CS 725: Foundations of Machine Learning Midterm Project Report

CV Mavericks*
23m2154, 23m2156, 23m2157, 23m2158, 23m2162

October 31, 2023

Abstract

Our project addresses the limitations of current food image datasets in accurately estimating calories. We introduce a food image dataset that includes both food annotations and detailed volume and mass records for each item, along with a calibration reference. Our dataset comprises 2978 images, each accompanied by annotations and essential food metrics. To estimate the calorie content of food in this dataset, we employ a deep learning approach utilizing Faster R-CNN for food detection and a GrabCut algorithm to outline the contours of each food item. With this information, we calculate the volume and subsequently, the calorie content of each food.

1 Problem Statement

Estimating the calorie content from food images requires the food image datasets which faces a critical challenge in accurately estimating calorie content due to the choice of object detection algorithm and volume estimation method. This project aims to fill this gap by using a detailed food image dataset provided by ECUST[1]. It includes precise food annotations, along with volume, mass records, and a reliable calibration reference (one Yuan coin) for each item. This dataset is a collection of 2978 images, each meticulously annotated and supplemented with necessary food metrics i.e. volume, density, calorie content etc. To achieve accurate calorie estimation, a deep learning methodology is employed. This involves the adaptation of faster Region-based Convolutional Neural Networks (RCNN) for precise food detection, followed by the implementation of the GrabCut algorithm to precisely outline the contours of each food item. The segmented images would be utilized to calculate the volume and, subsequently, the calorie content of each food item.

The project aims to address two key challenges in dietary assessment from food images:

Detection and Classification of Food Items: Accurate identification and categorization of food items from images represent the initial hurdle. Existing methodologies often fall short in providing precise detection and classification. This project seeks to leverage advanced techniques, particularly faster RCNN, to enhance the capability to not only detect but also classify food items within images. By employing faster RCNN, the system aims to achieve an improvement in the accuracy and efficiency of food item recognition.

*The project has undergone revisions based on the feedback provided on 18/10/2023. Initially, the project focused on *Image-Based Disease Diagnosis using Deep Learning*. The new project concept has received approval from Krishnakant Bhatt (21q050016@iitb.ac.in).

Estimation of Food Volume for Calorie Content Determination: The second significant challenge lies in estimating the volume of identified food items, a crucial step in determining their calorie content. Traditional approaches struggle to provide the necessary precision for this task. To overcome this limitation, the project proposes the integration of GrabCut, an advanced segmentation algorithm. By applying GrabCut, the system aims to accurately delineate the contours of each food item, enabling precise volume estimation. This, in turn, facilitates a more accurate and reliable assessment of calorie content.

By combining the faster RCNN for food item detection and classification with GrabCut for precise volume estimation, this project seeks to accurately estimate the calorie content from food images. The successful execution of this approach promises to benefit individuals seeking improved dietary choices, weight management, fitness goals, and specific dietary requirements.

2 Proposed Solution Approach

The solution methodology for estimating the calorie content from food images can be summarized as follows:

Object Detection: Our aim is to help people who want to keep track of the calories they consume. We would create a system that works with a smartphone. Before eating, users need to take pictures of their food from the top and side views. They should include a One Yuan coin in each picture. For the top view, we use faster RCNN to recognize the types of food and draw an outline or bounding box around them in the photos. We do the same for the side view. The primary goal of the faster RCNN is to develop a unified architecture that not only detects objects within an image but also locates the objects precisely in the image. It combines the benefits of deep learning, CNN, and region proposal networks(RPN) into a cohesive network, which significantly improves the speed and accuracy of the model. When we give it a picture, it gives us boxes around the objects. It also tells us what type of object it thinks it is.

Image Segmentation: Before figuring out the volume, we first break down each box into its separate parts. We would use something called GrabCut algorithm. It's a way to process images that helps us focus on specific areas. Normally, users would have to highlight areas themselves, but we've made it so the system can do this on its own. After this step, we get a precise outline or contour of each food item. Then, we can estimate the volume and calorie content. This process uses an image processing approach to segment each bounding box. As mentioned above, the bounding boxes around the object that GrabCut needs can be provided by faster RCNN. After segmentation, we can get a series of food images stored in matrix, but with the the values of the background pixels being replaced by zeros. This will leave only the foreground pixels.

Calorie Estimation: The user needs to provide the image clicked from both side and top view along with the One Yuan coin. According to the One Yuan coin detected in the top view, the true size of a pixel is known. Similarly, we know actual size of of a pixel in the side view. Then we use different formulas to estimate volume of each food. After getting volume, food's calorie is obtained by searching related calorie tables.

3 Code Survey

3.1 Object Detection in Faster RCNN

3.1.1 PyTorch

We have explored the official PyTorch documentation for using Faster RCNN model by importing from torch-vision and reference code to understand the use of functions implemented and the arguments to be passed. The GitHub link consists of source code of implementation of Faster RCNN in PyTorch used in documentation. The Medium article is walk-through of implementing Faster RCNN from scratch in PyTorch. The article implements the RPN network for creating anchor boxes and uses VGG16 as the backbone architecture. The PyTorch tutorial helps in understanding how to fine tune Pre-trained object detection models for custom dataset. The link fine tunes a mask RCNN model and explains how to change the underlying backbone architecture.

1. Faster RCNN in PyTorch
2. Source code of Faster RCNN in PyTorch
3. Faster RCNN from scratch
4. Fine tuning pre-trained RCNN model

3.1.2 TensorFlow

We have gone through the TensorFlow documentation on object detection API and explanation on different object detection models — one of them being Faster RCNN. The GitHub link consists of source code of Faster RCNN implemented by TensorFlow on different backbone architectures.

1. Faster RCNN in TensorFlow
2. Source code of Faster RCNN in TensorFlow

3.2 GrabCut Algorithm for Extracting Foreground from Bounding Boxes

The OpenCV documentation for understanding the GrabCut algorithm for foreground extraction and sample code for understanding its implementation.

3.3 Different Backbone Architectures Used for Implementing Faster RCNN

Different backbone architectures can be used for implementing Faster RCNN. Below are the links to the architectures we will be going through and their source code implementation in Keras. The study of different backbone architectures will help select the right model for our problem statement.

1. Resnet50
2. VGG16

3. MobileNet

4. InceptionResnetV2

3.4 Region Proposal Network Implementation

The key feature of implementing Faster RCNN is its Region Proposal Network (RPN). The article gives a detailed walk-through of implementing the RPN network while covering the important aspects of its implementation which will help us understand the network and allow us the ability to make changes as per our need.

4 Datasets

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

5 Implementation Details

5.1 Faster RCNN

This algorithm takes an image as input and produces a convolution feature map. It utilizes a separate network to suggest potential regions of interest. These suggested regions are then reshaped using a Region of Interest (ROI) pooling layer. Offset values for bounding boxes are predicted based on this reshaping, and the images within the proposed regions are classified. Faster RCNN is chosen for image detection due to its superior speed compared to other algorithms like RCNN and Fast RCNN. Anchors play a crucial role in Faster R-CNN. An anchor is essentially a box, and in the default setup of Faster R-CNN, an image is divided into 9 anchors. Using a well-chosen set of anchors can enhance both speed and accuracy. The Region Proposal Network (RPN) produces a set of boxes or proposals, which are then examined by the classifier and regressor to identify objects.

5.2 GrabCut Algorithm

The GrabCut algorithm employs Image Segmentation. In this method, we use a box to define the object, effectively separating it from the background. The outer region of the box represents the background, while the inner portion combines some background and the object. Subsequently, an iterative process designates each pixel as either background or foreground. Initially, the computer makes an educated guess based on user-provided data. Then, we utilize a Gaussian mixture model to characterize both the foreground and background. Next, we label pixels as probable foreground or probable background, designating others as unknown.

In this algorithm, every pixel becomes a node in a graph. Additional nodes like source and sink nodes are introduced. The source node connects to the foreground pixels, while the sink node connects to the background pixels. The likelihood of a pixel being foreground or background influences the edge weights connecting the pixels to these nodes. The similarity between pixels, indicated by the edge information, determines these weights. If there's a substantial color difference between pixels, a low weight is assigned to the edge. Following this, the graph undergoes Segmentation using min-cut. This operation divides the graph into two parts, separating the source and sink nodes, and does so with the least possible cost. The total weights of the cut edges constitute the cost function. Post-cut, all pixels connected to the Source node become foreground, while those linked to the Sink node become background. This process continues until the classification stabilizes.

5.3 Volume Calculation

To estimate the volume, we calculate the scale factors based on calibration objects. We use a One Yuan coin to show the specific process of calculating the volume. The diameter of the coin is 2.5 cm, and the side view's and top view's scale factor (α_S and α_T) is calculated with Equation 1 and 2.

$$\alpha_S = \frac{2.5}{(W_S + H_S)/2} \tag{1}$$

$$\alpha_T = \frac{2.5}{(W_T + H_T)/2} \tag{2}$$

Where W_S and H_S are the width and height of bounding box in side view. Moreover, W_T and H_T are the width and height of bounding box in top view.

Furthermore, we divide the foods into three categories based on shape: ellipsoid, column, irregular. Different volume estimation formula will be selected for different types of food, according to Equation 3.

$$v = \begin{cases} \beta \cdot \frac{\pi}{4} \cdot \alpha_S^3 \cdot \sum_{k=1}^{H_S} (L_S^k)^2, & \text{if shape is ellipsoid} \\ \beta \cdot s_T \cdot \alpha_T^2 \cdot H_S \cdot \alpha_S, & \text{if shape is column} \\ \beta \cdot s_T \cdot \alpha_T^2 \cdot \alpha_S \cdot \sum_{k=1}^{H_S} \left(\frac{L_S^k}{L_S^{max}} \right)^2, & \text{if shape is irregular} \end{cases} \tag{3}$$

Where L_S^k is the number of foreground pixels in side view of row k ($k \in 1, 2, 3, \ldots, H_S$). $L^{max} = \max_k L^k$, it records the maximum number of foreground pixels in side view. β is a compensation factor (default value = 1.0). s_T is the surface area of the top view.

After estimating the volume, the next step is to estimate each food's mass (m in g). It can be calculated in Equation 4, Where v represents the volume of food in cm^3, and ρ represents its density value in g/cm^3. Then the calorie (C in cal) of the food can be calculated by Equation 5 using the unit energy (c in cal/g).

$$m = \rho \cdot v \tag{4}$$

$$C = c \cdot m \tag{5}$$

The density, unit energy values and shape for each food items can be obtained from ECUST dataset.

6 Preliminary Results

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl.

Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

7 Roadmap - Soumen

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.[2][3]

References

[1] Yanchao Liang and Jianhua Li. “Computer vision-based food calorie estimation: dataset, method, and experiment”. In: *arXiv preprint arXiv:1705.07632* (2017).

[2] Andrew Ng. *Deep Learning Specialization.* 2017. URL: `https://youtube.com/playlist?list=PLpFsSf5Dm-pd5d3rjNtIXUHT-v7bdaEIe&si=pYyzJQk7P5FFmqGK`.

[3] Aston Zhang et al. *Dive into Deep Learning.* 2021. URL: `https://d2l.ai/`.