classification is that food can be parametrized with different toppings, different kind of sauces or varying methods of preparation. It is very hard to take these food parameters into account because of the huge variety of

different parameters. Not only can they completely change the appearance of the food item they also have a huge impact on the nutritions. Salads can be very healthy but combining a salad with an unhealthy topping entirely changes the healthiness of the dish.

Intraclass Variance

The major problem for food classification, however, is the huge intra class variance. Even the same kind of food can look completely different depending on preparation, ingredients and presentation. Normal objects in image recognition tasks mostly appear in some kind of standardized form. Cars for example always include some common features that are easy to distinguish. Figure 1.1 shows that even a food type that is normally very similar in appearance can have many visually different appearances.

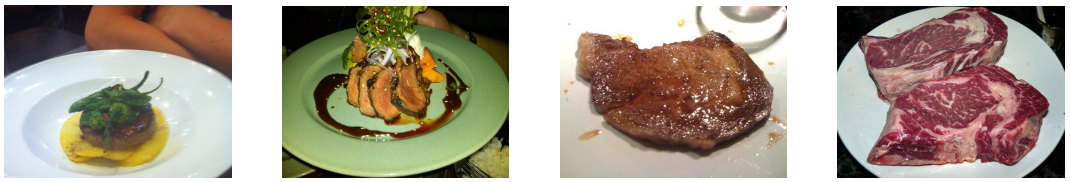


Figure 1.1: Examples for intra class variance. Every image shows a "steak" but the visual appearance varies greatly.

Interclass Variance

There is a tradeoff between the granularity of a food classifier and its accuracy for the nutrition problem. On one side, there are problems with items that look very different, but are not. On the other side, there are food items which tend to look almost the same, but are different. It is difficult to find a classifier that is able to recognize different preparations of a steak as a "steak" on the one side but is also able to determine the difference between a hamburger and a cheeseburger because the latter has more calories.

Unobservable Details

Even if a future system might solve all aforementioned problems there is still a problem even the most sophisticated image recognition systems can not solve alone. It is not

possible to analyze parts of the food that can not be seen. Even expert nutritionists are not able to determine from an image alone what sort of fat or sugar was used for cooking and if this food item contains some kind of stuffing that changes the whole nutritional value. Systems might leverage context information in the future to approximate the "hidden values" by taking additional information like position, time or past preference into account. It is also possible that in coming years smart devices will have build in spectrometers that can be used to analyze the chemical composition of the food [51].

Quantity Estimation

Unfortunately, a successful classification is not sufficient enough to deduct accurate nutritional information. It is also important to extract some kind of quantity information. For this, the image has to be segmented into different food parts because each food item has different nutritional values. Image segmentation alone is a difficult problem. Food image segmentation can be even harder because food items often lack clearly distinctive parts. A normal landscape image is easily separated into different parts but food items are often similar in color and share a common edge because the contrast between parts is usually very low.
In addition, food items are often obstructed from each other and may sometimes not even be visible at all like contents of a soup or food scalloped with cheese. But even with a perfectly separated image it is still difficult to get actual quantitative measurements about the items. Meal size area estimation is possible and was implemented for this thesis but for most foods (other than pizza) the area alone is not sufficient inasmuch as food usually has some kind of 3-dimensional shape. However, the approximation of volume and depth from single images is very challenging and not nearly as advanced as image recognition is. Models that are used to estimate depth and volume are trained on images of indoor areas [28] and not on small objects where absolute error is much more significant than on large objects. Meyers et al. incorporated such volume estimation and achieved a mean error that is still to high for real world scenarios [62].

1.3 Outline

This thesis is structured into 7 chapters. Chapter 2 outlines previous works on food recognition including information about food datasets, existing systems and common algorithms which are then further explained in chapter 3. This chapter includes common

concepts of machine learning and computer vision. Chapter 4 provides information about data preprocessing steps, algorithmic variations and the experimental setup. Chapter 5 describes the implementation and architecture of the proposed application. Results are then accumulated and discussed in chapter 6. Lastly, the results are summarized in chapter 7 and advice for future work is given.

2 Related Work

Since the introduction of the smartphone, taking images of food has become much more appealing for many people. There is an increasing trend to share images of food on social media. Since people get used to taking pictures of their food, images have become a much more interesting approach for data collection. Following this trend there is an growing number of publications in the field of food recognition. The goal of this chapter is to provide an overview of the key aspects of this field such as available datasets, common algorithms and existing systems.

2.1 Datasets

Comprehensive datasets are an important requirement for any modern image recognition task. For non-food objects like planes, animals or plants there are many huge, high-quality datasets with a sufficient amount of data such as ImageNet [75] or CIFAR-10 [52]. Since computer vision algorithms for food classification have to deal with more intra-class-variance (see Section 1.2) datasets have to be sufficiently big to be able to train accurate models. However, most of the existing food recognition datasets tend to be to small, have to much noise (false labels) or contain only images in controlled environments. Another problem that datasets are not able to cover so far is the high number of different classes for food. So even if a dataset covers 100 various types of meals this is still only a small amount compared to the number of actual food-recipes. The following section and table 2.1 will give a brief overview and comparison of datasets usable for food recognition.

2.1.1 Pittsburgh Fast Food Image Dataset

In 2009 Chen et al. released the first publicly available[1] food dataset also known as the Pittsburgh Fast Food Image Dataset (PFID) [20]. They collected food from eleven fast

[1] `http://pfid.rit.albany.edu/index.php`, accessed 2016-03-02

Name	# Classes	# Images	Source	Comments
ETHZ-Food-101	101	101000	[13]	
UPMC Food-101	101	101000	[54]	
Food201-MultiLabel	201	47374	[62]	to be released
UEC Food 256	256	31651	[41]	
Food201-Segmented	201	12625	[62]	to be released
UEC Food 100	100	9060	[61]	
50-data	50	5000	[21]	
PFID	7	4545	[20]	101 different foods
Menu-Match	41	646	[6]	includes calorie count

Table 2.1: Comparison of publicly available datasets for food classification.

food chains popular in the United States (US), then selected 101 different foods from these restaurants and took 4,545 still images in both controlled laboratory conditions and in the restaurants itself. Each restaurant was visited at least three times on different dates to get varying instances of the same food. For the still images in the lab they took six pictures of each item on a rotating turntable with each picture being 60 degrees apart. The images in the restaurant were taken with and without wrapper and from all four sides resulting in 8 images in the restaurant setting and 6 in the controlled environment per food and instance. In addition, they also took 606 stereo images (one or two for each food), a 360 degree video of the item on the turntable and 27 videos of people eating the food. Since their recognition rate for the 101 classes was very low they combined the 101 different items into seven more abstract categories as seen in figure 2.1. The PFID focuses on fast food because it is highly standardised (the food looks similar even across different restaurants) and a key factor for obesity. For these reasons many publications use this dataset to benchmark their models.

2.1.2 UEC-Food-100 and UEC-Food-256

UEC-Food-100 [61] and UEC-Food-256 [41] are two datasets from the University of Electro Communications (UEC) in Tokyo from 2012 and 2015 which were used to train "DeepFoodCam", a mobile standalone app for food classification.
The UEC-Food-100[2] dataset comprises of 100 categories of mostly Asian cuisine. Each

[2] `http://foodcam.mobi/dataset100.html`, accessed 2016-03-02

(a) sandwiches, subs, wraps (b) meat (c) salads (d) bagels (e) doughnuts, desserts, snacks (f) pizza (g) misc food like soup and Mexican food

Figure 2.1: Examples for food items for each of the seven suggested categories in the PFID. Source: Chen et al. [20]

class contains about 100 images manually gathered from the web. Furthermore there are also bounding box information for all images. In the context of this dataset bounding boxes are very helpful for training because it is quite common in Asian countries to have several different plates of foods as shown in figure 2.2d. The rice is only a very small part of the image and removing the other plates does significantly improve recognition performance.
With the UEC-Food-256[3] dataset Kawano et al. increased the number of categories from 100 to 256. To collect the new data from web sources they used their original UEC-Food-100 dataset to train Support Vector Machines (SVMs) that calculate "foodness", a value that measures if a picture is a food item or not. Images with a high "foodness" might be food but they might not correspond to any of the new categories of UEC-Food-256. To filter these images they applied transfer learning by selecting images with a high "foodness" value and visually similar classes from UEC-Food-100. Lastly, Kawano et al. used Amazon Mechanical Turk (ATM) for crowd sourcing to evaluate the results, remove remaining noise and get bounding boxes for the food items.

2.1.3 Food-101

Food-101 is the largest food dataset so far. The original Food-101[4] dataset was built by Bossard et al. in 2014 to create the first food dataset with more than 100 images per class [13]. They composed a list of the 101 most common keywords on Foodspotting.com. For each keyword they obtained 1000 images from the site. Figure 2.3 gives some examples of the images in the dataset. Contrary to other food datasets like UEC Food

[3] `http://foodcam.mobi/dataset256.html`, accessed 2016-03-02
[4] `http://www.vision.ee.ethz.ch/datasets/food-101/`, accessed 2016-03-02

(a) burger (b) croquette (c) dipping noodles (d) rice (e) shrimp with chill source (f) simmered pork (g) toast

Figure 2.2: A sample of the food items and categories of the UEC-Food-100 dataset. Source Matsuda et al. [61]

256, they did not remove any noise from the dataset which results in many false positive labels as figure 2.4 shows.

(a) French fries (b) fried rice (c) hamburger (d) pizza (e) sashimi (f) spagetti (g) steak

Figure 2.3: A sample of the food items and categories of the ETHZ-Food-101 dataset. Source: Bossard et al. [13]

A year later in 2015 Kumar et al. released the UPMC-Food-101[5] dataset which contains the exact same categories as the first Food-101 dataset from the Eidgenössische Technische Hochschule Zürich (ETHZ). This makes it possible to combine these twin datasets into one big dataset since they contain images from different sources. The data of UPMC-Food-101 originates from Google Image search followed by the keyword "recipe" to reduce noise because some words can be associated with non-food-items (eg. Hamburger-menu-icon, escargots as living snails). However, despite the keyword "recipe" UPMC-Food-101 contains even more noise than ETHZ-Food-101. Additionally, for 100,000 images, they included the Hypertext Markup Language (HTML) of the pages the images were embedded in [54].

[5] `http://visiir.lip6.fr/explore`, accessed 2016-03-02

(a) French fries (b) fried rice (c) hamburger (d) pizza (e) sashimi (f) spagetti (g) steak

Figure 2.4: Examples for noise in the Food-101 datasets. Source: Kumar et al. [54] and Bossard et al. [13]

2.1.4 Food-201

With the upcoming Food201 datasets Meyers et al. are trying to address several shortcomings of the Food-101 datasets [62]. Food201-MultiLabel will be a subset of the original ETHZ-Food-101 dataset containing only 50% of the images. They took 47,374 of those images to ATM to label the contents of the images individually since a major problem of Food-101 is the lack of multiple labels for images as demonstrated in figure 2.3c. A picture could contain some French fries and a burger but might be only labeled in the category of Burger. Even if a classifier correctly predicts "french fries" as a possible solution for the problem the result will still be marked as incorrect. The new dataset now contains 201 categories in contrast to the 101 previously with a mean of 1.9 different labels per image.
Food201-Segmented will be another smaller subset of Food201-MultiLabel with 12,625 images and pixelwise labeling of the category. This dataset will enable a better training of segmentation oriented classifiers.
At this point in time the Food201 datasets are not released[6].

2.1.5 ImageNet

ImageNet is the biggest general image recognition dataset to this date containing 21,841 classes and 14,197,122 images [75]. Considering that a sizeable amount of the dataset consists of food, ImageNet could be counted as a food image recognition dataset. However, the effort one has to spend to actually get only the food data is quite high. Categories in ImageNet are organized as Synonym Sets (synsets). Synsets can have different children called hyponyms. The WordNet ID (wnid) for the food

[6] `https://storage.googleapis.com/food201/food201.zip`, accessed 2016-03-02

synset is n00021265. By using an Application Programming Interface (API) call[7] and the food wnid, ImageNet returns a list of all food-children of a synsets (hyponyms). With these wnids the image Uniform Resource Locators (URLs) can be obtained by another API call. ImageNet contains 1527 food categories, yet, some of these categories only contain a small amount of images while others include many images making the dataset imbalanced which makes it harder to train certain classifiers like SVMs [3].

2.1.6 Menu-Match

Despite its small size of only 646 images distributed over 41 classes, Menu-Match is an important dataset in the domain of computer vision food calorie estimation because it is the only dataset available in public[8] which includes professional calorie estimations for each food [6]. Beijbom et al. collected meals from three restaurants and made images with seven different capture devices (six smartphones and one point and shoot camera) in a real world scenario without fixed camera angles or distances.

2.1.7 Other Smaller Food Datasets

In addition to the mentioned food datasets there are several other notable datasets in the extended domain of food classification. Chen et al. published a small Chinese food dataset with 50 classes and 100 images per class [21]. They did not give a name to the dataset so this dataset is refereed to as "50-data" is this thesis. Moreover, there are three other food datasets which belong to the extended set of food classification datasets.

Technology Assisted Dietary Assessment (TADA) is a project of the Purdue University. For their project they developed an image database called I-TADA that includes images as well as meta-data [45].

FoodCast Research Image Database (FRIDa) is a food image database with 877 images belonging to eight classes including natural-food (e.g. fruit), processed food (e.g. french fries), rotten-food, natural-non-food items (e.g. pinecone), and classes that contain non food items [31].

[7] http://www.image-net.org/api/text/wordnet.structure.hyponym?wnid=wnid&full=1

[8] http://research.microsoft.com/en-us/um/redmond/projects/menumatch/data/, accessed 2016-03-03

Lastly, "Food.pics" is a dataset of 568 food items that was developed to asses the relevance of visual food cues for the human brain [9].

2.2 Algorithms

Although food is much harder to recognize and classify than common object recognition, it is considered as a computer vision problem so most of the common computer vision and machine learning algorithms can be applied. Unfortunately, it is not easy to compare algorithms and methods of different papers based on their results. Different authors seldom use the same benchmarks and often even use their own closed source datasets which makes even basic comparison impossible. In addition, these algorithms can often be parametrized and not all papers publish all parameters so even identical algorithms can achieve different accuracies. As a result, this section will expose the algorithms and methods but not compare them. Figure 2.5 gives an overview of the different techniques used in 43 food recognition publications.
The most used approach for food classification in literature is still the use of various feature descriptors transformed into fixed length vectors and SVM classification. Out of 43 publications in the domain of food recognition, 72% of these papers used a support vector machine.

2.2.1 Color Features

In contrast to normal object recognition color features such as histograms play a much more important role for food recognition because color is one of the main characteristics of food. The color of a car does not matter but it makes a huge difference if the pasta sauce is red or white.
The use of color features for food classification can be divided into three approaches:

1. Histograms
2. First- and second moment statistical values
3. Color descriptors

Since using approaches (1) or (2) means neglecting any spatial relations most publications sub divide the images into 2x2 [38, 34, 43] or 4x4 blocks [21] and apply the algorithms separately to keep basic spatial relations.

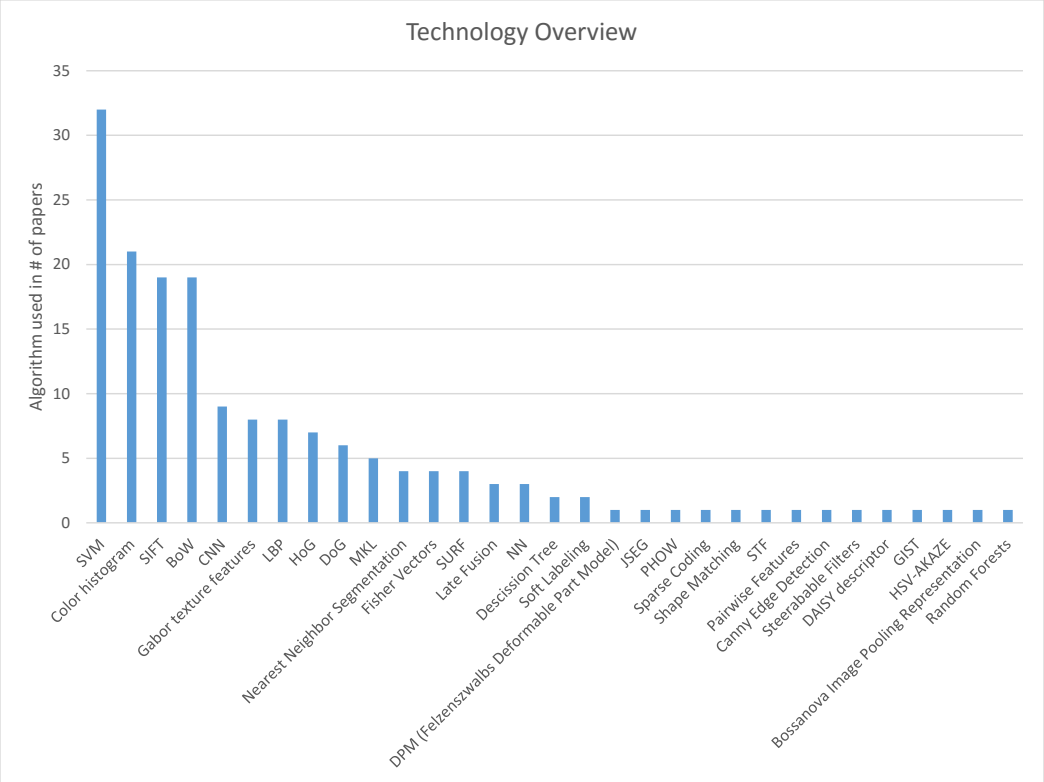


Figure 2.5: Number of occurrences of different computer vision and machine learning algorithms used in 43 papers in the domain of food recognition.

(1) Histograms

The most common color histogram feature is a 64 bin Red Green Blue (RGB) histogram [38, 34, 20]. Prabu tested five different histograms based on the color space and normalization but didn't provide any results in his paper [70].

(2) First- and second moment statistical values

There are several possible values that can be utilized to create a meaningful feature vector such as mean of pixel values in different color spaces [49, 46, 43, 11] or variance / deviation [49, 43, 11]. Bosch et al. also used the most dominant color in a block as an additional value [11].

(3) Color descriptors

The performance of primitive color features like (1) and (2) can suffer greatly from changes in the environment like illumination changes. Therefore, recent publications used colour descriptors instead of histograms or statistical measurements [6, 8].

2.2.2 Feature Detectors and Descriptors

Following color features, Scale-invariant Feature Transform (SIFT) [60] is the most used feature detector and descriptor. Out of 43 publications, 21 used SIFT to classify food [50, 38, 83, 20, 48, 87, 34, 21, 47, 30, 63, 11, 39, 54, 70, 6, 8, 71]. Unfortunately, there is no paper with a comparison between different numbers of SIFT keypoints or parameters for food recognition.
SIFT is often used in combination with a Bag of Words (BoW) [38, 20, 48, 87, 34, 21, 61, 47, 63, 11, 39, 54, 6, 71]. For BoWs, according to Joutou et al., a higher BoW codebook dimension of 2000 is better than a lower dimension of 1000 [38].
Other popular image features include Histogram of Oriented Gradients (HOG) [72, 34, 61, 47, 42, 40, 6] and Gabor textures [38, 14, 34, 46, 61, 40, 21, 69, 11] but in comparison to other image features they do not achieve high accuracies [47, 6]. Hoashi et al. compared several HOG classifiers to Gabor filters and BoWs. Even a BoW with random keypoint sampling achieved a better result (30.35%) than the best HOG- or Gabor classifiers (21.84% and 25.35%) [34].
Publications also made use of Local Binary Patterns (LBP) [87, 21, 22, 70, 6]. Farinella et al. tested Pairwise Rotation Invariant Co-Occurrence Local Binary Pattern (PRICoLBP), a slightly different LBP algorithm which outperformed SIFT [30] and Nguyen et al. compared LBP and Non Redundant Local binary patterns (NRLBP) with NRLBP achieving marginally better results than normal LBP.

2.2.3 Convolutional Neural Networks

Recent approaches showed that deep learning can produce much higher accuracies than the classical feature and SVM combinations.
Kim was one of the first to use a Convolutional Neural Network (CNN) for food recognition but only achieved 22.38% of accuracy on the UEC-Food-100 dataset due to the lack of images because UEC-Food-100 only contains about 100 images per class [47]. Therefore, his proposed Pyramid Histogram of Visual Words (PHOW) approach achieved a much higher accuracy with 32.12%.

Kawano et al. used a slightly different approach on UEC-Food-100 [42]. They took a modified 8-layer-CNN which was pretrained on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 1000 class dataset. They removed the last layer (layer 8) and utilized the output of layer 7 as a feature vector which was then used in combination with Fisher vectors to classify the image. With this CNN they outperformed every other result on UEC-Food-100 with an accuracy of 72.26%. Their second CNN got even better results. It was a CNN pretrained on the ILSVRC 2000 class dataset which they further enhanced by using 1000 food-synsets from ImageNet. This more food related CNN could achieve an accuracy of 77.35%.
Kumar et al. used a very similar approach in their publication [54]. They also used the output of the layer before the last layer as a feature vector and they also used a pretrained net.
Bossard et al. trained the CNN which Krizhevsky originally proposed in [53] (AlexNet). They used this net as a baseline for their new Food-101 dataset and got a result of 56.40% [13].
Kagaya et al. tested different architectures of small CNNs against their own closed source "FoodLog" dataset with 10 classes [39]. With 73.70% they achieved the best result by using a two layer network with small 5x5 kernels.
Christodoulidis and Anthimopoulos also compared different smaller CNN architectures on their own dataset with 7 different classes and got the best results (71.79%) with a 6-layer CNN that used dropout [22].
Shimoda and Yanai used a pretrained AlexNet CNN to segment images of the UEC-Food100 dataset into different components [76].
Myers et al. recently released a paper for their "Im2Calories" system which almost solely relies on CNNs [62]. Like other approaches they utilize CNNs which are trained on ImageNet. In this case they took a GoogLeNet CNN, replaced the last layer with a 101-way softmax layer and trained the net on Food-101. The reported 79% accuracy is much better than the previous benchmark for Food-101 with 50.76%. They also used CNNs for automatic food segmentation and volume estimation. For the segmentation task they used "DeepLab", a CNN in combination with a Conditional Random Field (CRF) from [19]. This method and additional context information accomplished a top-1 accuracy of 76% on their Food201-Segmented dataset. For the volume estimation they used the network from [28] and got an average relative error of 0.18 meters.

2.3 Existing Systems

The following section includes three existing systems. One of them is already available estimating food groups. There are also two other systems that are not publicly available but include calorie estimation.

2.3.1 FoodLog

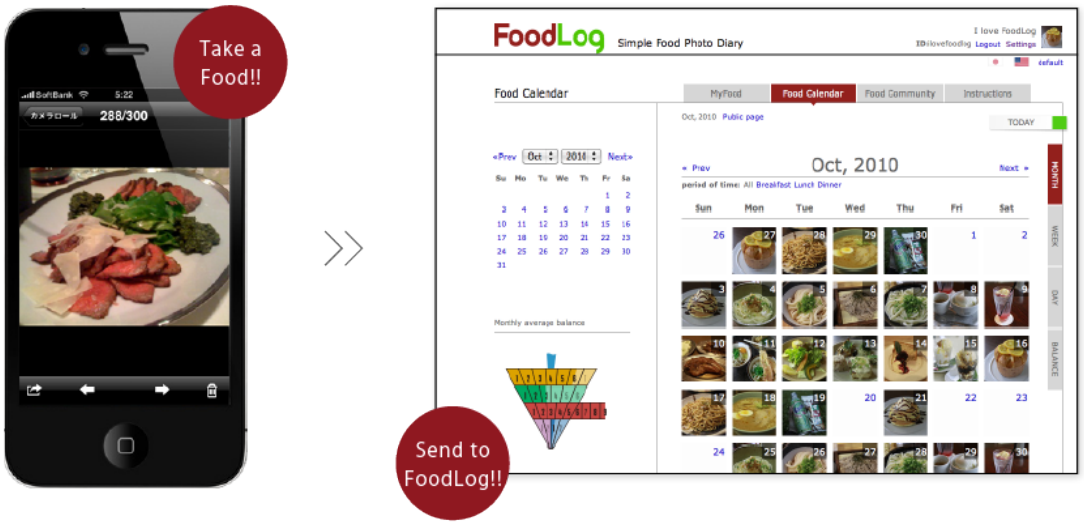


Figure 2.6: Image of the FoodLog system taken from the website. Source: [37].

FoodLog[9] is a online webservice by researches from the University of Tokyo that has been online since march 2009 [49, 50, 48]. This service allows users to record their daily food intake with photos to support diets and weight loss. FoodLog analyzes the submitted photos and verifies that the uploaded image contains food. If an image contains food the service calculates the "food balance" which is the share of five food groups (Grain, Vegetable, Meat + Beans, Dairy Products, Fruit). FoodLog analyzes the images using both global features like Hough transforms and color statistics and local features like SIFT. Their current best result for their food balance classifier is 43% [48].

[9] `http://www.foodlog.jp/en/`, accessed 2016-03-07

2.3.2 Menu-Match

"Menue-Match" by Beijbom et al. [6] is a collaboration between Microsoft and the University of California . They proposed a system that works exclusively on restaurant menu items and tries to estimate the amount of calories in a meal. By restricting the application to restaurants they can utilize the phone's Global Positioning System (GPS) sensor to get the current position and restaurant. Knowing the restaurant's menu they can then use the menu as additional context information. The system uses five image features and a one-vs-rest linear SVM for each feature and each food class resulting in a 205 dimensional feature vector which is then trained for each image on other SVMs. This joint-feature-late-fusion approach achieved a mean absolute calorie estimation error of 232 calories per meal. At this point in time there are no apparent plans to release Menu-Match to the public.

2.3.3 Im2Calories

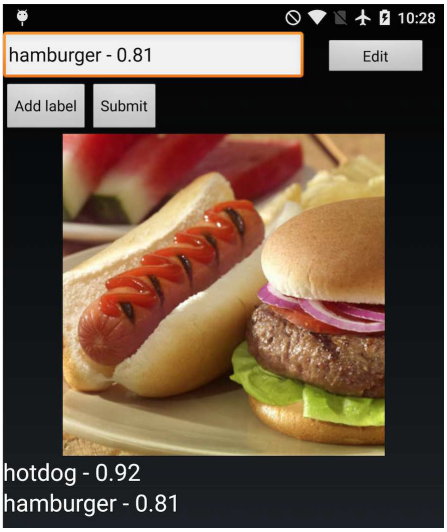


Figure 2.7: Screenshot of the Image2Calories mobile Android. Source: Myers et al. [62]

Im2Calories by Myers et al. [62] is Google's approach for a computer vision aided food calorie estimator. The proposed system is similar to Menue-Match. It also tries to estimate nutritional information from food and they also set their focus on restaurant meals. However, Im2Calories tries to take it a step further. They extend the number of

restaurants from 3 to 23 and instead of 41 food items they try to recognize 2517 food items. By using a deep learning approach instead of SVMs (more details in Section 2.2) they could improve the mean absolute calorie estimation error from 232.0 calories to 152.95. They created an Android mobile app which can classify images without an internet connection by storing the CNN models on the device itself. A screenshot of the app can be seen in figure 2.7. Google plans to release Im2Calories but has given no release date as of now.

3 Theory

3.1 SVM

A Support vector machine (SVM) is a machine learning algorithm which tries to separate binary-class training data. The SVM achieves this by searching for a hyperplane or a set of hyperplanes to separate the points of training data in a multidimensional space. There might be several possible hyperplanes that sufficiently separate the data but a SVM searches for the hyperplane with the biggest perpendicular distance between the hyperplane and the borders of both classes that should be separated. The space between the hyperplane and the closest of the training data points is defined as the margin. If the algorithm finds the maximal margin it has also found the optimal hyperplane. See figure 3.1 for an example of possible hyperplane separations.

Formal Definition

Since a SVM is a binary classifier a data point x has to either belong into class A or class B. Let the training dataset of n points be represented as $x_0, \ldots, x_n$ and the target values be $y_0, \ldots, y_n \in -1, +1$. The values of y_n should be

$$y_i = \begin{cases} +1 & \text{if } x_i \in \text{class } A \\ -1 & \text{if } x_i \in \text{class } B \end{cases}. \tag{3.1}$$

The optimal hyperplane should separate all vectors x_i with a value of $y_i = 1$ (class A) and those with a value of $y_i = -1$ (class B) so that the distance between those two groups is maximal [80]. Let the hyperplane be

$$D(x) = W \bullet x + b \tag{3.2}$$

where W is a vector normal to the hyperplane (see figure 3.2) and b is a bias term [12]. $D(x)$ has to be calculated so that

$$x \in \begin{cases} \text{class } A & \text{if } D(x) > 0 \\ \text{class } B & \text{if } D(x) < 0 \end{cases} \tag{3.3}$$

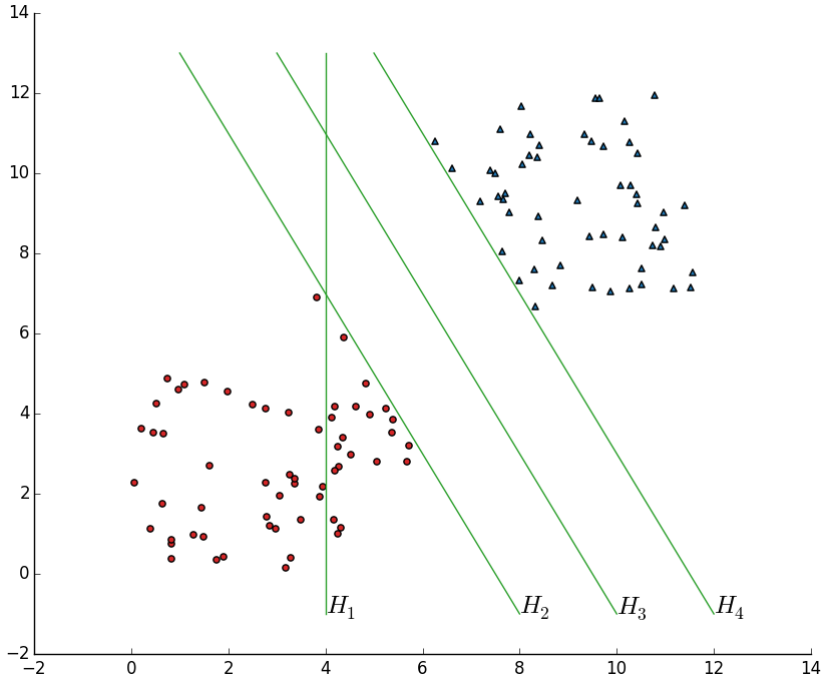


Figure 3.1: Separation of two classes by hyperplanes in 2 dimensional space. H_1 does not separate, H_2 and H_4 separate but the margin is very slim, H_3 separates with a much better margin.

is true. The distance between a point X and the hyperplane $D(x)$ is given as

$$\frac{D(X)}{\|W\|} \tag{3.4}$$

which is illustrated in figure 3.2 [80]. Assuming that the data is linear separable (as in figure 3.1) we can then select two hyperplanes $D(x) > 0$ and $D(x) < 0$ so that there are no points in between them and try to maximize the distance between those two hyperplanes (in figure 3.1 H_2 and H_4) which is then given as

$$\frac{2}{\|W\|}. \tag{3.5}$$

To maximize the distance we must therefore minimize $\|W\|$ but with the constraint (3.6) that no data gets between the two hyperplanes (hard-margin classification). Both

constraints

$$\begin{aligned} x_i \bullet w + b \geq +1 \quad & \text{if } y_i = +1 \\ x_i \bullet w + b \leq -1 \quad & \text{if } y_i = -1 \end{aligned} \tag{3.6}$$

can be combined to $y_i(x_i \bullet w + b) \geq 1 \quad \forall i, i = 1 \ldots n$. The solution for a maximum margin can therefore be found by solving

$$\begin{aligned} \arg\min_{w,b} \left\{ \frac{1}{2} \|W\|^2 \right\} \text{ subject to} \\ y_i(x_i \bullet w + b) - 1 \geq 0 \quad \forall i, i = 1 \ldots n \end{aligned} \tag{3.7}$$

with the Lagrangian multiplier method [80].
If the data is not linear separable we have to relax the constraints from 3.6 and allow some points in the margin (soft-margin-classification).
Classification of unknown data is now simply done through a calculation of the decision function $D(x)$

$$\text{prediction} \begin{cases} x \in \text{class } A & \text{if } D(x) > 0 \\ x \in \text{class } B & \text{if otherwise} \end{cases} \tag{3.8}$$

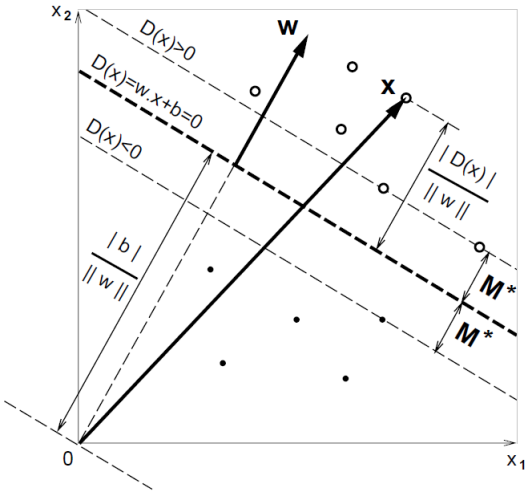
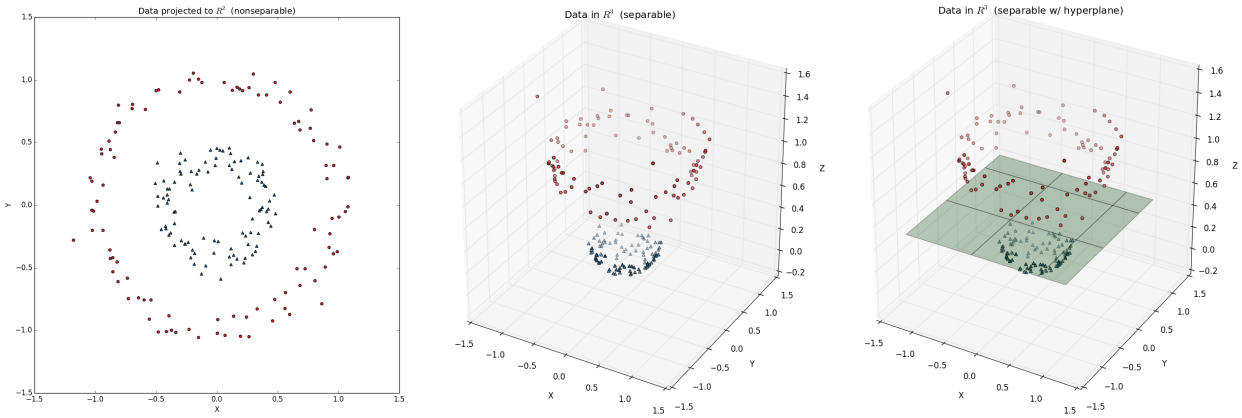


Figure 3.2: Visualization of optimal margin and hyperplane in 2 dimensional space. The distance of Point X to $D(x)$ is $\frac{D(X)}{\|W\|}$. Source Bosner et al. [12].



(a) Data not separable (b) Data separable (c) Data separable with hyperplane

Figure 3.3: Illustration of Kernel trick. Data in (a) is not separable but if the data gets mapped into a higher dimensional space (b) a linear hyperplane can be fitted (c). Figures plotted with code adapted from `http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Kernel Trick

However, there are still cases where the data can not be separated by a linear function as shown in figure 3.3. In this case the data is mapped into another higher dimensional feature space (Kernel trick) so that it becomes linear separable again. The function that maps the data is called kernel. For evaluation in chapter 6 the following four kernels from OpenCV [15] and sklearn [68] were used:

- Linear kernel: $k(x, y) = x * y$
- Polynomial kernel: $k(x, y) = (\gamma \langle x, y \rangle + coef_0)^{\text{degree}}$
- Radial basis function kernel: $k(x, y) = \exp(-\gamma ||x - y||^2)$
- Additive χ^2 kernel: $k(x, y) = -\sum \left[(x - y)^2 / (x + y) \right]$

Multiclass SVMs

In case of a multiclass SVM problem the most popular approach is to use several different binary class SVMs instead of only one multiclass SVM [36, 27, 80]. There are

two common solutions for the multiclass problem: One-VS-Rest and One-VS-One.
An One-VS-Rest multiclassifier with n classes consists of n distinct SVMs. Each SVM is trained on one of the classes as class A (positive features) and all other samples as class B (negative features). Classification is done in a "winner-takes-it-all" fashion meaning that the SVM with the highest output function assigns the class label.
In the One-VS-One approach $n\frac{n-1}{2}$ SVMs are constructed. Each class is paired in a SVM with each other class. Classification is done in a "majority-vote" fashion so each SVM votes for its class and the class with the most votes is assigned as the class label.

3.2 K-Nearest Neighbors

The K-nearest Neighbors (KNN) algorithm is a lazy classification algorithm which stores the complete training data in the learning phase [44]. In the classification phase it searches for the k points in the training data that are nearest to the point that should be classified. The label that gets the majority of votes from the k closest points (neighbors) is the result of the classification. See figure 3.4 for a three class example and $k = 3$.
The most popular metric to get the distance between the neighbors is the Euclidean distance, however, the accuracy can greatly be improved by using a learned distance metric such as the Mahanalobis distance which tries to maximize the margin between classes [82].

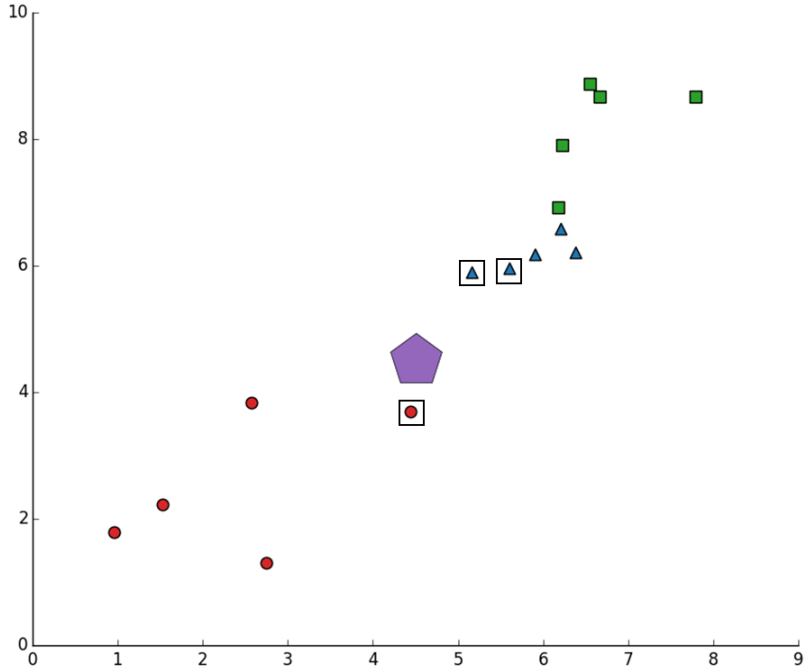


Figure 3.4: Example for a knn classification with $k = 3$. The three nearest elements to the purple pentagon (point that should be classified) are marked with a square. Since two of the three neighbors are triangles the new point should also be a triangle.

3.3 Convolutional Neural Networks

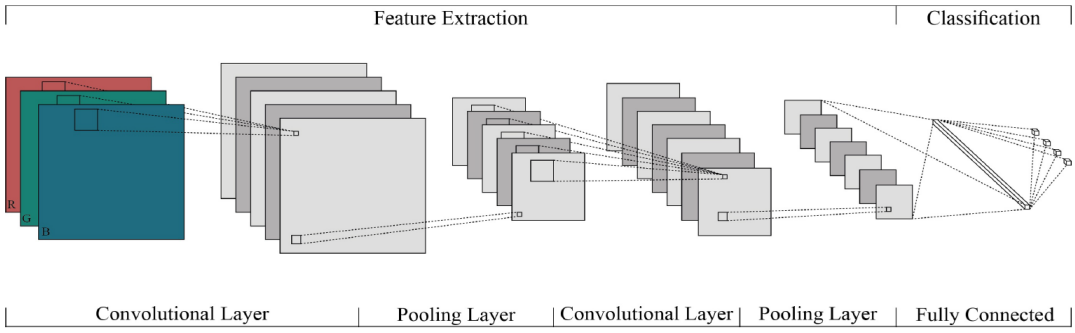


Figure 3.5: Sample architecture of a CNN. Source: Christodoulidis and Anthimopoulos [22]

A Convolutional Neural Network (CNN) is a multi-layer-feed-forward neural network that emulates the processes in the visual cortex by employing feature detection with small convolutional filters. CNNs generally outperform other gradient based learning techniques [55] and are currently state-of-the-art [75] for image recognition.
CNNs usually consist of a succession of trainable convolutional, dense and pooling layers. Normally, the first layers comprise of several convolutional and pooling layers for feature extraction. For the classifications of features extracted in the previous layers, the network has at least one dense layer of neurons before the last layer. The last layer is usually the output layer and for the standard classification task the number of neurons is equal to the number of classes to classify so that the output of a neuron is the probability of a specific class at the same time. Figure 3.5 shows an example of a CNN architecture with two convolutional and pooling layers and a dense, fully connected layer before the output layer.

Convolutional Layers

Convolutional layers contain multiple convolutional filters or kernels. They convolve the image to extract image features. Each filter convolves the output of the previous layer. In case of the first layer the filters operate directly on the input image. In later layers they collect more complex features from previous layers. Convolution happens by calculating the dot-product of the input matrix with the convolutional filter. Filters in

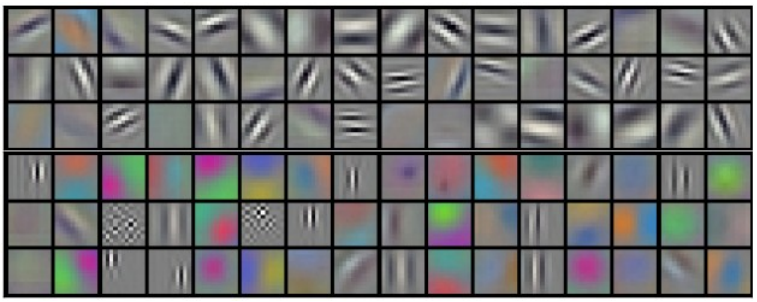


Figure 3.6: 96 convolutional filters of size $3 \times 11 \times 11$ in the first layer. Filters trained on ILSVRC-2010. Source: Krizhevsky et al. [53]

CNNs usually overlap which helps to get a better translation invariance since each pixel is convolved multiple times by the same filter. The main goal of convolution in CNNs is to get distinctive image features. These convolutional filters are not handcrafted like filters in typical feature detectors. They learn the best filters by learning large quantities of data and change over the time and are ultimately able to extract features that are characteristic for the problem. Often, these kernels extract simple features like horizontal, vertical or diagonal edges (figure 3.6). For food, prominent first layer kernels typically include many color kernels that specialize in extracting a range of similar colors [22]. The basic idea behind convolutional filters is that many features like edges or corners appear in multiple locations in images. By using the filters as a sliding window, one filter is able to detect the feature in any position.
Therefore, the main befit of using convolutional filters is the reduction of learnable weights. Filters are very small (7x7 as the maximum size for a current CNN [79]) but are applied on the whole image in a sliding window approach. This means that a single convolutional filter shares weights for the image which significantly improves learning because it reduces the number of parameters to learn [55]. GoogLeNet, a 22-layer CNN, uses RGB input images with a size of 224x224 [79]. A normal Neural Net without convolutions would not be able to handle this input size.

Pooling Layers

Once a feature is detected the exact position becomes less important. Only the spatial relation to other features remains valuable. Knowing the exact position of a feature may even be harmful for generalization because the model becomes less invariant to

position [55]. One way to solve this problem and reduce the number of weights to learn is subsampling. In CNNs this normally occurs in pooling layers. Currently, the most common pooling layer type is max-pooling. This filter basically divides the input into $z \times z$ blocks and extracts the maximum value of each block. A popular max-pooling layer instance is a $z = 2$, stride $s = 2$ layer. This layer downsamples the input by two. Recently, overlapping max-pooling layers have become popular [79]. They do not downsample as much as non-overlapping pooling layers (in fact $z = 3$ and $s = 2$ outputs the same dimensions as the input) but they seem to slightly decrease the error [53].

Overfitting and Dropout

Although CNNs need fewer parameters to train high dimensional inputs than conventional neural nets, they also have the tendency to overfitt like conventional nets do. Overfitting is a common problem in machine learning that occurs if models are very complex. An overfitted model is optimized across the training data and describes or "fits" this data so well that it can not explain unseen data [29]. Figure 6.6a shows an overifitting neural network. After epoch 80 the error on the training set continues to decrease but the error on the validation set stops to decrease which means that although the net gets better at predicting the training data it gets worse for predicting unseen data.
Dropout is a novel way for neural nets to reduce the problem of overfitting by randomly dropping out nodes from the model. This forces the net to generalize better [77].

Learning Rate and Momentum

Neural networks try to find the global minimum of their cost function by adjusting the network weights. If there is only one minimum this is fairly easy as the network only has to adjust the weights towards the downward gradient. Real world scenarios, however, are much more complex and include many local minima as well.
Learning rate and momentum are measurements of how weights are adjusted in back-propagation in search of the global minima. The learning rate denotes the magnitude of the weight change. That means that networks with a high learning rate advance faster along the gradient. Momentum is a term that multiplies a fraction of the previous weight update to the weight adjustment. By applying momentum, the steps the network takes on the gradient towards the minimum get bigger with each iteration. This helps

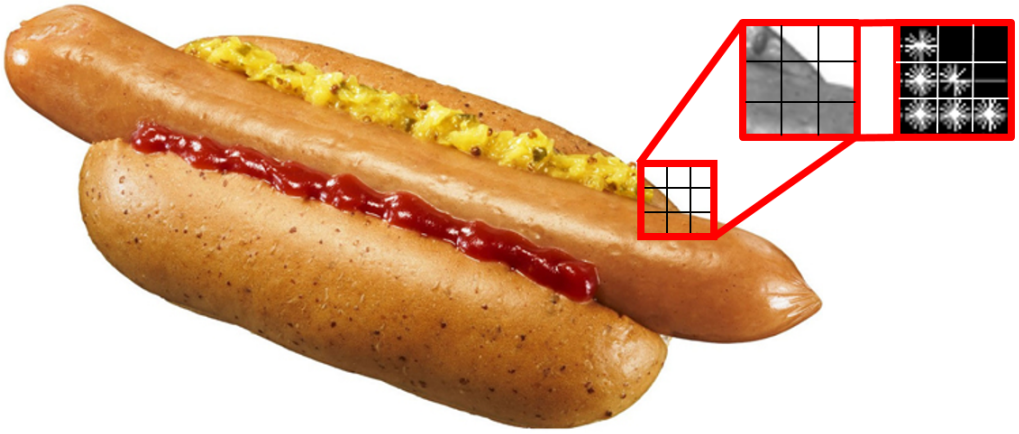


Figure 3.7: Visualization of HOG descriptor.

to increase the speed of learning and may also prevent the network from getting stuck on local minima and saddle points because of the "momentum" the network simply "steps over" these points.

3.4 Feature detectors and descriptors

A common approach in image classification is the use of feature detectors and feature descriptors. A feature detector detects interest points or keypoints in images. Ideally those keypoints are invariant to image transformations like rotation, scale or illumination changes so that the points can be found even if the image is rotated or taken from another perspective. In most cases, keypoints are corners, as corners can be localized quite easily. Knowing that a corner exists, however, does not help much with recognition. An algorithm also needs to describe the characteristics of an interest point so that it is possible to compare different interest points. The part of describing a keypoint is done by keypoint descriptors.

3.4.1 Histograms of Oriented Gradients (HOG)

Histogram of Oriented Gradients (HOG) is a popular image descriptor for human detection, developed by Dalal and Triggs in 2005 [24]. The algorithm makes use of small contrast changes by describing the distribution of local gradients. This approach

is believed to be based on biological processes of neurons in the primary visual cortex [58].
The first step is to divide the image into small blocks with a size of 16x16 pixels and a 50% overlap for better results. Each block is then subdivided by four 8x8 pixel cells. For each of these cells, gradient intensities are computed by applying the 1-D centered point derivative convolution mask horizontally and vertically:

$$D_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad D_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{3.9}$$

The magnitude of the gradients is

$$m = \sqrt{(I * D_x)^2 + (I * D_y)^2} \tag{3.10}$$

where I is the image cell and the gradient orientation is given as

$$\theta = \arctan \left[\frac{(I * D_y)}{(I * D_x)} \right]. \tag{3.11}$$

To achieve a better illumination invariance against shadowing, it is useful to contrast-normalize each cell.
To get a feature vector for each cell, a 9-bin histogram of the gradient orientations (0°-180°) is calculated. The gradient orientations θ are then scaled by the corresponding magnitude m. Figure 3.7 shows a visualization of gradient histograms. The gradient histogram in the center row on the right, is a single horizontal line meaning that the gradients in this cell are all horizontal which is logical because the original image in this cell is a horizontal edge.
To get the feature vector for the whole image all histograms are concatenated to form a large gradient histogram.
HOG is not rotation invariant which does not matter for human detection as humans tend to always have the same upright orientation. However, for food, rotation invariance is quite important.

3.4.2 Local Binary Pattern (LBP)

Local Binary Patterns (LBP) is a simple texture descriptor which was originally proposed by Ojala et al. in 1994 [65]. For the calculation of the LBP descriptor the 8 neighbors

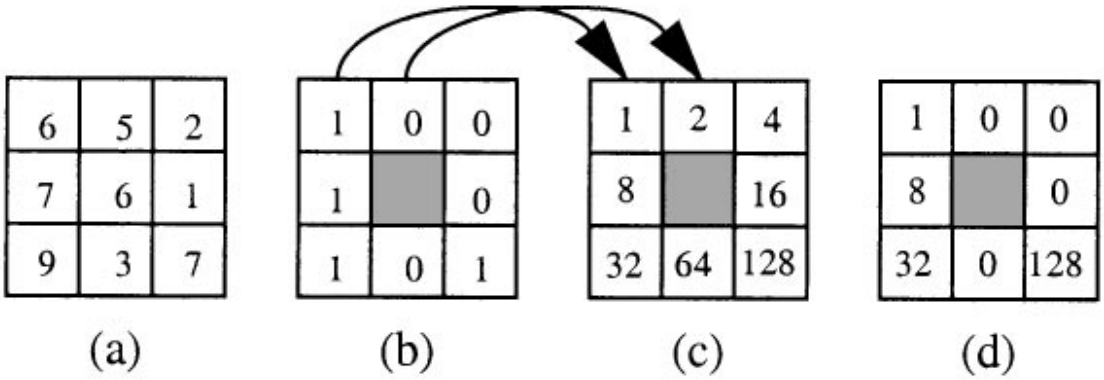


Figure 3.8: Example for a LBP coding of the center pixel with value 6 (a). The neighboring pixels are thresholded against the center value (b) and multiplied by values in (c) with the result in (d). Source: Ojala et al. [67]

in a 3x3 block for each pixel are taken into account (fig 3.8a). The neighbors of the center pixel are then thresholded by comparing the value (grayscale intensity) of the pixel in the center to the intensities of the neighboring pixels. Pixels with a higher or equal value are getting labeled as "1" and "0" otherwise (fig 3.8b). Each position in the 3x3 block is then assigned a weight of 2^n (fig 3.8c) where n is ascending from left-to-right, top-to-bottom. In recent publications these weights are labeled clockwise ascending. The binary values are than multiplied by the corresponding weights (fig 3.8d). The values are then summed up and assigned as the new value for this "texture unit". The values can then be inserted into a histogram to form a 256-dim feature vector since there are 256 possible LBP values ($2^8 = 256$). Local binary patterns are invariant to grayscale changes because LBP describes spatial texture structure but not contrast intensities [67].
In the following years LBP has been adapted and extended with additional contrast information [67] or different neighborhood sizes to achieve rotation invariance with uniform LBPs [66].

3.4.3 Scale invariant feature transform (SIFT)

Scale-invariant Feature Transform (SIFT) is a very popular local image feature detector, descriptor and matcher by Lowe from 1999 [60]. It is invariant to uniform image scaling,

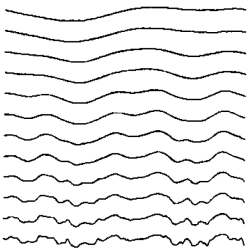


Figure 3.9: Sequence of Gaussian smoothings of a 1D graph with σ increasing from bottom to top. Source Witkins [84]

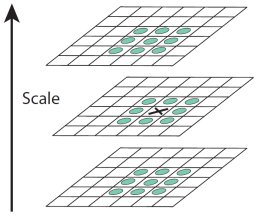


Figure 3.10: Interest points are detected by searching for minima and maxima. The value of the center pixel (marked with X) is compared to its 26 neighbors (marked with circles) on the current and adjacent scales. Source: Lowe [58]

translation and rotation and partially invariant to affine distortions, 3D viewpoint and illumination changes. It is also invariant to noise, partial occlusion and clutter [60].

Keypoint detection

The basic idea of SIFT's keypoint extraction follows Witkins scale-space filtering [84]. Witkins found out that by applying Gaussian smoothings at different scales of σ on a 1D-graph he could find robust edges depending on the scale of σ (figure 3.10a). Since this process can also be applied to 2D images scale-space filtering is a good way to find scale invariant interest points (also called Laplacian of Gaussian (LoG)). SIFT, however, uses a much more computation efficient approximation of LoG called Difference of Gaussians (DoG) [58]. Difference of Gaussians (DoG) can be computed by subtracting two adjacent scales from each other. Figure 3.11 shows the process of creating DoGs at different scales. SIFT samples 3 scales per octave. After each octave the image size is halved and DoG is applied again on the smaller image (figure 3.11).
To check if a pixel qualifies as an interest point at a certain scale, SIFT compares the values of the 3x3 neighboring pixels. If the pixel has the maximum or minimum value SIFT then compares the neighbors on the scales above and below. If the pixel has indeed the lowest or highest value across all 27 points it is taken as a potential interest point (figure 3.10). The cost of this operation is reasonably low because most of the

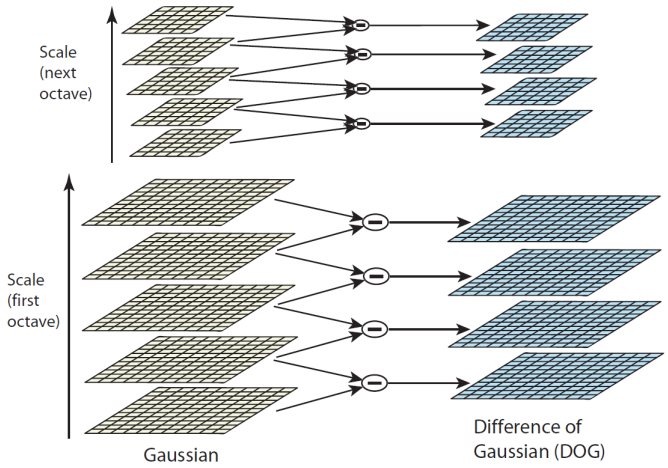


Figure 3.11: For each octaves the image is blurred with Gaussian smoothings at rising scales. A DoG is formed by subtracting two adjacent scales. After each octave the image size is halved and the process repeated. Source: Lowe [58]

pixels will be eliminated after the first few comparisons [58]. To filter weak interest points, SIFT tries to eliminate low contrast points and points on edges since edges are not very robust features. After the elimination of weak points, SIFT computes a 32-bin gradient orientation histogram in the interest point neighborhood. Orientations are weighted by the magnitude of the gradient. Each bin represents 10°of the 360 possible orientations. The bin with the highest peak is then selected as the dominant orientation for this keypoint. Bins with at least 80% of the highest bin value are selected to form an additional new keypoint.

Keypoint Description

The SIFT keypoint descriptor relies on gradient orientation histograms. They are more stable features than just raw intensity values and less sensitive to 3D rotation. To calculate a descriptor for a keypoint, a neighborhood region of 16x16 pixels is selected oriented along the dominant orientation of the keypoint. The 16x16 region is then divided into 4x4 blocks and for each of these blocks a 8-bin gradient orientation histogram is calculated (similar to the keypoint extraction). This results in a 128 dimensional feature vector for each keypoint (16 blocks $*$ 8 bins per histogram $=$ 128)

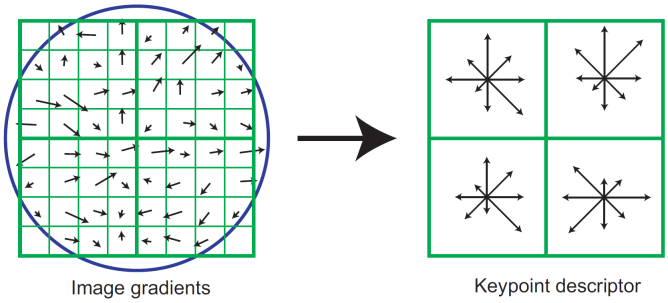


Figure 3.12: Creation of the SIFT descriptor. Around the keypoint a 16x16 region (this image is only half the size) is taken and gradient orientations are calculated. The region forms 16 4x4 blocks with 8-bin gradient histograms. Source Lowe [58]

[58]. Figure 3.12 shows the process of the descriptor creation.

3.4.4 Speeded up Robust Features (SURF)

Speeded up Robust Features (SURF) is a feature detector and descriptor which was proposed in 2006 [4] and later revised in 2008 by Bay et al. [5]. It was developed with the goal to provide a faster alternative for the popular SIFT algorithm without trade offs in recognition performance.

Keypoint detection

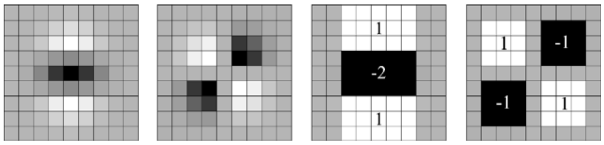


Figure 3.13: Left half: discretized and cropped Gaussian second order partial derivates in y-direction and xy-direction. Right half: corresponding box filter approximations. Gray ares are equal to zero. Source Bay et al. [5]

Like SIFT, SURF also uses scale spaces to detect interest points. SURF, however, relies on an approximation of the Hessian matrix

The first step is to compute the integral image. In integral images, pixel values are the sum of all pixels within a square between the pixel and the origin. These representations are efficient to compute and allow to find the sum of pixel values for any sized box within the image with only four arithmetic operations. Throughout the algorithm this benefit is leveraged to speed up detection and description.
Bay et al. approximate the Hessian matrix using box filters (in the right of figure 3.13). By using integral images in combination with box filters the process of scale space creation can be sped up significantly. The benefit of box filters contrary to Gaussians which have to be applied iteratively is, that box filters can be applied directly on the original image which makes parallelization possible. In addition, SURF does not downscale after each octave. It scales the image up which has the advantage of not causing aliasing. To localize interest points across scales, SURF applies non-maximal suppression on a 3x3x3 neighborhood.

Keypoint Description

To achieve rotation invariance, SURF computes a dominant orientation for each interest point from a circular $6s$ neighborhood by calculating Haar-wavelet responses in x (horizontal wavelet response) and y (vertical wavelet response) directions with s being the scale of the interest point. Again, the combination of integral images and wavelets helps to calculate Haar-wavelets efficiently. The dominant orientation is calculated by the use of a sliding rotating window with an angle of $\frac{\pi}{3}$. The wavelet responses are then represented as vectors weighted with a Gaussian of $\sigma = 2.5s$. Horizontal and vertical vectors are added and the highest sum is the dominant interest point orientation.
To describe a keypoint, SURF forms a square neighborhood region oriented along the dominant orientation with a size of $20s$ (so keypoints at higher scales include a bigger neighborhood). The region is then divided into 4x4 smaller sub-regions. In each of these sub-regions SURF computes features at 5x5 evenly spaced sample points. Contrary to SIFT, SURF does not utilize orientation histograms but again Haar-wavelet responses in horizontal d_x and vertical d_y direction relative to the dominant keypoint orientation. d_x and d_y are then summed up over each subregion. Those two and the sum of absolute response values $|d_x|$ and $|d_y|$ form the four dimensional feature vector for each subregion. Therefore, each keypoint has a 64 dimensional feature vector taking into account that there are 16 subregions each with a four dimensional vector. To achieve additional invariance to contrast and illumination changes the feature vector is turned into a unit vector.

3.4.5 Oriented FAST and Rotated BRIEF (ORB)

Oriented FAST and Rotated BRIEF (ORB) is a combination of the corner detection algorithm Features from Accelerated Segment Test (FAST) [73] and the binary descriptor Binary Robust Independent Elementary Features (BRIEF) [17]. ORB was introduced by Rublee and Bradski in 2011 [74] as a replacement for SIFT for low-power devices and real time performance.

FAST Keypoint Detection

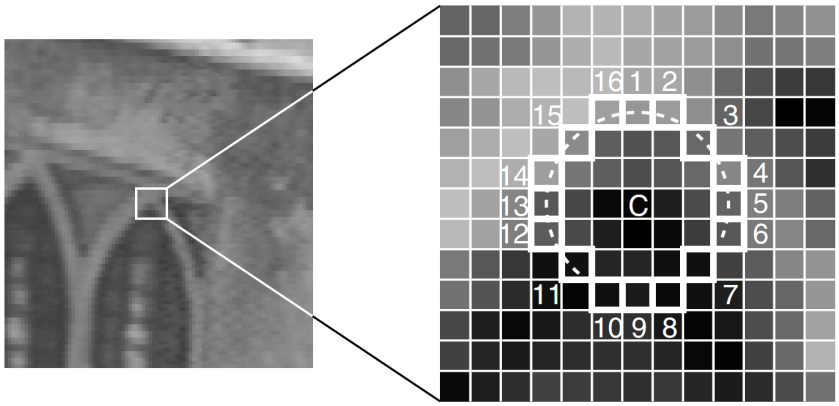


Figure 3.14: FAST corner detection. Interest points are selected if at least 12 adjoining pixel values that are either above or below the center pixels intensity (dashed line). Source: Rosten and Drummond [73].

FAST is a corner detection algorithm by Rosten and Drummond from 2006 that does not rely on Gaussians or scale spaces. FAST compares 16 pixel intensities arranged on a circle with radius 9 around a possible corner point in the center. It is an interest point if it has at least 12 adjoining pixel values that are either above or below the center pixels intensity (figure 3.14). FAST, however, is prone to have many responses along edges which are weaker features so BRIEF tries to filter these edge responses by using a Harris corner measure. To achieve scale invariance BRIEF applies FAST on different image pyramid scale spaces. To also make the keypoints rotation invariant, BRIEF calculates the intensity centroid which assumes that a corner's maximum intesity is offset from the actual center of the corner. This makes it possible to calculate the orientation by creating the vector from the intensity centroid to the actual corner point.

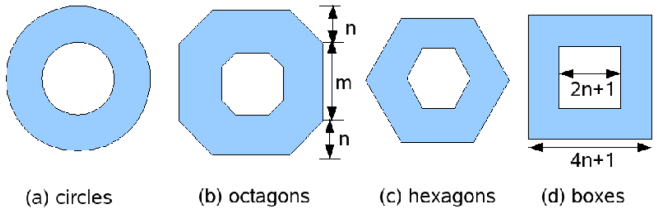


Figure 3.15: Center surround filters. From left to right: decreasing accuracy and computational cost. Source Agrawal et al. [2]

BRIEF Keypoint Description

BRIEF is a binary descriptor from 2010 by Calonder et al. [17]. Unlike SIFT or SURF, a binary descriptor creates the descriptor values by comparing intensity values of pixels in the neighborhood of the keypoint. If the intensity of pixel a is lower than the intensity of pixel b than the test yields "1" and "0" otherwise. To create the feature vector BRIEF performs 256 binary tests and concatenates the results. The pixels that are compared are chosen by a Gaussian distribution around the center of the keypoint. Prior to these tests the image is smoothed. Additional in-plane rotation invariance is achieved by the proposed "steered" BRIEF. Steered BRIEF rotates the matrix of binary tests by a rotation matrix along the keypoint orientation. A lookup table with precomputed ORB features is computed with a 12°interval. However, due to the very rotation by steered BRIEF, the binary tests loose most of their variance so the expressiveness of the descriptor suffers. To prevent this, Rublee and Bradski proposed an algorithm to "learn" the optimal location of the pixel-pairs that are compared for the binary tests.

3.4.6 Center Surround Extremas (CenSurE)

The Center Surround Extrema (CenSurE) feature detector was proposed by Agrawal et al. in 2008 as a framework of several bi-level filters for interest point detection [2]. CenSurE is scale and rotation invariant and similar to the way SIFT and SURF perform interest point detection. Contrary to the aforementioned keypoint detectors, CenSurE does not up- or downscale the image in scale space and therefore does not inherit the disadvantages of up- and downscaling like inaccurate keypoint locations.
To detect corners CenSurE employs approximations of LoG with center-surround discrete bi-level filters. The closest approximation of LoG is a circular bi-level filter as

shown in figure 3.15a. Bi-level means that the filter only contains two values namely 1 or -1. The circular filter is rotation invariant due to its symmetry but it is also expensive to compute. Therefore, CenSurE uses octagons and boxes as approximations of the circular filter. Like SURF, CenSurE also leverages integral images for box filters and applies them on five scales. Since a major shortcoming of box filters is the lack of rotation invariance Agrawal et al. also proposed an octagon shaped filter which resembles the circle filter much more closely. Integral images, however, do not work with diagonal edges and thus do not work with octagons. To make computation still possible Agrawal et al. introduced "slanted" integral images which are able to construct any trapezoidal area in the same amount of time as rectangular integral images.
To detect interest points, CenSurE performs non-maximal suppression over scale space across a 3x3x3 neighborhood. Weak interest points are filtered by threshold and edges are filtered the Harris corner measure.

3.5 Bag of Words

Bag of Words (BoW), bag of features, bag of visual words or bag of keypoints is a simple, yet powerful image feature quantization method. The original idea of a BoW approach for images came from texture classification [56]. Leung and Malik found out that by quantification of small texture patches (textons) in histograms they were able to accurately classify different textures. In 2004 Csurka et al. refined this approach and applied it to general image classification [23].
The idea behind a BoW is very simple: During learning the BoW clusters image descriptors using k-means clustering. The clustered representation of the image descriptors is called vocabulary. The vocabulary size is k, so using $k = 1000$ for the k-means clustering results in a vocabulary of 1000 image features. Choosing k is a trade-off between speed and accuracy and has to be determined during experiments. In an ideal world, a BoW clusters image features together that are unique and expressive so that we would have a vocabulary of bread, salad, meat and noodle features.
During classification the BoW gets a set of image descriptors and matches these features with the previously learned vocabulary. The output of this process is a histogram of vocabulary occurrences. Referring to the ideal world example, the BoW would get a set of features including many that look like bread, some that look like meat and a few that are similar to the salad texton. The output would be a histogram with a large value for the bread bin and some smaller values for the salad and meat bins. Giving this

occurrence vector to a trained SVM would reveal that the image might be a Hamburger.

3.6 Methodology

3.6.1 Performance Measurements

Precession - Recall

The most used measure for the precision of food classifiers is the average accuracy which is calculated by dividing the number of correct matches and the total number of samples. Accuracy, however, gives no information about the underlying conditions. It is a measure of overall performance. To have a higher chance of suggesting the correct items, future systems may present a list of options that the user can chose from. Intuitively, the accuracy is much higher if a classifier can present a list of items with high confidences instead of only one item because the problem is much easier. Accuracy, however, does not measures how easy a problem is. If a classifier were able to suggest all classes as options the accuracy would always be 100% although the results are not useful at all.
The combination of precision and recall objectively measures the actual relevance and performance of a classifier for a class of images because it includes the amount of considered items and the correct predictions. In this case the amount of considered items changes based on how many items the classifier can suggest. Precision and recall is defined as:

$$Precision = \frac{T_p}{T_p + F_p} \quad Recall = \frac{T_p}{T_p + F_n}. \tag{3.12}$$

- True positives T_P is the number of correctly classified images of a class.
- False positives F_P are all images that the classifier predicted to be positive but are in reality negative. (Type I Error)
- False negatives F_N are all images that are positive (belong to the class) but are labeled as negative (do not belong to class) (Type II Error)

A high recall means that many images were matched correctly and a high precision denotes a low number of incorrectly classified images. The bigger the area under the Precision-Recall curve the better the classifier.

Null Error Rate

The null error rate is a baseline for any classification task that calculates the accuracy if a classifier would just predict the class with the most images.

Confusion Matrix

Confusion matrices are one of the most important metrics to understand why a classifier struggles with certain classes while getting a high precision with others. As the name suggests, a confusion matrix tells if the classifier "confuses" two classes.
A confusion matrix for n classes is always a $n \times n$ matrix where columns represent the actual images classes and rows represent the predicted image classes so if the diagonal of the matrix has high values this means that the classifier makes correct predictions.

Categorical Cross-Entropy

The categorical cross-entropy L_i is an error function that is used for the training of neural networks in classification tasks as the objective function. It is more versatile than the accuracy or the Mean Squared Error (MSE) because it takes the deviations of the predicted label $p_{i,j}$ and the actual label $t_{i,j}$ into account and weights the "closeness" of the prediction with the logarithm. For classification, cross entropy is more useful than MSE because MSE gives too much emphasis on incorrect predictions. The categorical cross entropy function is defined as:

$$L_i = -\sum_j t_{i,j} \log(p_{i,j}) \tag{3.13}$$

The loss values that are used for the discussion of results for neural networks are the average values of the categorical cross-entropy (Average Cross-Entropy Error (ACE)).

3.6.2 Cross Validation

Cross validation is one of the most essential techniques to evaluate real-world classification performance. Classifiers like SVMs or neural networks are always better on data they have already seen. This is called overfitting (see section 3.3). By training and testing on the same data the classification performance would be much better than the actual real world performance. To test if a classifier can actually work with samples it has not seen cross validation divides the dataset into different partitions.

For most tasks it is sufficient to divide the dataset into a training and a test set. The data in the training set is used to train the classifier and the test data is used to evaluate it with data is has not seen before.

k-fold Cross Validation

To make the classification evaluation even more robust, k-fold cross validation is used. By applying k-fold cross validation the dataset is randomly partitioned into k different parts. $k-2$ parts are used for training and two parts are used for the evaluation. This process is repeated k-times and after each iteration the parts are exchanged so that at the end, each sample was used for training and for validation. Calculating the mean of the k evaluations gives a much more robust measurement because the evaluation does not depend on the difficulty of the test partitions.

4 Experimental Setup

The following chapter describes the experimental setup for the discussion of results in chapter 6. All classifiers were tested with cross validation using a train - test split ratio of 80% - 20%. 5-fold cross validation was only used for the evaluation of the histogram classifier but not for the experiments with the other classifiers since the computational cost for 5-fold cross validation is very high.

4.1 Hardware

The training and the evaluation of the classifiers was done on 4 different servers all running Ubuntu. Two of the servers (Schlichter2 and Schlichter4) belong to the faculty of applied informatics and the other two are cloud instances. One is an azure virtual compute instance with 8 Central Processing Unit (CPU) cores and 28 Giga Bytes (GB) of Random Access Memory (RAM) and the other is an Amazon EC2 g2.2xlarge Graphics Processing Unit (GPU) instance with a Intel Xeon E5-2670 processor, 15 GB of RAM and a NVIDIA Grid K520 GPU. See table 4.1 for more details.

Table 4.1: Used hardware for model training and evaluation.

	OS	**CPU**	**RAM**	**GPU**
Schlichter 2	Ubuntu 12.04	Intel Core i7-3930K @ 3.20GHz	63 GB	NVIDIA Titan X
Schlichter 4	Ubuntu 14.04	Intel Xeon E5-2620 @ 2.00GHz	28 GB	-
Azure	Ubuntu 15.10	Intel Xeon E5-2673 v3 @ 2.40GHz	28 GB	-
Amazon AWS	Ubuntu 14.04	Intel Xeon E5-2670	15 GB	NVIDIA K520

4.2 Datasets

The classifiers are primarily evaluated on two datasets. For the parameter optimization a smaller 8-class subset of the UEC Food-100 dataset is used because the computational cost of training on huge datasets is very high. The 8 classes from the original dataset were selected randomly. The dataset consists of the classes

1. Beef
2. Croissant
3. Eels
4. Egg hotchpotch
5. Hamburger
6. Pizza
7. Sashimi
8. Sauteed burdock

The sampled dataset is very imbalanced with a mean of 134.5 images per class. The class with the lowest number of images contains 108 images and the class with the highest amount of images has more than twice as much images (233 images).
Each feature classifier is then also tested on the full 50-data dataset (50 classes). The color histogram is also tested on the complete UEC Food-100 dataset. The other classifiers are not evaluated on this dataset because evaluating the histogram classifier with a 5-fold cross validation took about 42 hours on the azure cloud virtual machine. The neural networks are not tested on 50-data or Food-100. The number of images per class of 50-data and Food-100 are not sufficient enough for convolutional neural network training. 50-data only contains 5000 images and Food-100 includes 14,357 images. As a compromise between training time and images per class a special dataset is used. To increase the number of images to 2000 images per class, the intersect between ETHZ Food-101, UPMC Food-101, 50-data and Food-100 is formed and the classes are combined. The false positive rate for the UPMC Food-101 dataset is very high so most of the false positive images were manually removed. This dataset consists of 6 classes with an average of 2004 images per class. A smaller net is also evaluated on a smaller

subsampled dataset of the 50-data dataset. This subsampled set contains the classes "Arepas", "Braised pork", "bread", "buns", "chaisu", "chicken rice" and "chocolate". Table 4.2 accumulates statistics about the used datasets.

Name	Classes	Images	Null Score	Mean # Images	Standard Deviation (std) #
50data	50	4967	0.0201	99	2
Food-100	100	14,357	0.0507	144	87
Food-100 8-class	8	1076	0.2165	135	41
Big Intersect	6	12,029	0.1893	2005	172
50data 8-class	8	800	0.125	100	0

Table 4.2: Dataset details used for the results in chapter 6.

4.3 Data Preprocessing

For all algorithms every image is loaded by OpenCV and resized so that each image has exactly the same image area while keeping the aspect ratio locked. This is extremely important because otherwise the quality and quantity of features that can be extracted from the images varies based on the image size. This variation greatly affects the accuracy.
In addition, neural nets require a fixed aspect ratio of the input over all images. The aspect ratio 1×1 was chosen because most images are already very close to that ratio and it is generally a common practice to have this kind of input ratio for neural net inputs. If images have a different aspect ratio, they are cropped around the center. If the aspect ratio is smaller than 0.3 or bigger than 3 the image is skipped because cropping would omit too much of the image. The images are not simply resized since the resulting distortion might decrease accuracy.
For most of the algorithms the image is loaded in gray scale. Exceptions are color histograms and neural nets that take color images as input. Color correction, denoising, or contrast adjustments are not performed because while testing they could not improve accuracy and often led to poorer overall results.
Again, neural networks get the image data a little bit differently. The used framework for neural nets requires the data to be scaled to a $[0, 1]$ interval, 32-bit floats and be reshaped to [number of channels $\times$ width $\times$ height]. All other models get data scaled as

normal 8-bit $[0, 255]$ pixel values and shaped as [width $\times$ height $\times$ number of channels].

Data Augmentation

(a) Original image (b) Increased brightness (c) Random cropping (d) 20°rotation (e) rotation and horizontal flip

Figure 4.1: Examples for label preserving data augmentation.

To reduce overfitting of neural networks data augmentation is performed. Data augmentation is a very common method to artificially increase the size of the dataset by performing label-preserving image altering methods [26, 53, 22].
In this case there are nine operations that can be applied to an image. The maximum number of successive data augmentations on a single image is 4, although, the probability for such operations is linearly decreasing with a probability of 60% for the first and 15% for the fourth augmentation.
The first data augmentation category consists of image brightness (fig. 4.1b) and saturation changes in the Hue, Saturation, Value (HSV) color space. Brightness and saturation are scaled by an uniform random number in the interval $[0.25, 1.75]$. Dosovitskiy et al. proposed to raise these values by a power of up to 4 [26]. However, changing saturation or brightness by 4 increases these values far to much.
The second category contains horizontal- and vertical flipping (fig. 4.1e) and $[90°, 180°, 270°]$ rotations.
The last category requires that the images are loaded slightly bigger than the input size for the neural net requires. On the larger images random cropping can be performed (fig. 4.1c). Larger image patches also allow "free-rotation" in the interval of $[-30°, +30°]$ (fig. 4.1d). This is done by cropping the image after the rotation so that the area that is cropped away is minimal.

4.4 Algorithms Variations

4.4.1 SVMs

SVMs are binary classifiers so it's not possible to train all n food classes with m_i images in the i-th class directly on them. To solve this problem n One-VS-Rest SVM are trained (see section 3.1). Since there are $n-1$ times more negative samples than positive samples, the negative samples get undersampled so that there is a balance between positive and negative samples. In this case undersampling is done by selecting only $\frac{\bar{m}}{n-1}$ images per class. In contrast to random undersampling, this assures an even distribution of all negative classes in the negative sample space.
Normally, the data for SVMs is scale normalized before the training or prediction but for the used SVM implementation scaling did not affect accuracy.

4.4.2 Color Histograms

This classification method relies on histograms of pixel color intensities. The classifier computes 3 histograms of an image patch. One for each channel in the HSV color space though the color space does not really impact accuracy. RGB channels produce the same accuracy rate.
Histograms are global features which means that the pixel intensity distribution applies to the whole image and any spatial information of the features is lost. To provide some basic spatial information and to increase the size of the feature vector without sacrificing robustness to color changes the image is segmented into multiple image patches. Since there is no reliable component segmentation yet, the image is divided into n rectangular blocks (instead of component segmentation). For each image patch 3 histograms are computed. By concatenating each histogram a feature vector is created. The classification is then done by a SVM.

4.4.3 Feature Detectors

The original purpose of feature detectors and descriptors like SIFT (section 3.4.3) or SURF (section 3.4.4) was to provide reliable matching between different views of the same scene [58]. Food classification is different because the task is not to recognize the same object from different angles but to classify similar looking objects into different classes. Therefore, only the keypoint detection and description parts are used and not the additional keypoint matching functionality SIFT and SURF provide.

The recognition task using SIFT-like classifiers comprises of four steps:

1. Keypoint detection
2. Keypoint description
3. Feature quantization using a BoW
4. Classification using One-VS-Rest SVMs

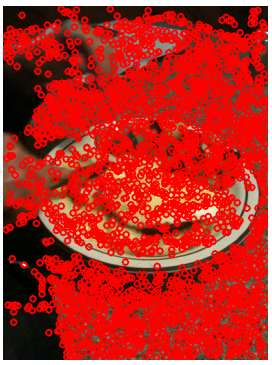
Keypoint Filtering

In the keypoint detection stage, detectors often select false keypoints (figure 4.2a) because they do not differentiate between relevant points of the food object and edges in the background. Another problem is that the number of extracted keypoints changes, based on the number of image features. A low contrast image has less keypoints than a high contrast image with many edges which leads to problems in the classification stage. So images have either too many or too few keypoints. To solve these problems an additional keypoint selection algorithm is applied after the keypoint detection. It leverages the fact that people who take photos of their food almost always center the food objects in the middle. This makes it possible to weight the keypoint response values r_i with the distance to the center d_i so that a keypoint score can be calculated for all n keypoints. For this score the distance of all keypoints is scaled to the interval $[0, 1]$ with "0" being the closest to the center and "1" being the farthest away. The same is done for the keypoint responses ("0" is a weak keypoint, "1" is a robust keypoint). The score S_i is then given as

$$S_i = 1 - (d_i * (1 - r_i)) \quad \forall i, i = 1 \dots n. \tag{4.1}$$

A high score $S_i \in [0.9, 1]$ means that a keypoint is probably valuable because it is close to the center and therefore describes the object and not the background and it also has a strong response value which means that the point should be robust. The list of keypoints is then sorted by the descending score value. Since the number of extracted keypoints should be the same for all images only the best m keypoints are chosen. Figure 4.2b shows the result of this process. In this special case the table surface has a high-contrast texture which SIFT recognizes as keypoints. The filtered version does not contain the irrelevant background keypoints.
For many low-contrast images which lack enough distinctive features, SIFT, SURF or FAST provide less than m keypoints. For these cases the algorithm augments existing

keypoints. To do this, points are randomly sampled using a Gaussian distribution around the mean of all sampled keypoint positions. The standard deviation equals 0.5 the standard deviation of the keypoint positions. The result of this imputation is shown in figure 4.2c.



(a) Unfiltered SIFT keypoints

(b) Filtered SIFT keypoints

(c) SIFT keypoints (red) and augmented keypoints (green)

Figure 4.2: SIFT keypoint filtering and imputation with $m = 600$ keypoints.

4.4.4 Neural networks

Learning Rate and Momentum

Learning rate and momentum are important hyperparameters. Especially with the learning rate and momentum, there are problems if they are fixed. By using a low learning rate the network learns very quickly but may not learn a good model if it oversteps the global minima. If the learning rate is two low it takes many epochs to train the network and it might even get stuck on a local minima.
By adjusting the values dynamically most of these problems can be solved. Learning rate and momentum are adjusted in the same way although they have both different starting values. For the fist 50 epochs both values are linearly decreased to $1/5$ of the start value. For the next 50 epochs they retain this value and after the 100th epoch momentum and learning rate are again linearly decreased until they reach $1/500$ of the value after the 100th epoch.

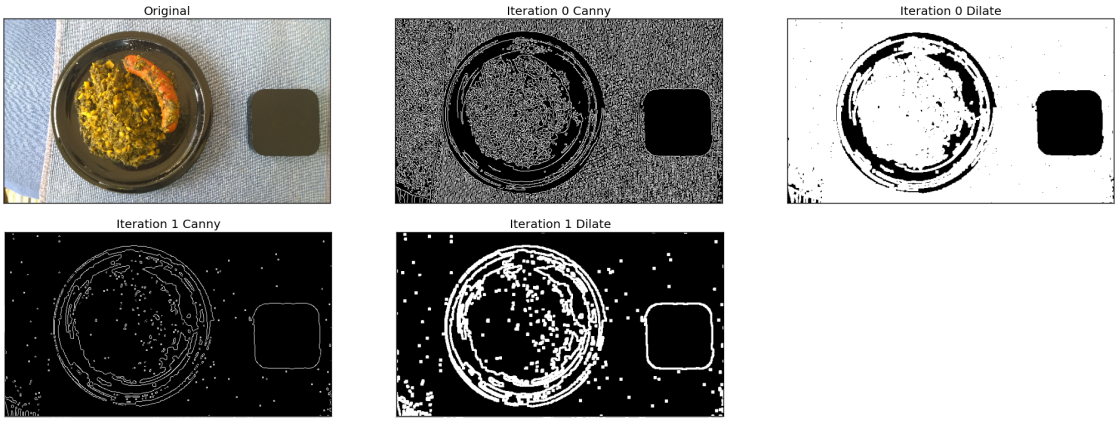


Figure 4.3: Edge detection process on difficult image. From left to right: a: Original image, b: Image after first Canny operation, c: Canny image after dilation, d: Canny applied on dilated image, e: canny again to close gaps in the plate edge.

Early Stopping

Many complex neural networks suffer from overfitting if trained to long. To prevent overfitting on a good network, the training is stopped and restored if the loss on the validation set has not improved for 100 consecutive epochs.

4.4.5 Size Estimation

Portion size estimation using images is extremely difficult. To get accurate measurements of weight an algorithm has to calculate volume and has to correctly classify an image to determine the density of the food. Volume estimation is difficult even for state of the art neural networks [62] so the developed algorithm will instead focus on area estimation of meal plates and meal parts.

Meal Size Estimation

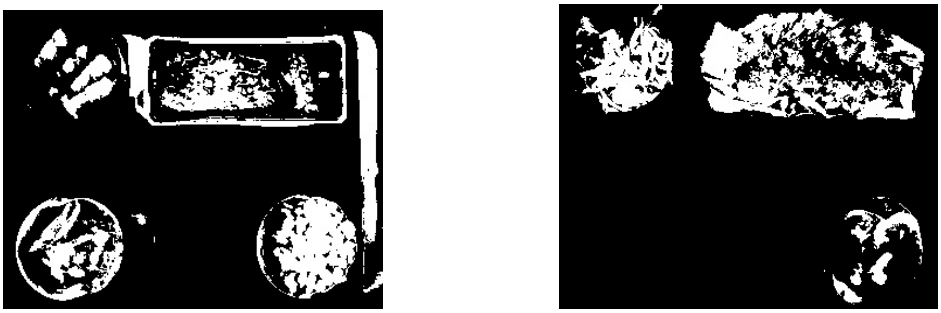
The basic idea behind the algorithm is based on arbitrary markers of known sizes besides the meal plate. When the size of the marker and the meal in the image is known, the real world meal area can be approximated. Therefore, the goal is to find the plate and the marker.

The first step to reach this goal is to find objects. This is done by applying Canny edge detection [18], an extremely popular edge detection algorithm on the image. Sometimes the table surface itself is textured like in figure 4.3a. In this case the output of the edge operation is not very useful since it has to much noise (figure 4.3b). To solve this problem the image is dilated which reduces the noise (figure 4.3c) but also blurs edges so canny and dilation are applied again which leads to a clean edge image. This process can also be applied on images with no table texture without reducing the performance in the following object detection step.
In the next step the algorithm searches for the meal and marker regions by iterating over the image objects. This iteration is done by searching for the largest remaining contour in the image and then deleting this contour. If the edge preprocessing was perfect there are only two contours remaining. The marker and the meal. If the marker was not yet found the current image region is added to the list of potential meal regions. If the marker was found and there are potential meal objects the iteration is finished and the size of the meal is estimated based on the size of the marker in the image. The marker can be anything rectangular although a chessboard pattern is more reliable. The chessboard is recognized by a camera calibration algorithm from OpenCV. In case of custom markers the user has to take a closeup or cropped image of the marker and input the size of the custom marker. By using SIFT keypoints the learned marker is matched against potential image regions and if enough matches are found and the ratio between good matches and bad matches is greater than 0.45 the image region is marked as the marker region and the size is calculated by fitting a bounding rectangle around all SIFT keypoints which is shown in figure 4.4.

Meal Part Size Estimation

For one-component food items this approach works really well but as soon as a plate contains different kinds of food, this approach becomes inaccurate because the size is measured for the whole plate and not for the components.
This is solved by color filtering the image. To get color filters the 10 most common colors in the meal are calculated by a modified median cut algorithm by Bloomberg [10]. The image is then converted into the HSV color space and filtered against each of the most common colors which creates an image mask that is used to cut out the meal regions. Figure 4.5 shows an example of two color masks. Regions under a certain threshold are discarded. The same marker-principle is then applied and the size of the image parts is calculated.

Figure 4.4: Example of custom marker matching. The figure shows the matching of a learned marker (top left) and the matchings between the marker and an image region. Detected keypoints for the bounding rectangle are shown in the bottom left.



(a) 4/4 items detected

(b) 3/3 items detected

Figure 4.5: Color masks after calculation of dominant color palette.

5 Implementation

For the evaluation of various image recognition algorithms on different kinds of food datasets a program called TUM Food Cam (TFC) was implemented. It is able to calculate several performance measurements, load and preprocess image datasets, use these datasets to train image classifiers from different libraries and optimize classifier parameters.
Section 5.1 gives general information about the infrastructure of the program and section 5.2 describes the modules in detail.

5.1 Infrastructure

5.1.1 Architecture

Figure 5.1 describes the architecture of the most important classes. The class diagram does not show all classes, methods and attributes but only the general relationship and interaction between the classes. This was necessary because the scope of the program is quite big and including all data would not have helped to understand the overall view. All classifiers in the "TUM Food Cam" application are controlled by the *ModelController* Singleton. Both the main class *TumFoodCam* as well as the class *ParameterTester* use the controller to create and load classifers. The *ModelController* instantiates exactly one *Classifier* object which can either be a *NeuralNetClassifier* or a *FeatureClassifier*. *Classifier* is an abstaract[1] class that provides access to *ModelSaver*, *Testdata* and the *ModelTester* class.
NeuralNetClassifier contains one *NetModel* which is an abstract wrapper for a concrete neural net architecture like *net_100_nnSimple* and extends the abstract *Model* class. *FeatureClassifier*, the other class that extends *Classifier*, provides methods for the creation and management of SVMs or k-nearest-neighbors. *Histogram* and *Lbp* derive directly from *FeatureClassifier* since they both calculate a global feature vector that can be used

[1] since there are no abstract classes or interfaces in python, an "abstract" class is implemented by raising a *NotImplementedError* in each method body.

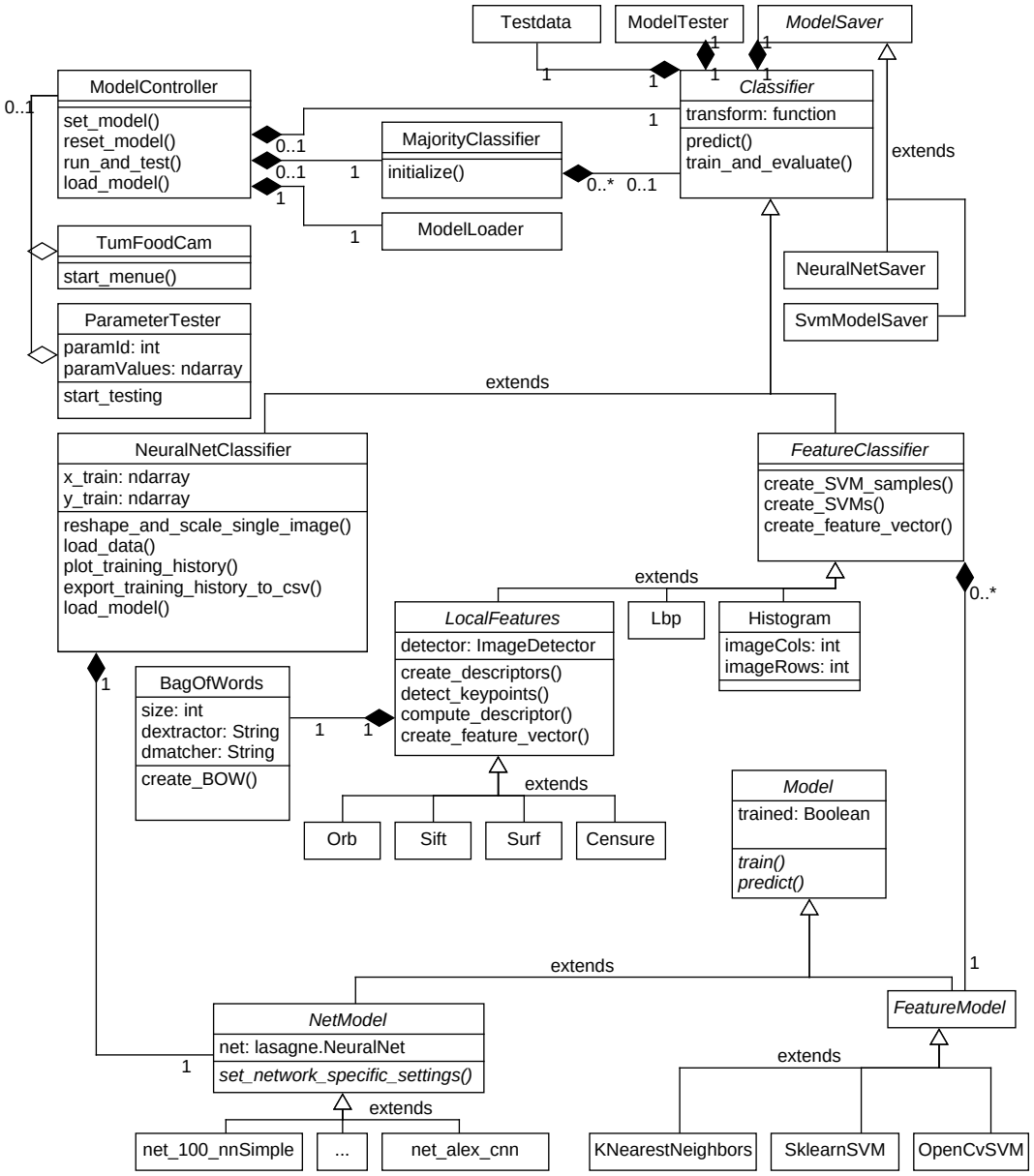


Figure 5.1: Reduced class diagram of TumFoodCam architecture.

directly for SVM training. The other classifiers extend *LocalFeatures* which derives from *FeatureClassifier*. *LocalFeatures* controls the creation of keypoints, descriptors as well as training and use of a BoW. The classification of the images is done by one or multiple instances of *FeatureModel*.
Almost all parameters for methods and general application behavior are globally defined in the *Settings* module. Settings can be loaded and saved by the *SettingsService* (not part of figure 5.1).

5.1.2 Patterns

Strategy Pattern

The application uses several different libraries for classification and feature extraction. To unify the general process of learning, all classifiers are abstracted by the *Classifier* class. It acts as an interface and mainly declares two methods to train and predict images. Subclasses will implement those methods so that the *ModelController* does not need to know the difference of training a neural net or a SVM feature classifier which makes it possible to chose a classifier at runtime and add new classifiers easily. This pattern is called "Strategy pattern".

Template Pattern

The "Template pattern" is almost the same as the strategy pattern but is applied on the *FeatureClassifier* level. *FeatureClassifier* is an abstract class and provides methods for SVM creation. It acts as a template for any feature classifier that relies on SVMs.

Facade Pattern

This pattern is applied at the last layer where the actual API is accessed. There is no common naming convention between different APIs so although `censure.detect(image)` and `sift.detect(image)` might have the same syntax, they work differently because OpenCV returns the keypoints and skimage does not. Classes like *Sift* or *Censure* therefore have to act as a wrapper or facade for the actual API calls and unify them for their super-classes.

5.1.3 Libraries

OpenCV

OpenCV which stands for "Open Source Computer Vision" is a cross platform framework with over 2,500 optimized algorithms [15, 1]. OpenCV is written in C++ and includes a wide variety of computer vision algorithms, image recognition toolkits and a machine learning library and is the de facto standard for computer vision. OpenCV is licensed under BSD which makes it free for academic and commercial use[2]. OpenCV is the most important library in TUM Food Cam (TFC) because it handles most image file Input / Output (IO), transformations and classification.

Scikit-learn

Scikit-learn or sklearn is an open source machine learning library for python [68]. Scikit-learn is used because its LibSVM implementation is much more open and allows more kernels than OpenCV does.

Scikit-image

Scikit-image is a library with many image processing and computer vision algorithms [81]. It provides keypoint detectors and descriptors like HOG or CenSurE that OpenCV does not include.

Lasagne and Nolearn

Lasagne is a library that is used to build and fit neural networks in Theano [25]. Lasagne was selected as the library for all neural networks because it is based on theano, a commonly used numerical computation library which supports Compute Unified Device Architecture (CUDA) for efficient execution on GPUs. Lasagne adds a layer of abstraction to theano which makes it easier to create and train neural networks because it specifically sets the focus on an easy to learn syntax that is compatible to sklearn.
Lasagne is further refined by nolearn which makes the process of neural network architecture creation even more easier to read and provides additional useful methods

[2]SIFT and SURF are patented [59] [32] and should not be used in commercial applications.

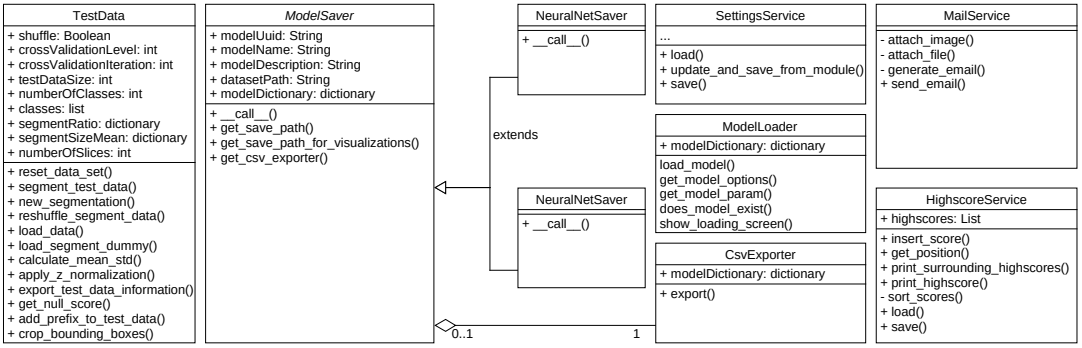


Figure 5.2: Class diagram of data_io package.

for the training and visualization process like layer activations. All neural networks in this application are nolearn-lasagne-theano nets.

5.2 Modules

5.2.1 Input / Output

The *data_io* package contains modules focused on the input and output of data.

Test Data Module

This module manages and loads data from different image datasets. It builds a class directory by iterating over the folders in the dataset root path. Depending on `crossValidationLevel` which is the *k* in *k*-fold cross validation and `segmentRatio` it slices the data into *k* parts. After each cross validation iteration `new_segmentation()` should be called which shifts the segments for a new validation. After the last iteration it returns `False`.
`load_data()` is the method which iterates over the actual images. It can be called with different parameters to restrict certain classes, the number of images per class, or apply different transformation, data preprocessing and size specifications.

ModelIO Service

model_io is a module which contains classes for saving and loading of models as well as a class to export data to Comma Separated Values (csv) files. *NeuralNetSaver* and *SvmModelSaver* both derive from *ModelSaver*. *ModelSaver* manages the save directory of the model by giving each model a unique id which is saved in the `modelDictionary`.

Settings Service

The class *SettingsService* uses the global settings variables in the module *settings* to store and restore the settings.

5.2.2 Neural network modules

Neural networks

There are several modules in the "deep" package which is a sub-package of "classification". *neural_net* includes the *NeuralNetClassifier* which is the central controller class for every neural network. It instantiates a *NetModel*, loads dummy data in case the dataset will be augmented or loaded lazily and provides data logging methods.
Classes that extend *NetModel* are the wrappers for the actual neural net API. They each instantiate a specific net architecture. For better evaluation and training performance these classes are able to make use of the modules *batch_iterators*, *learning_functions* and *training_history*.
The module *batch_iterators* manages the loading of image batches for each iteration. It contains the classes:

- *LazyBatchIterator* which only loads image batches when they are needed. Without it the whole dataset would have to be loaded into memory prior to the training.
- *AugmentingBatchIterator* which augments images. The process is detailed in section 4.3.
- *AugmentingLazyBatchIterator* is a combination of both batch iterators.

learning_functions contains the classes *AdjustVariable* and *EarlyStopping*. Both are explained in section 4.4.4.
training_history provides methods for logging and plotting of the model fitting.

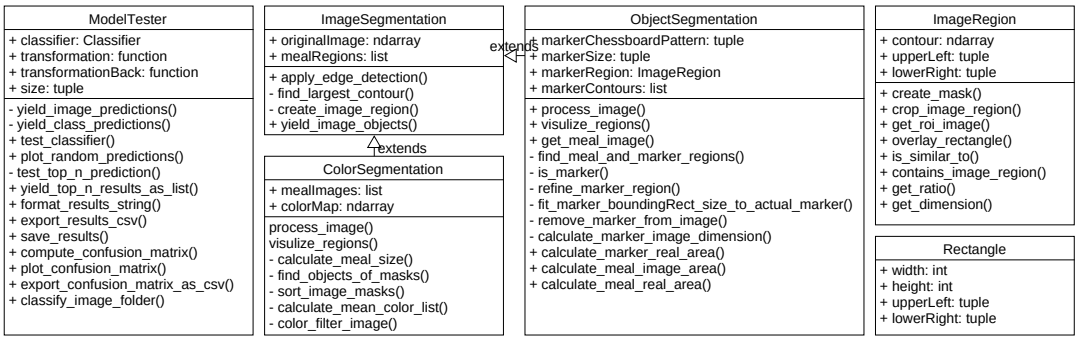


Figure 5.3: Class diagram of ModelTester and classes from the segmentation package.

5.2.3 Testing

Model Tester

The task of *ModelTester* is the evaluation of models. Calling `test_classifier()` on a test data segment will test the classifier by comparing the predicted class to the actual class. For each class as well as for the whole segment the accuracy, recall and precision are calculated and exported. The class does also plot a confusion matrix and a prediction of random images visualization.

Parameter Tester

ParameterTester is part of the *model_controller* module. Given a classifier, and a *Testsuite* with a list of parameters and corresponding parameter values it will test each parameter value on the classifier and plot and export the results.

5.2.4 Image Segmentation

image_segmentation contains 3 image segmentation classes for segmentation, meal area size estimation and meal part size estimation.

- *ImageSegmentation* is the super class for *ObjectSegmentation* and *ColorSegmentation*. It applies canny edge detection and image dilation and provides a method which iterates over found image objects detected by the OpenCV `findContours()` algorithm.

- *ObjectSegmentation* searches for the meal and a marker and approximates, based on the marker, the area of the meal.
- *ColorSegmentation* further analyzes the meal region for different meal parts by applying selective color filtering. For each meal region the size is approximated.

5.2.5 Helper Functions

The *utils* contains over 60 functions used in all program parts from image transformations over console menus and output.

6 Discussion of Results

In this chapter the performance of different food image classification methods is evaluated. In addition the impact of important classifier parameters is compared.

6.1 Hyper Parameter Optimization

The hyper parameter optimization is done using a simplified grid search and random search. An exhaustive grid search tests every parameter value by using a cartesian product with the other parameters. Due to the high dimensionality of classifier parameters and the high computational cost of classifier training, exhaustive grid search is not possible in a feasible amount of time and Bergstra and Bengio even showed that random search can outperform grid search [7]. Therefore, parameters are tested individually in addition to random searches on important parameters. This is an acceptable tradeoff between accuracy and time since the goal is not to get optimal parameters but to get a general direction of good parameters for this problem. The following section will first focus on the impact of common parameters like SVM kernels or image size on the performance and then attend to more classifier specific parameters like the number of image segments for color histogram features.

6.1.1 SVMs and K-nearest Neighbors

For classifiers, the model that they are trained on, is the most important parameter. Color histograms, SIFT, SURF, ORB and LBP were tested on the k-nearest Neighbors model and SVMs with four different kernels:

- Linear kernel (OpenCV)
- Polynomial kernel with degree 2 (OpenCV)
- RBF kernel (OpenCV)
- Additive χ^2 kernel (sklearn)

Model	Color	SIFT	SURF	ORB	LBP
Linear	0.418	0.444	0.407	0.285	0.312
Poly	0.494	0.347	0.339	0.282	<u>0.519</u>
RBF	0.125	0.523	0.455	0.263	0.364
χ^2	<u>0.593</u>	<u>0.648</u>	<u>0.606</u>	<u>0.423</u>	0.391
KNN	0.412	0.294	0.364	0.258	0.431

Table 6.1: Results of classifiers with linear SMV kernel, Polynomial SVM kernel, Radial Basis Function (RBF) SVM kernel, additive χ^2 SVM kernel and KNN. Best results are underlined. Test details: HI-4, SI-7, SU-7, O-1, LBP-1

The results are shown in table 6.1. The classifiers were not trained with optimal parameters so it is not possible to deduct a classifier ranking based on this evaluation. Not surprisingly, the χ^2 SVM kernel performs best over all feature classifiers (except LBP) because in general, χ^2 kernels are best suited for histogram style feature vectors and those classifiers are all histogram classifiers. However, it should be noted that training and predicting a classifier with a χ^2 kernel can take more than 2 times longer than the same process with a polynomial kernel. Interestingly, the results for KNN were better than expected considering the fact that KNN is extremely overfitting on this dataset even for a high k. KNN achieves an accuracy of 0.960 on the training partition and only 0.363 on the test partition for SURF features for example. Even the worst SVM kernel do not overfitt that much.

6.1.2 Image Resolution

Different image resolutions were tested on SURF, SIFT and color histograms to determine if a higher image resolution automatically means a higher accuracy. Intuitively, image size or resolution should be a classical tradeoff between computational cost and classification performance. To test if higher resolutions yield better accuracies, 6 resolutions were tested from 1024 pixels (equals a 32×32 image) to 1,048,576 pixels (1024×1024). The results are visualized in figure 6.1.
Up to a certain point the accuracy of SIFT and SURF gets better until it drops and levels out. Since for every test run the number of extracted descriptors for the classification was the same, it is not the quantity of keypoints that affects the performance but the quality. SURF extracts fewer keypoints on small images. For the 1024 image size SURF

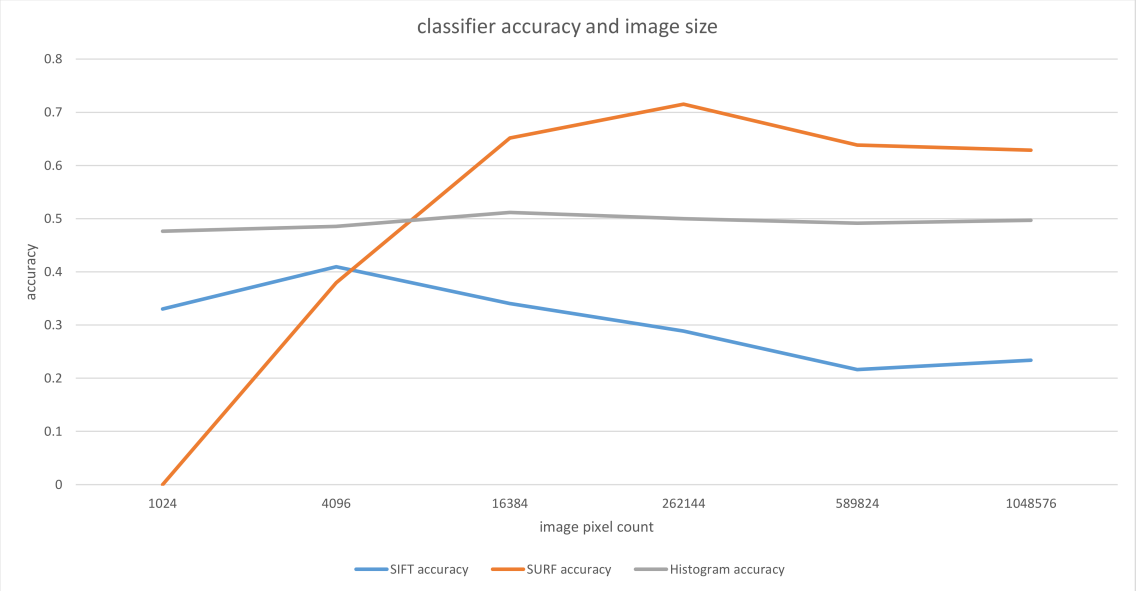
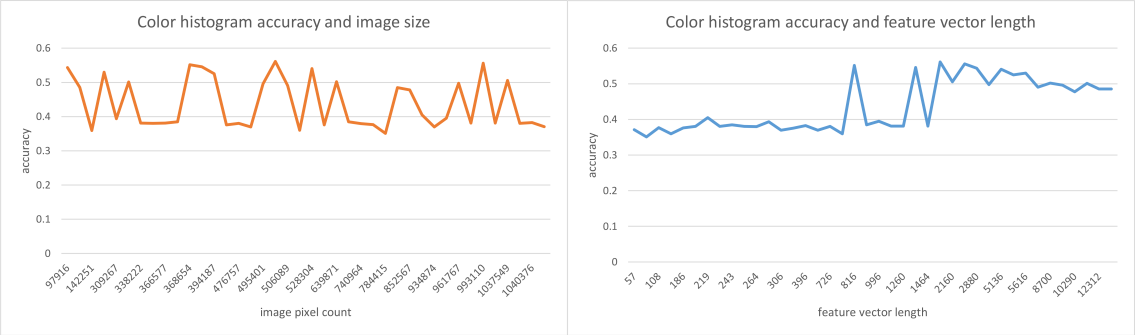


Figure 6.1: Impact of image size on histogram, SIFT and SURF classifier performance. Test details: HI-6, SI-6, SU-6

was not able to detect even one keypoint for most of the images which means that for smaller images the algorithm has to sample more random keypoints which are lower quality keypoints and decrease performance. SIFTs keypoint detector seems to have less problems with smaller images and increasing the image size only slightly increases performance. For big images the accuracy is decreasing again which may be due to fact that the number of extracted keypoints increases but the actual number of used keypoints remains the same so potentially more stable keypoints are filtered out.
Just changing the image size does not impact performance for histograms since the number of image segments does not change so the extracted histogram is always the same regardless of the image size. This is further proven by 40 random tests (HI-7) on the histogram parameters: number of histogram bins, image segments and the image size. Figure 6.2a shows that even with different numbers of bins or image segments the image size has no impact on performance at all. However, there is a significant positive correlation (correlation coefficient: $+0.538$, p-value: 0.00034) between the length of the feature vector and the accuracy (fig. 6.2b).



(a) Image size (b) Feature vector length

Figure 6.2: Histogram classifier performance under the impact of number of bins, image segments and image size. Test details: H-7

6.1.3 Number of Extracted Keypoints

The number of extracted keypoints is basically the length of the extracted feature vector. Each keypoint produces a 128 dimensional vector in case of SIFT and a 64 dimensional vector for SURF.
Deciding on a fixed number of keypoints is a very difficult problem because the quality and quantity of extractable keypoints is dependent on many various parameters like contrast and edge thresholds or image size. Figure 6.3 shows the change in accuracy for changes in the number of used keypoints. Other parameters are the same over all iterations. There is a positive correlation for both SIFT and SURF between number of keypoints and accuracy which means that up to a certain point adding more keypoints helps to improve performance.
Intuitively, the performance degrades after a certain point because the keypoint detectors can't extract enough relevant points of the region of interest (food item) so more irrelevant points of background objects have to be included. After the background points are all included the missing points have to be randomly augmented which should decrease performance even more.
The tests were performed with an image resolution of 16,384 pixels which corresponds to a size of 128×128. For this specific image size and the 8-class Food-100 subdataset SURF detects a mean of 105.3 keypoints per image (red circle) which means that after 105 keypoints in this test most of the other points were randomly sampled but the accuracy

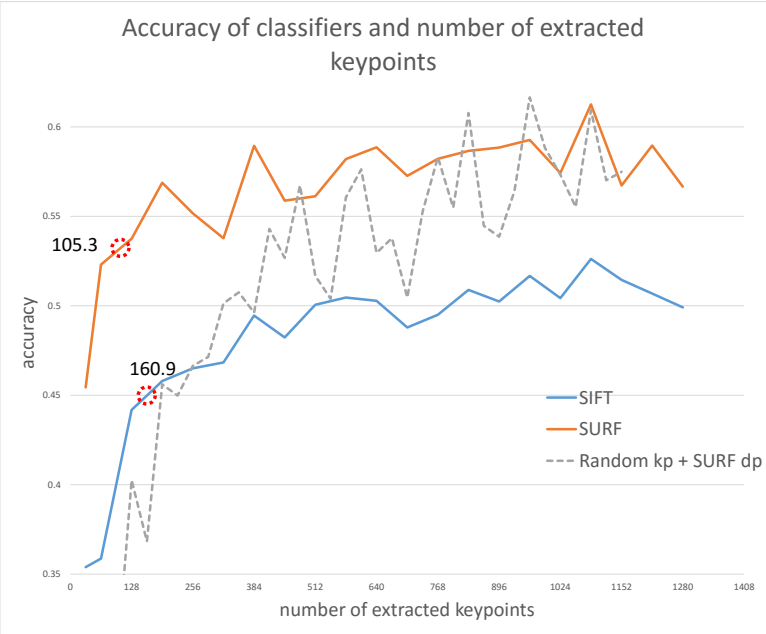


Figure 6.3: Impact of keypoint quantity on SIFT and SURF accuracy. The red circle denotes the number of mean extracted keypoints for this classifier. Test details: SI-1, SU-1, R-1

increased nonetheless. For SIFT the mean number of detected keypoints per image is 160.9. So the prior assumption that randomly sampled keypoints decrease performance is not completely correct. Though the gradient of the performance increment is much less after this point than the increment with real sampled keypoints before.
To compare the two keypoint extractors and further proof this assumption, a third and fourth test were performed. In these tests all keypoints were randomly sampled around the image center so no edge detection was performed. For keypoint description SURF was used in the third test and SIFT was used in the fourth test (not shown in figure 6.3). The gray dotted line in figure 6.3 shows the random test with the SURF feature descriptor. These tests make it possible to derive the following assumptions:

1. Detected keypoints allow for a higher accuracy than randomly sampled keypoints because it takes a long time for the random keypoint sampler to catch up with the performance of the SURF or SIFT classifier.

2. The random sampler does not use an edge detector. Only the SURF/SIFT descriptors are used. This makes it possible to compare the performance of the descriptors since it takes the detectors out of the equation. For this specific task it seems that SURF is not only the faster descriptor but also more accurate in describing an image.

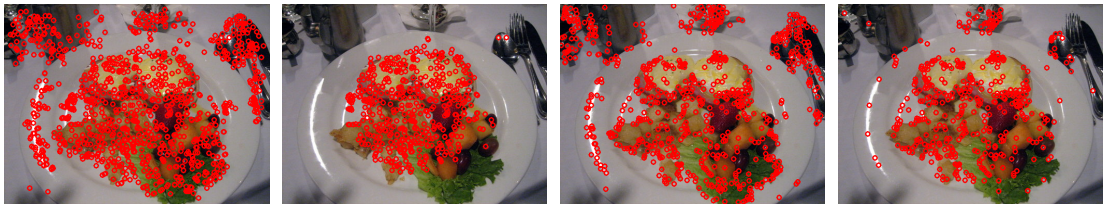
3. Random keypoint sampling is a positive alternative for images with low contrast values were detectors are not able to sample many keypoints.

6.1.4 Keypoint Filtering

	SIFT	SURF
Filtering	0.568	0.568
No filtering	0.440	0.599

Table 6.2: Results of keypoint filtering for SURF and SIFT. Test details: SU-3, SI-3

The algorithm proposed in section 4.4.3 filters keypoints by weighting their response values with the distance to the image center. As table 6.2 shows, this dramatically improves the performance of SIFT from 0.44 to 0.57. For SURF, however, the filtering does even reduce the accuracy although filtering SURF keypoints should also be applicable since the SURF keypoint detector produces even more false positive keypoints than SIFT which is shown in figure 6.4. So while this feature may not useful for SURF keypoints it is still effective for SIFT.



(a) 1102 unfiltered SURF keypoints (b) 500 filtered SURF keypoints (c) 835 unfiltered SIFT keypoints (d) 500 filtered SIFT keypoints

Figure 6.4: Comparison of SIFT and SURF keypoint filtering.

6.1.5 Bag of Words Dimension

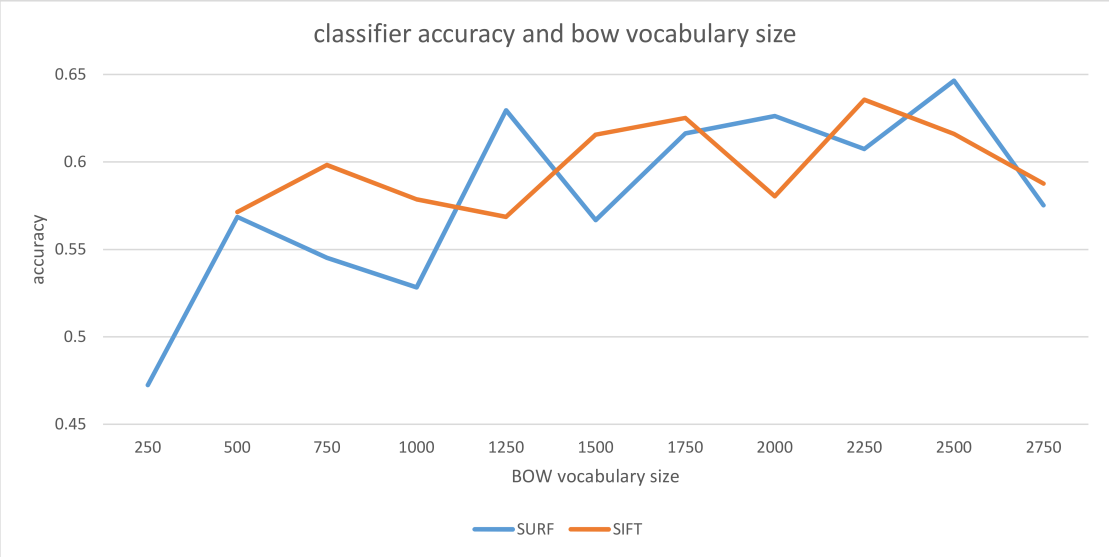


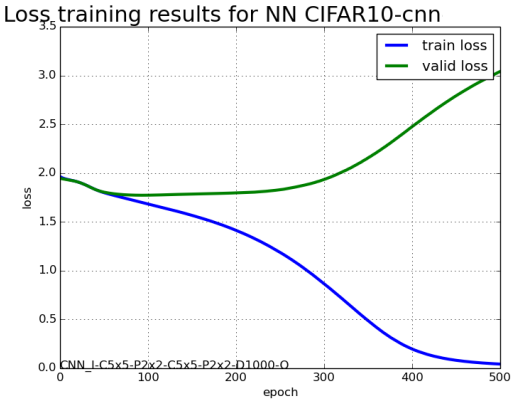
Figure 6.5: Impact of bow vocabulary size on SIFT and SURF classifier performance. Test details: SI-2, SU-2

The size of the bag of words vocabulary is a very important parameter because the BoW histogram is the actual feature vector that is used for classification and not the feature vector from classifiers like SIFT or SURF. A bad BoW vocabulary can lead to poor classification performance even if the input feature vectors are good. One way to influence the BoW vocabulary is its size. Figure 6.5 visualizes the impact of the vocabulary size on the overall performance of the classifier. The best accuracy was achieved with a vocabulary size of 2250 for SIFT and 2500 for SURF. For SURF there is a significant positive correlation (correlation: 0.695, p-value: 0.018) between the vocabulary size and the accuracy in the interval $[250, 2750]$. The correlation of SIFT is not significant enough (p-value: 0.063). Although the classifier performance for bigger vocabularies is higher, it is not feasible to use huge vocabularies since the training time also increases with an increasing BoW size.

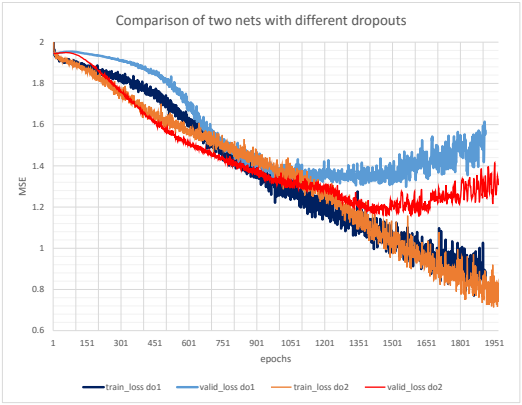
6.1.6 Neural Networks

Training and performance of neural networks hugely depends on the selected architecture. In the course of this thesis 9 models were evaluated on small datasets. On the one hand, small datasets allow for fast training but on the other hand, neural networks need as many samples as possible and the dataset the neural nets were tested on were 8 classes from the 50-data dataset which only contains 100 images per class. The result of a relatively small convolutional neural network is shown in figure 6.6a and could serve as a perfect example of overfitting. The training and validation error both decrease till epoch 80. After the 80th epoch the training error continues to decrease but the validation error stagnates and even begins to increase after the 250th epoch.

Dropout



(a) No dropout



(b) Two different dropout values

Figure 6.6: Three different dropout settings on a small net which was originally used for training on CIFAR-10.

A way to reduce overfitting is dropout (see section 3.3). Figure 6.6b shows the training loss of two networks with same architecture as in figure 6.6b only this time with dropout applied before and after the dense layers. The blue lines correspond to a network with 0.5 dropout applied and the red and orange lines represent a network with 0.6 dropout applied. This time the network overfits much later than before and until epoch 1000 the validation loss for the second dropout setting is even less than the

training loss. This is due to the fact that the dropout is only applied on the training data and not on the validation data.

Data Augmentation

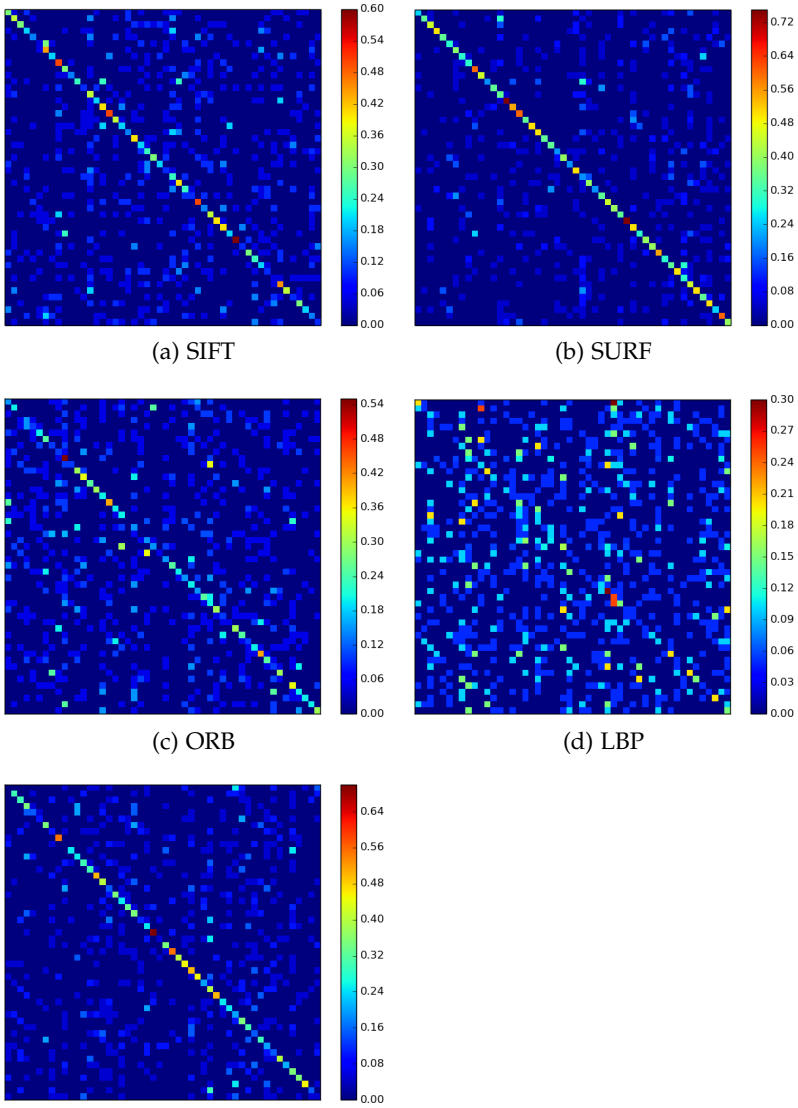
To further reduce overfitting and improve performance, augmentation on the data is performed (see section 4.3). Similar to the network with dropout in figure 6.6b, the validation loss is significantly lower than the testing loss. This is caused, because data augmentation is only applied on the training data. Depending on the CPU and GPU the additional time per epoch for data augmentation may vary but on the Azure virtual machine (see section 4.1) the average duration for each epoch increased from 5.41 to 24.1 seconds. Image value modifying operations like brightness or saturation adjustments are very time consuming. Disabling these operations can decrease the duration for each epoch from 24.1 back to 9.3 seconds and since this does not decrease the performance by that much (in this specific example the loss even decreased. This could be caused by a better weight initialization.), the tradeoff between performance and epoch duration is acceptable.
The training error for the network with no augmentation might not be that much higher but even after 1000 iterations the accuracy did not get better than the null score which indicates that the network did not learn at all and only predicted the majority class.

Table 6.3: Impact of data augmentation on the training of a small neural net on a 8-class dataset. All values are taken after 240 epochs of training.

	train loss (ACE)	valid loss (ACE)	duration (seconds)
Augmentation	2.069	2.066	24.116
Augmentation, no value changes	2.022	2.015	9.262
No Augmentation	2.295	2.295	5.412

6.2 Results for Food Classification

In this section the classifiers (except neural nets) are tested on the 50-data dataset and table 6.4 shows the results of this evaluation.



(a) SIFT (b) SURF

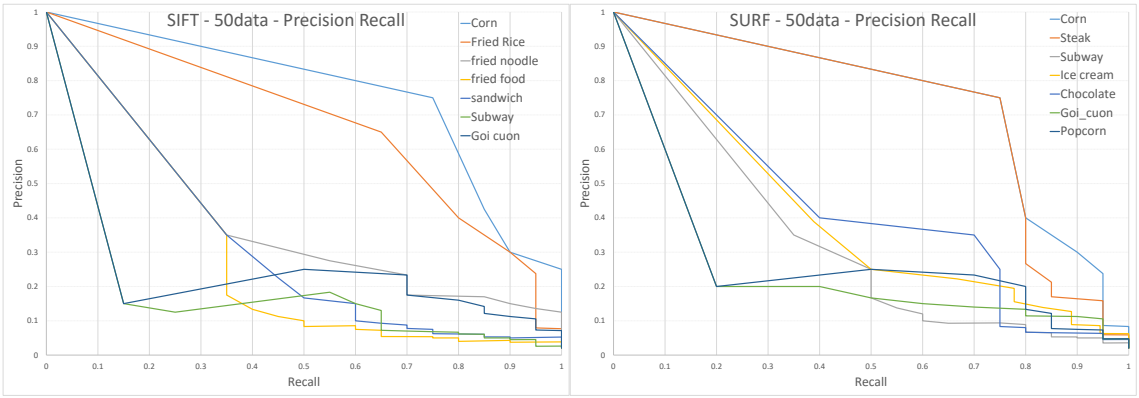
(c) ORB (d) LBP

(e) Color Histograms

Figure 6.7: Confusion matrix for Top-1 performance on the 50-data dataset validation segment.

Table 6.4: Evaluation results for feature classifier training on the 50-data dataset.

	Top-1 accuracy	Top-5 accuracy
SIFT + BoW + χ^2 SVM	0.3576	0.7203
SURF + BoW + χ^2 SVM	0.4028	0.7496
ORB + BoW + χ^2 SVM	0.2011	0.5136
CenSurE + BoW + χ^2 SVM	0.2217	0.7341
LBP + Poly SVM	0.119	0.3036
Color histograms + χ^2 SVM	0.2931	0.6715



(a) SIFT

(b) SURF

Figure 6.8: Precision recall curve for SIFT and SURF on the 50data dataset validation segment.

6.2.1 SIFT

The SIFT feature classifier was evaluated with an image area size of 262,144 pixels. 1088 weighted and augmented keypoints were extracted. This produced an feature vector with 139,264 dimensions. A bag of words vocabulary with the size of 2500 was used and the classification was done by a χ^2 SVM. Prior to these settings, a SIFT classifier with a much smaller image size and bow vocabulary was tested against the dataset but it only achieved an accuracy of 0.282. By increasing the image size the accuracy could be improved to 0.3576.

6.2.2 SURF

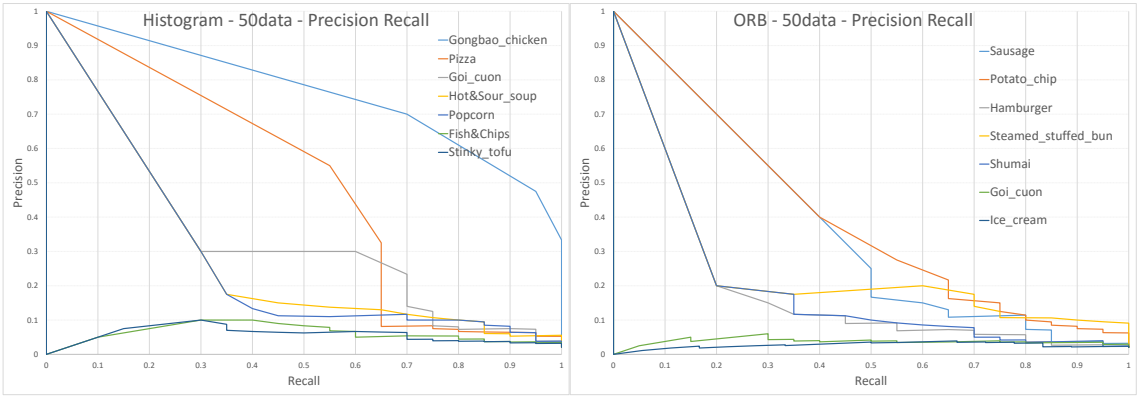
SURF was tested using an image size of 262,144 (corresponds to 512×512) pixels and 1088 keypoints per image so each image produces a 104,448 dimensional feature vector. The vectors are quantified in a 2500 bin bag of words. Keypoint filtering and keypoint sampling are both turned off. For classification a χ^2 SVM is used.
SURF achieves the best overall accuracy of 0.4028 which outperforms every other classifier.

6.2.3 Color Histogram

The histogram classifier was tested with an image size of 16384 pixels (128×128), 16 image segments, 150 histogram bins using the HSV color space and χ^2 SVMs. This model achieved a validation accuracy of 0.7285 on the 8-class parameter optimization dataset.

50-data

On the full 50-data dataset the histogram classifier achieved an average top-1 validation accuracy of 0.2931. This is 14.7 times better than the null score. The standard deviation of the validation accuracy across all 50 classes is 0.151.
Figure 6.9a shows the precision recall curve for the two best and worst classes as well as 3 classes around the median of the Top-1 validation accuracy. Figure 6.7f shows the confusion matrix for the histogram evaluation.
The histogram classifier had the most problems with "fish and chips" and "stinky tofu". The color of the class "Fish and chips" is very similar to the class "fried food" and "fish



(a) Color Histogram (b) ORB

Figure 6.9: Precision recall curve for color histograms and ORB on the 50-data dataset validation segment.

and chips" gets often confused with "fried food". The two classes with the highest accuracy are "pizza" and "gongbao chicken".

Food-10

The computational cost for the training of color histograms is very low. Therefore, this classifier was also tested on the much larger Food-100 dataset. The classifier was tested using 5-fold cross validation and achieved an average Top-1 accuracy on the validation set of 0.1758 with a maximum accuracy of 0.185.

6.2.4 ORB

ORB was tested with 1088 keypoints per image with both keypoint filtering and random point sampling turned on. A BoW with a vocabulary size of 2000 bins was used and images were loaded with a size of 262,144 pixels. For classification, a χ^2 SVM was utilized.
The model achieved an accuracy of 0.2011 with a standard deviation over the accuracy of 0.1027. The accuracy is only half as good as SURF features. Nonetheless, this is still a better result than expected. ORB uses BRIEF as a descriptor which is a binary-test descriptor but the clustering used for the vocabulary creation by the OpenCV library is

only indented to be used with floating point descriptors. So the descriptor clustering is not optimal for an ORB classifier which explains the poor performance.
Figure 6.9b shows the precision recall curve for ORB. "Sausages" and "Potato chips" achieved the highest accuracies and "Goi cuon" and "ice cream" achieved the lowest accuracies.

6.2.5 LBP

LBP achieved an accuracy of 0.119 which is only barely better than the null score. This is probably due to the fact that the basic LBP is a texture classifier and only contrast and illumination invariant. For food, however, rotation invariance is very important since food has no dominant orientation and plates can be freely rotated.

6.2.6 CenSurE

CenSurE was trained on images with the size 262,144 with the keypoint filtering turned off because the CenSurE API of skimage does not provide a keypoint response so the score can not be calculated. SURF was used as the keypoint descriptor. Therefore, the number of extracted keypoints and the size of the bag of words vocabulary are the same. Since this is also a classifier with a BoW, the χ^2 SVM kernel was used. The training of CenSurE took almost 33 hours and the top-1 accuracy is not very good. The top-5 accuracy, however, is very good and is actually the second best performance.

6.2.7 Neural Network

Three convolutional neural network architectures were used for evaluation. The first one is a deep CNN with 15 layers:

Input	$3 \times 48 \times 48$	
Convolutional Layer	$32 \times 46 \times 46$	5×5 filters
Max Pooling Layer	$32 \times 23 \times 23$	2×2 pooling
Dropout		0.3
Convolutional Layer	$64 \times 21 \times 21$	5×5 filters
Max Pooling Layer	$64 \times 10 \times 10$	2×2 pooling
Dropout		0.5
Convolutional Layer	$128 \times 8 \times 8$	3×3 filters
Max Pooling Layer	$128 \times 4 \times 4$	2×2 pooling
Dropout		0.6
Dense Layer	512	
Dropout		0.7
Dense Layer	512	
Dropout		0.7
Output Layer	6	

On 6 food classes this architecture achieved an accuracy of 0.2487 after 375 epochs and 7.5 hours of training.
The second network is a very small and compact 8-layer convolutional network:

Input	$3 \times 32 \times 32$	
Convolutional Layer	$20 \times 32 \times 32$	5×5 filters
Max Pooling Layer	$20 \times 16 \times 16$	2×2 pooling
Convolutional Layer	$20 \times 16 \times 16$	5×5 filters
Max Pooling Layer	$20 \times 8 \times 8$	2×2 pooling
Dropout		0.5
Dense Layer	1000	
Output Layer	8	

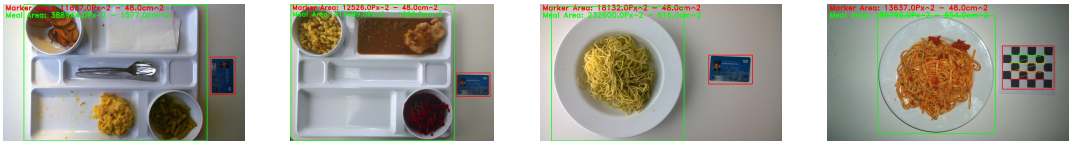
The advantage of this network is the fast training time and the ability to train it on smaller datasets. The network was trained on a 8 class subsample of the 50-data dataset and achieved a Top-1 accuracy of 0.4857 after 1912 epochs of training.

6.3 Results for Size Estimation

The image segmentation and size estimation modules are simple proof of concepts and were therefore only tested on images of menus from the canteen in Garching for the

multi food item segmentation as well as home cooked, single type meals for the meal segmentation. All images were taken using a 41 mega pixel smartphone camera and were resized to fit a resolution of 1000×564.

6.3.1 Single Meal Segmentation and Size Estimation



(a) canteen menu 1 (b) canteen menu 2 (c) single food 1 (d) single food 2

Figure 6.10: Meal region segmentation and size estimation. Calculated meal sizes: (a) $1577cm^2$, (b) $1586cm^2$, (c) $616cm^2$, (d) $654cm^2$

If the image of the food is taken from above with a clutter free and low contrast background the meal segmentation is reasonably fast and accurate. Almost all images were correctly segmented. Figure 6.10 shows 4 examples of the image segmentation with two canteen multi food menus measured with a custom marker and two single food images one with the custom marker and one with a chessboard marker. The difference between 6.10a and 6.10b is only $9(cm)^2$ and $38(cm)^2$ between the other two. Marker recognition is not a problem either although there are issues with low contrast markers or markers that are too small or not rectangular. Due to SIFT's rotational invariance the recognition of rotated markers is still accurate. In figure 6.10a the marker is rotated by 90°but the measurement and recognition is still accurate. Chessboard markers (figure 6.10d) are even more accurate than arbitrary markers since they provide more fixed reference points.

6.3.2 Multi Meal Segmentation and Size Estimation

Although generally the concept of multi meal region segmentation works, this feature is still a work in progress and should be further developed in future works. Figure 6.11 shows images of the current state of the multi meal region segmentation. The problem here is, that dark or very bright colors are sometimes not correctly filtered. In addition there are issues with overlapping bounding boxes that can cause objects to be

(a) original image (cropped) (b) filtered image (c) filtered image (d) filtered image (e) filtered image

Figure 6.11: Color filtering of 4 food items.

recognized multiple times because if two colors are to similar to each other they filter the same object.

7 Conclusion

In the course of this bachelor's thesis the foundation for future food recognition systems was built from ground up. The developed application is able to use different classifiers for food classification, segmentation and size estimation.
In addition, several classifiers were benchmarked to indicate models that are useful for the specific task of food recognition. Out of 7 models that were tested SURF-features in combinations with χ^2 SVMs achieved the highest accuracy with 40.3%. If the classifier is able to suggest 5 possible food items the accuracy increases to 75%. It was shown that despite the simplicity of color histograms this classification method is important for food recognition because color is an integral part of food.
Convolutional neural nets were also tested but the accuracy was far below the accuracy of traditional feature classifiers and there are three reasons for that:

1. **Network architecture**: The training of neural networks takes a very long time and is potentially very expensive. Very deep neural networks like the recent winner of the of ILSVRC are not possible to train without very powerful hardware. The experiments on neural networks that were conducted for this thesis used a Nvidia Grid K520 GPU with 3072 CUDA cores which is the highest amount of CUDA cores in a graphics card. Even with this powerful GPU the training of a 20 layer CNN took about 60 minutes per epoch. Even smaller networks with 15 layers took about 72 seconds per iteration which adds up to 40 hours of training for a single network.

2. **Dataset Size and Time**: This is also a problem with the training time. Deep neural networks need huge datasets but actually training these datasets increases the epoch duration even more which made it not feasible to utilize Food-101 for this thesis. One epoch of training a 15-layer CNN with the full Food-101 dataset took almost 70 minutes.

3. **Dataset size**: Even with booth Food-101 datasets combined the number of images per class is still not sufficient enough to train huge neural nets from scratch.

Recent publications all used CNNs that were pretrained on ImageNet because ImageNet contains more than 140 times more images than Food-101.

7.1 Future Work

This thesis may form a corner stone for future food classification systems that extend the methods that were used for this work. There are many possible ways to expand the system and increase performance.
Convolutional neural networks play a very important role in image recognition and machine learning in general so it is only logical to further extend the use of neural networks. Due to the high computational cost and the restricted time frame very few tests on neuronal networks could be performed. Pretraining networks is a good method to leverage neuronal networks as feature extractors and refine them with actual food images. It would also be very interesting to test if those pretrained networks can be refined with a combination of the ETHZ Food-101 and the UEC Food-101 dataset. This would double the size of the original Food-101 dataset which could potentially lead to better results than the current benchmark from the Im2Calories team [62]. In addition, it would be very easy to take advantage of the increasing amount of online machine learning APIs[1] which could be used to filter non-food items from the Food-101 datasets.

It may also be crucial to further investigate the impact of preprocessing techniques on classifier performance. In the course of this thesis histogram equalization, adaptive histogram equalization, Zero-Phase Component Analysis (ZCA) whitening, normalization and z-score normalization were implemented and added. It is very common to normalize data before neural net training but for the networks that were tested in this thesis, normalization did not improve performance. Due to the lack of time it was not possible to study the exact effects these preprocessing techniques have on classifiers.

Another way to significantly improve classification performance is the use of multiple kernel learning, boosting and late fusion. The latter was implemented in this thesis in the form of a uniform majority voter, an early fusion classifier that combined multiple feature vectors which was discarded because of poor performance and a late

[1] Microsoft recently released a free set of machine learning APIs which also include a computer vision API. This API can distinguish food and non-food items very well. https://www.microsoft.com/cognitive-services/en-us/computer-vision-api

fusion classifier that combined the output scores of SVMs to form a final decision.

In addition, new features should be evaluated. The performance of the histogram classifier especially in comparison with SIFT has shown that color is very important for food classification. Most of the edge feature classifiers work on grayscale images so it would be interesting to test if edge classifiers that combine color information and structure like C-SIFT actually improve performance.

Another field for improvement is the segmentation and size estimation algorithms. The proposed algorithm is only a proof of concept and was not the focus of this thesis. Therefore it would be valuable to refine this process with more advanced algorithms that can approximate depth information or even with 3D cameras which estimate volume.

Acronyms

ACE Average Cross-Entropy Error.

API Application Programming Interface.

ATM Amazon Mechanical Turk.

BoW Bag of Words.

BRIEF Binary Robust Independent Elementary Features.

CenSurE Center Surround Extrema.

CIFAR Canadian Institute for Advanced Research.

CNN Convolutional Neural Network.

CPU Central Processing Unit.

CRF Conditional Random Field.

csv Comma Separated Values.

CUDA Compute Unified Device Architecture.

DoG Difference of Gaussians.

ETHZ Eidgenössische Technische Hochschule Zürich.

FAST Features from Accelerated Segment Test.

FRIDa FoodCast Research Image Database.

GB Giga Bytes.

GPS Global Positioning System.

GPU Graphics Processing Unit.

HOG Histogram of Oriented Gradients.

HSV Hue, Saturation, Value.

HTML Hypertext Markup Language.

hyponym children of a synset.

ILSVRC ImageNet Large Scale Visual Recognition Challenge.

IO Input / Output.

KNN K-nearest Neighbors.

LBP Local Binary Patterns.

LoG Laplacian of Gaussian.

MSE Mean Squared Error.

NRLBP Non Redundant Local binary patterns.

ORB Oriented FAST and Rotated BRIEF.

PFID Pittsburgh Fast Food Image Dataset.

PHOW Pyramid Histogram of Visual Words.

PRICoLBP Pairwise Rotation Invariant Co-Occurrence Local Binary Pattern.

RAM Random Access Memory.

RBF Radial Basis Function.

RGB Red Green Blue.

SIFT Scale-invariant Feature Transform.

std Standard Deviation.

SURF Speeded up Robust Features.

SVM Support vector machine.

synset Synonym Set.

TADA Technology Assisted Dietary Assessment.

TFC TUM Food Cam.

TUM Technische Universität München.

UEC University of Electro Communications.

UPMC Université Pierre et Marie Curie.

URL Uniform Resource Locator.

US United States.

wnid WordNet ID.

ZCA Zero-Phase Component Analysis.

List of Figures

List of Tables

Bibliography

[1] *ABOUT | OpenCV http://opencv.org/about.html*.

[2] M. Agrawal, K. Konolige, and M. R. Blas. "CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching." In: *Eccv08* (2008), IV: 102–115. DOI: 10.1007/978-3-540-88693-8{_}8.

[3] R. Batuwita and V. Palade. "Class Imbalance Learning Methods for Support Vector." In: *Imbalanced Learning: Foundations, Algorithms, Applications* (2013), pp. 83–100. DOI: 10.1002/9781118646106.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. "Speeded up robust features." In: *Lecture notes in computer science* 3951 (2006), p. 14. ISSN: 03029743. DOI: 10.1007/11744023{_}32.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. "Speeded-Up Robust Features (SURF)." In: *Computer Vision and Image Understanding* 110.3 (June 2008), pp. 346–359. ISSN: 10773142. DOI: 10.1016/j.cviu.2007.09.014.

[6] O. Beijbom, N. Joshi, D. Morris, S. Saponas, and S. Khullar. "Menu-Match: Restaurant-Specific Food Logging from Images." In: *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, Jan. 2015, pp. 844–851. ISBN: 978-1-4799-6683-7. DOI: 10.1109/WACV.2015.117.

[7] J. Bergstra and Y. Bengio. "Random Search for Hyper-Parameter Optimization." In: *Journal of Machine Learning Research* 13 (2012), pp. 281–305. ISSN: 1532-4435.

[8] V. Bettadapura, E. Thomaz, A. Parnami, G. D. Abowd, and I. Essa. "Leveraging Context to Support Automated Food Recognition in Restaurants." English. In: *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, Jan. 2015, pp. 580–587. ISBN: 978-1-4799-6683-7. DOI: 10.1109/WACV.2015.83.

[9] J. Blechert, A. Meule, N. A. Busch, and K. Ohla. "Food-pics: An image database for experimental research on eating and appetite." In: *Frontiers in Psychology* 5.JUN (2014), pp. 1–10. ISSN: 16641078. DOI: 10.3389/fpsyg.2014.00617.

[10] D. S. Bloomberg. *Color quantization using modified median cut*. 2008.

[11] M. Bosch, F. Zhu, N. Khanna, C. Boushey, and D. Edward. "Combining global and local features for food identification in dietary assessment." In: *IEEE Trans Image Process. 2011 ; 2011: 1789–1792.* 15.10 (2014), pp. 1203–1214. ISSN: 08966273. DOI: 10.1016/j.drugalcdep.2008.02.002.A. arXiv: NIHMS150003.

[12] B. E. Boser, I. M. Guyon, and V. N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers." In: *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory* (1992), pp. 144–152. ISSN: 0-89791-497-X. DOI: 10.1.1.21.3818. arXiv: arXiv:1011.1669v3.

[13] L. Bossard, M. Guillaumin, and L. Van Gool. "Food-101 – Mining Discriminative Components with Random Forests." In: *Computer Vision - ECCV 2014* (2014), pp. 446–461. ISSN: 978-3-319-10598-7. DOI: 10.1007/978-3-319-10599-4{_}29. arXiv: 978-3-319-10599-4{_}29 [10.1007].

[14] C. Boushey, D. Edward, D. S. Ebert, K. D. Lutes, and D. Kerr. *Dietary Assessment System and Method*. May 2010.

[15] G. Bradski et al. "The opencv library." In: *Doctor Dobbs Journal* 25.11 (2000), pp. 120–126.

[16] L. E. Burke, J. Wang, and M. A. Sevick. "Self-Monitoring in Weight Loss: A Systematic Review of the Literature." In: *Journal of the American Dietetic Association* 111.1 (Jan. 2011), pp. 92–102. ISSN: 00028223. DOI: 10.1016/j.jada.2010.10.008.

[17] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "BRIEF : Binary Robust Independent Elementary Features." In: *European Conference on Computer Vision (ECCV)* (2010), pp. 778–792. ISSN: 978-3-642-15560-4. DOI: 10.1007/978-3-642-15561-1{_}56.

[18] J. Canny. "A Computational Approach to Edge Detection." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. ISSN: 01628828. DOI: 10.1109/TPAMI.1986.4767851.

[19] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs." In: (Dec. 2014). arXiv: 1412.7062.

[20] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, and J. Yang. "PFID: pittsburgh fast-food image dataset." In: *ICIP'09 Proceedings of the 16th IEEE international conference on Image processing*. IEEE Press, Nov. 2009, pp. 289–292. ISBN: 978-1-4244-5653-6.

[21] M. Chen, Y. Yang, C. Ho, and S. Wang. "Automatic chinese food identification and quantity estimation." In: *SIGGRAPH Asia 2012 ...* (2012).

[22] S. Christodoulidis and M. Anthimopoulos. "Food Recognition for Dietary Assessment Using Deep Convolutional Neural Networks." In: *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*. Vol. 9281. 2015, pp. 458–465. ISBN: 978-3-319-23221-8. DOI: 10.1007/978-3-319-23222-5.

[23] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. "Visual categorization with bags of keypoints." In: *Proceedings of the ECCV International Workshop on Statistical Learning in Computer Vision* (2004), pp. 59–74. ISSN: 18703453. DOI: 10.1234/12345678.

[24] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection." In: *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1* (2005), pp. 886–893. ISSN: 1063-6919. DOI: 10.1109/CVPR.2005.177. arXiv: 9411012 [chao-dyn].

[25] S. Dieleman, M. Heilman, J. Kelly, M. Thoma, D. K. Rasul, E. Battenberg, H. Weideman, S. K. Sønderby, Instagibbs, Britefury, C. Raffel, J. Degrave, Peterderivaz, Jon, J. D. Fauw, Diogo149, D. Nouri, J. Schlüter, D. Maturana, CongLiu, E. Olson, B. McFee, and Takacsg84. "Lasagne: First release." In: (Aug. 2015). DOI: 10.5281/zenodo.27878.

[26] A. Dosovitskiy, J. T. Springenberg, and T. Brox. "Unsupervised feature learning by augmenting single images." In: (Dec. 2013), p. 7. arXiv: 1312.5242.

[27] K.-B. Duan and S. S. Keerthi. "Which Is the Best Multiclass SVM Method? An Empirical Study." In: *Multiple Classifier Systems* 3541 (2005), pp. 278–285. ISSN: 0302-9743 (Print) 1611-3349 (Online). DOI: 10.1007/b136985.

[28] D. Eigen and R. Fergus. "Predicting Depth, Surface Normals and Semantic Labels With a Common Multi-Scale Convolutional Architecture." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2650–2658.

[29] E. Falkenauer. "On method overfitting." In: *Journal of Heuristics* 287 (1998), pp. 1–6. ISSN: 13811231.

[30] G. M. Farinella, D. Allegra, and F. Stanco. "A Benchmark Dataset to Study the Representation of Food Images." In: *European Conference Computer Vision Workshops* (2014).

[31] F. Foroni, G. Pergola, G. Argiris, and R. I. Rumiati. "The FoodCast research image database (FRIDa)." In: *Frontiers in human neuroscience* 7.March (2013), p. 51. ISSN: 1662-5161. DOI: 10.3389/fnhum.2013.00051.

[32] R. Funayama, H. Yanagihara, L. Van Gool, T. Tuytelaars, and H. Bay. *ROBUST INTEREST POINT DETECTOR AND DESCRIPTOR*. 2012.

[33] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." In: *Arxiv.Org* 7.3 (2015), pp. 171–180. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2013.00124. arXiv: 1512.03385.

[34] H. Hoashi, T. Joutou, and K. Yanai. "Image Recognition of 85 Food Categories by Feature Fusion." English. In: *2010 IEEE International Symposium on Multimedia*. IEEE, Dec. 2010, pp. 296–301. ISBN: 978-1-4244-8672-4. DOI: 10.1109/ISM.2010.51.

[35] J. F. Hollis, C. M. Gullion, V. J. Stevens, P. J. Brantley, L. J. Appel, J. D. Ard, C. M. Champagne, A. Dalcin, T. P. Erlinger, K. Funk, D. Laferriere, P.-H. Lin, C. M. Loria, C. Samuel-Hodge, W. M. Vollmer, and L. P. Svetkey. "Weight loss during the intensive intervention phase of the weight-loss maintenance trial." In: *American journal of preventive medicine* 35.2 (Aug. 2008), pp. 118–26. ISSN: 0749-3797. DOI: 10.1016/j.amepre.2008.04.013.

[36] C.-W. Hsu and C.-J. Lin. "A comparison of methods for multiclass support vector machines." In: *IEEE Transactions on Neural Networks* 13.2 (2002), pp. 415–425. ISSN: 1045-9227. DOI: 10.1109/72.991427.

[37] F. inc. *Introduction to FoodLog*. 2013.

[38] T. Joutou, K. Yanai, and T. Joutou. "A Food Image Recognition System With Multiple Kernel Learning." In: *IEEE International Conference on Image Processing (ICIP)* (2009), pp. 285–288. ISSN: 9781424456543. DOI: 10.1109/ICIP.2009.5413400.

[39] H. Kagaya, K. Aizawa, and M. Ogawa. "Food Detection and Recognition Using Convolutional Neural Network Food Detection and Recognition Using Convolutional Neural Network." In: *ACM Multimedia*. August. 2015. ISBN: 9781450330633. DOI: 10.13140/2.1.3082.1120.

[40] Y. Kajiwara, M. Nakamura, and H. Kimura. "Classification of Single-Food Images by Combining Local HSV-AKAZE Features and Global Features." In: *International Research Journal of Computer Science (IRJCS)* 2.1 (2015), pp. 12–17.

[41] Y. Kawano and K. Yanai. "Automatic expansion of a food image dataset leveraging existing categories with domain adaptation." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8927 (2015), pp. 3–17. ISSN: 16113349. DOI: 10.1007/978-3-319-16199-0{_}1.

[42] Y. Kawano and K. 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.

[43] Y. Kawano and K. Yanai. "FoodCam: A real-time mobile food recognition system employing Fisher Vector." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8326 LNCS.PART 2 (2014), pp. 369–373. ISSN: 03029743. DOI: 10.1007/978-3-319-04117-9{_}38.

[44] J. M. Keller and M. R. Gray. "A Fuzzy K-Nearest Neighbor Algorithm." In: *IEEE Transactions on Systems, Man and Cybernetics* SMC-15.4 (1985), pp. 580–585. ISSN: 21682909. DOI: 10.1109/TSMC.1985.6313426.

[45] N. Khanna, C. J. Boushey, D. Kerr, M. Okos, D. S. Ebert, and E. J. Delp. "An overview of the Technology Assisted Dietary Assessment project at Purdue University." In: *Proceedings - 2010 IEEE International Symposium on Multimedia, ISM 2010* (2010), pp. 290–295. DOI: 10.1109/ISM.2010.50.

[46] S. Kim. "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. DOI: 10.1109/JSTSP.2010.2051471.The.

[47] T. Kim. *Food Image Recognition : Boosting with Shallow Handcrafted Feature and Deep Convolutional Feature*. 2014.

[48] K. Kitamura, C. D. Silva, T. Yamasaki, and K. Aizawa. "Image processing based approach to food balance analysis for personal food logging." In: *Multimedia and Expo (ICME), 2010 IEEE International Conference on* (2010), pp. 625–630. ISSN: 1945-7871. DOI: 10.1109/ICME.2010.5583021.

[49] K. Kitamura, T. Yamasaki, and K. Aizawa. "Food log by analyzing food images." In: *Proceeding of the 16th ACM international conference on Multimedia - MM '08*. New York, New York, USA: ACM Press, Oct. 2008, p. 999. ISBN: 9781605583037. DOI: 10.1145/1459359.1459548.

[50] K. Kitamura, T. Yamasaki, and K. Aizawa. "FoodLog: Capture, Analysis and Retrieval of Personal Food Images via Web." In: *Proceedings of the ACM multimedia 2009 workshop on Multimedia for cooking and eating activities - CEA '09* (2009), p. 23. DOI: 10.1145/1630995.1631001.

[51] M. Kremp. *Mobile World Congress 2016: Ist das wirklich ein Tricorder? - SPIEGEL ONLINE*. 2016.

[52] A. Krizhevsky. "Learning Multiple Layers of Features from Tiny Images." In: *... Science Department, University of Toronto, Tech. ...* (2009), pp. 1–60.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* (2012), pp. 1097–1105. ISSN: 10495258. arXiv: 1102.0183.

[54] D. Kumar, N. Thome, M. Cord, and F. Precioso. "Recipe recognition with large multimodal food dataset." In: *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* 1 (2015), pp. 1–6. DOI: 10.1109/ICMEW.2015.7169757.

[55] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2323. ISSN: 00189219. DOI: 10.1109/5.726791. arXiv: 1102.0183.

[56] T. Leung and J. Malik. "Representing and recognizing the visual appearance of materials using three-dimensional textons." In: *International Journal of Computer Vision* 43.1 (2001), pp. 29–44. ISSN: 09205691. DOI: 10.1023/A:1011126920638.

[57] M. B. E. Livingstone, P. J. Robson, and J. M. W. Wallace. "Issues in dietary intake assessment of children and adolescents." English. In: *British Journal of Nutrition* 92.S2 (Mar. 2007), S213. ISSN: 0007-1145. DOI: 10.1079/BJN20041169.

[58] D. G. Lowe. "Distinctive image features from scale-invariant keypoints." In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[59] D. G. Lowe. *Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image*. 2004.

[60] D. G. Lowe. "Object recognition from local scale-invariant features." In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. [8. IEEE, 1999, pp. 1150–1157. ISBN: 0-7695-0164-8. DOI: 10.1109/ICCV.1999.790410. arXiv: 0112017 [cs].

[61] Y. Matsuda, H. Hoashi, and K. Yanai. "Recognition of multiple-food images by detecting candidate regions." In: *Proceedings - IEEE International Conference on Multimedia and Expo* (2012), pp. 25–30. ISSN: 19457871. DOI: 10.1109/ICME.2012.157.

[62] A. Meyers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. P. Murphy. "Im2Calories: Towards an Automated Mobile Vision Food Diary." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1233–1241.

[63] D. T. Nguyen, Z. Zong, P. O. Ogunbona, Y. Probst, and W. Li. "Food image classification using local appearance and global structural information." In: *Neurocomputing* 140 (Sept. 2014), pp. 242–251. ISSN: 09252312. DOI: 10.1016/j.neucom.2014.03.017.

[64] C. L. Ogden, M. D. Carroll, L. R. Curtin, M. M. Lamb, and K. M. Flegal. "Prevalence of high body mass index in US children and adolescents, 2007-2008." In: *JAMA* 303.3 (Jan. 2010), pp. 242–9. ISSN: 1538-3598. DOI: 10.1001/jama.2009.2012.

[65] T. Ojala, M. Pietikainen, and D. Harwood. "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions." In: *Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR)* 1 (1994), pp. 582–585. ISSN: 00313203. DOI: 10.1109/ICPR.1994.576366.

[66] T. Ojala, M. Pietikainen, and T. Maenpaa. "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns." English. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (July 2002), pp. 971–987. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2002.1017623.

[67] T. Ojala and M. Pietikäinen. "Unsupervised texture segmentation using feature distributions." In: *Pattern Recognition* 32.3 (1999), pp. 477–486. ISSN: 00313203. DOI: 10.1016/S0031-3203(98)00038-7.

[68] F. Pedregosa and G. Varoquaux. "Scikit-learn: Machine learning in Python." In: *... of Machine Learning ...* 12 (2011), pp. 2825–2830. ISSN: 15324435. DOI: 10.1007/s13398-014-0173-7.2. arXiv: arXiv:1201.0490v2.

[69] D. S. Ponrani, S. N. Suveka, and S. K. Brabha. "Performance Analysis of SVM to Measure Calorie and Nutrition from Food Images." In: *International Journal of Advanced Research Trends in Engineering and Technology* 1.3 (2014), pp. 93–98.

[70] K. G. Prabu. "A Food Recognition System for Diabetic Patients using SVM Classifier." In: *International Journal of Advanced Technology in Engineering and Science* 3.2 (2015), pp. 371–378.

[71] Y. Probst, D. Nguyen, M. Tran, and W. Li. "Dietary Assessment on a Mobile Phone Using Image Processing and Pattern Recognition Techniques: Algorithm Design and System Prototyping." In: *Nutrients* 7.8 (2015), pp. 6128–6138. ISSN: 2072-6643. DOI: 10.3390/nu7085274.

[72] M. Puri, Z. Zhiwei, J. Lubin, T. Pschar, A. Divakaran, and H. S. Sawheney. *Food recognition using visual analysis and speech recognition*. July 2010.

[73] E. Rosten and T. Drummond. "Fusing Points and Lines for High Performance Tracking." In: ().

[74] E. Rublee and G. Bradski. "ORB - an efficient alternative to SIFT or SURF." In: (2011). ISSN: 1550-5499. DOI: 10.1109/ICCV.2011.6126544.

[75] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "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.

[76] W. Shimoda and K. Yanai. "CNN-based Food Image Segmentation without Pixel-Wise Annotation." In: *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*. Ed. by V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone. Vol. 9281. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 449–457. ISBN: 978-3-319-23221-8. DOI: 10.1007/978-3-319-23222-5.

[77] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

[78] L. P. Svetkey, V. J. Stevens, P. J. Brantley, L. J. Appel, J. F. Hollis, C. M. Loria, W. M. Vollmer, C. M. Gullion, K. Funk, P. Smith, C. Samuel-Hodge, V. Myers, L. F. Lien, D. Laferriere, B. Kennedy, G. J. Jerome, F. Heinith, D. W. Harsha, P. Evans, T. P. Erlinger, A. T. Dalcin, J. Coughlin, J. Charleston, C. M. Champagne, A. Bauck, J. D. Ard, and K. Aicher. "Comparison of strategies for sustaining weight loss: the weight loss maintenance randomized controlled trial." In: *JAMA* 299.10 (Mar. 2008), pp. 1139–48. ISSN: 1538-3598. DOI: 10.1001/jama.299.10.1139.

[79] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going Deeper with Convolutions." In: *arXiv preprint arXiv:1409.4842* (2014), pp. 1–12. ISSN: 1550-5499. DOI: 10.1109/ICCV.2011.6126456. arXiv: 1409.4842.

[80] A. C. G. Thome. "SVM Classifiers – Concepts and Applications to Character Recognition." In: *Advances in Character Recognition* (2012), pp. 25–49. DOI: 10.5772/2575.

[81] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and T. scikit-image contributors. "scikit-image: image processing in {P}ython." In: *PeerJ* 2 (2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453.

[82] K. Q. Weinberger, J. Blitzer, and L. K. Saul. "{D}istance {M}etric {L}earning for {L}arge {M}argin {N}earest {N}eighbor {C}lassification." In: *{A}dvances in {N}eural {I}nformation {P}rocessing {S}ystems (NIPS)* 18 (2005), pp. 1473–1480.

[83] W. Wen and Y. Jie. "Fast food recognition from videos of eating for calorie estimation." In: *Proceedings - 2009 IEEE International Conference on Multimedia and Expo, ICME 2009* (2009), pp. 1210–1213. ISSN: 1945-7871. DOI: 10.1109/ICME.2009.5202718.

[84] A. P. Witkin. "Scale-space filtering." In: *International Joint Conference on Artificial Intelligence* 2 (1983), pp. 1019–1022. DOI: 10.1109/ICASSP.1984.1172729. arXiv: /dl.acm.org/citation.cfm?id=1623607 [http:].

[85] World Health Organization. *WHO | Obesity and overweight*. 2015.

[86] L. Zepeda and D. Deal. "Think before you eat: photographic food diaries as intervention tools to change dietary decision making and attitudes." In: *International Journal of Consumer Studies* 32.6 (2008), pp. 692–698. ISSN: 14706423. DOI: 10.1111/j.1470-6431.2008.00725.x.

[87] Z. Zong, D. T. Nguyen, P. Ogunbona, and W. Li. "On the combination of local texture and global structure for food classification." In: *Proceedings - 2010 IEEE International Symposium on Multimedia, ISM 2010* (2010), pp. 204–211. DOI: 10.1109/ISM.2010.37.