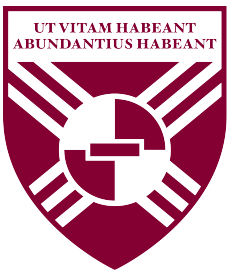

AMERICAN
UNIVERSITY OF BEIRUT

MAROUN SEMAAN FACULTY OF
ENGINEERING & ARCHITECTURE

EECE 490 Project Report

Hossam Mostafa and Paolo Hadaie

April 24, 2025

Contents

1 Project Overview and Timeline

We started developing our project by making use of a pre-trained model that is MobileNetV2 as the backbone for food image classification. This model was initialized without the top layers and pretrained on ImageNet purposes so that transfer learning can be exploited. For training and fine-tuning the model, we considered the complete Food-101 dataset that has 101 classes of varieties of food. Various data augmentation, for instance, random flipping and rotation, was applied to improve generalization capacity. Despite the powerful base model and a long training through Google Colab using the GPU, the training process from the full dataset proved to be very time-consuming and computationally expensive. The model reached a plateau at around 43 % validation accuracy, which is not considered to be effective for real-world food classification operations. This was solved by training over multiple stages of the model, saving the model after each phase, and using callbacks such as EarlyStopping, ModelCheckpoint, and ReduceLROnPlateau for performance optimization and mitigation of overfitting. Thus, we shifted attention to Roboflow and began creating a custom dataset using photos that we had personally taken with our mobile phones. This move was to simulate the real user's experience of working with CAMHEALTH as closely as possible. Over two days, we manually labeled and annotated each image with bounding boxes and then trained using a YOLOv11 model. Unfortunately, although we achieved about 80% accuracy, the limited dataset consisted of just 7 food classes, which were not sufficient for our intended objectives. Therefore, we pivoted to a classification approach on Roboflow with an image selection from the Food-101 dataset. We curated 40 food classes with about 1,000 images each for training on a fine-tuned Vision Transformer (ViT). After several hours of training, we achieved an accuracy of 99%.This model became the core of CAMHEALTH. To further ensure reliability, we inserted one more prompt for the users to either reaffirm the predicted food or select the correct one in case it was misclassified from the dropdown menu.

Next, we focused on relating every confirmed food item with its calorie count. As such, to make the system more practical and flexible, we integrated different units for each food type converting grams directly into calories or using pre-defined selections like slice, cup, or tablespoon. The list of considered units was very large, including the cross-calibration of all these units among different kinds of foods. Following the entry of the main dish, users are enabled to add in any other extra ingredients to further customize the calorie calculation to actual meals. Last but not least, the OpenAI API was integrated to realize our vision of a chatbot that converses in return with nutrition feedback on calorie count. Other than the computational output, the chatbot answers specific questions regarding micronutrients, meal planning, or general dietary consulting. We also implemented safeguards to redirect or reject non-food-related queries, ensuring a focused and streamlined user experience. With that, CAMHEALTH was brought to life; transforming from a simple concept into a fully functioning Streamlit application.

2 Abstract

CAMHEALTH is a prototype of Streamlit web application that would provide estimates of the calories consumed by a user by a lively interaction with an AI-enhanced system. Its design uses both computer vision for food identification (Roboflow), and natural language processing with the help of openAI Assistants API. This application is intended to act as a substitute for the usually longer manual calorie documentation. All that is required of the user is to upload a photo of a meal, approve or disprove the food item nominated by AI, define portion sizes using easy-to-follow measurements, and receive a detailed calorie breakdown through a natural, chat-like interaction with an AI assistant. The subsequent report will outline the system architecture and core functionalities, technology stack, data management strategy, challenges, and prospects.

3 Introduction

Tracking calories manually is what a lot of people do when trying to live a healthy lifestyle, fitness-wise, or for allergy or disease reasons. However, this is very slow, needs in-depth knowledge of food databases due to extensive manual intervention in estimating portions, and it frequently results in rather imprecise estimates. CAMHEALTH aims towards integrating AI technologies automatically to realize food identification, which will reflect in facilitating user input in an interface made inviting and familiar under conversation. Such solutions would be the primary objective in making a more fun and engaging experience for users when trying to attribute calorie content to what they have recently consumed. This document presents the construction and operation of the prototype CAMHEALTH system.

4 System Architecture

Taking into consideration the CAMHEALTH application, it is built based on modular architecture composed of three layers:

4.1 Presentation Layer (Streamlit UI - stream.py):

The layer that includes all interactions with the user. Here, the user interacts with a multi-step web interface for image upload, confirmation, quantification, and results through the application of Streamlit framework. It also manages the state of the application and orchestrates calls to the backend logic.

4.2 Backend Logic Layer (Utilities - ut.py):

This is the core Python module with the processing functions, interfacing logic for: Loading and accessing processed calorie/portion data; Generating dynamic unit options based on food items; Converting user-provided quantities (in different units) into grams; Calculating calorie estimates based on gram weights and database looksups; Interfacing with external AI APIs (Roboflow and OpenAI).

4.3 External AI Services:

Roboflow: Food classification based on images. A specific pre-trained model (food-101-ih2pp/4) which is the classifier based on a fine-tuned vision Transformer(ViT) is invoked through their API. OpenAI: The Assistants API provides the conversation part of the program. A dedicated Assistant (asst _ Qwxm9bSS2gu771Cqx3jG46B6) handles conversations with users at different stages in the process, giving an answer based on the context and is used to answer food-related questions, and if the user asks something unrelated to food, built-in safeguards (or "guards") are triggered to handle such cases appropriately. Data Layer: Consists of: calorie _ table _ processed.csv: A pre-processed csv file consisting of food labels, calories per gram, small/medium/large portion size estimates in grams, generated by csv.py. CONVERSIONS Dictionary (in ut.py): A hard-coded dictionary built to detail unit-to-gram conversions for many specific food items and units.

Workflow: The application takes the user through a sequential process where he uploads-¿predicts-¿confirms-¿quantifies main item-¿adds extras (optional)-¿calculates and summarizes-¿opens chat. State is determined within Streamlit's session state, while AI services are called at specific steps for identification and conversational context.

5 Core Features

Photo-Based Food Upload: Meal photos are uploaded by users to initiate the process. Food Identification by AI: The application sends the image to the Roboflow API for its AI to predict the food item present. Interactive Confirmation: The suggested item by AI is presented to the user for confirmation or manual item selection from a long list based on calorie database. Flexible Quantification: To quantitate portion, a user can qualify the units according to the food item, such as weights (grams, oz), volumes (tsp, tbsp), or qualitative/piece-based information like slice, scoop, small, medium, large, etc. Multi-Item Meal Composition: Adding items like sides, toppings, or drinks to the servant container allows obtaining a more all-comprehensive count of what a meal consists of. Calorie Estimation: Calculates estimated calories for items according to their gram weight (deduced from user's input plus conversions), calories-per-gram data. Hence, provides a total meal calorie estimate. Conversational AI Assistant (CAMHEALTH): An OpenAI Assistant provides interactive feedback at key stages (e.g., reacting to the identified food, summarizing the calorie count) and engages in open Q & A about the meal, nutrition, or ingredients in the final step, restricted to relevant topics. Detailed Results: Presents a breakdown of each meal component, its estimated gram weight, and estimated calories.

6 The Stack

Programming Language - Python 3.x Web Framework - Streamlit Core Libraries: OpenAI: For using the OpenAI Assistants API. inference-sdk: Official Roboflow SDK to interact with the API. Pillow (PIL Fork): For image manipulation (opening, displaying). Standard Libraries: csv, os, io, json, tempfile, sys. External Services: Roboflow API (for image classification) OpenAI API (for Assistants API).

7 Data Managing

One crucial thing about this application is the processing of collecting nutritional Source data: Source Data Implicit in csv.py. The system originates from a simple Label-Calories-Unit format. Pre-processing (csv.py): A script that runs only once transforms this source data into a canonical Calories _ per_ Gram figure and, most importantly, estimates Small, Medium, and Large portion sizes in grams based on food category keywords (e.g., liquids, nuts, meats, baked goods). In this way, an output is produced-calorie _ table _ processed.csv. Runtime Data (calorie _ table _ processed.csv): This is the most important file that plays the role of the major source of truth at application runtime for calories/gram and S/M/L portion lookups. Conversion Dictionary (ut.py::CONVERSIONS): a great, hardcoded dictionary complements the csv data by providing the actual gram equivalents for a whole lot of nonstandard units (slice, piece, cup, tsp, etc...) across a range of foods, giving users even more possibilities to find their desired input. Fallback Logic: The system keeps fallback logics while searching for calorie data where exact label match -¿ base label match -¿ first word match is tried to maximize cover.

8 AI Integration

Roboflow: The predict_food_label_roboflow function is what would handle all the communication. It uses the InferenceHTTPClient from inference-sdk to take image bytes through a temporary file to a specified classification model endpoint, then parses the returned JSON. It checks for that predicted _ classes list and its corresponding confidence value within the predictions dictionary in the response structure typical of Roboflow classification models. It has a confidence threshold to reject unreliable predictions. OpenAI: The chatbot _ response function talks to the Assistants API. With Threads, it preserves a conversational context. Based on the stage parameter sent from the Streamlit app, the system gives different instructions to the Assistant (identified by ASSISTANT _ ID) about the chat to take-from initial good-morning kinds of greetings about the food to summing the total number of calories to a simple Q & A restricted to nutrition.

9 Challenges and Limitations

Identification Accuracy: Dependant on the performance of the Roboflow classification model, identifies a single core item, thus not multi-component meal identification. Data Dependency: The accuracy of the calories is directly based on how rich and robust the food data is the calorie _ table _ processed.csv and the CONVERSIONS dictionary. Portion sizes estimated in csv.py may lead to errors due to imprecise calculation methods. Scope: The system only provides calorie estimation with no macronutrient or micronutrient inputs. Unit Ambiguity: Flexible units like serving or piece are often intrinsically ambiguous without precise definitions. The system depends on the averages defined in the data/conversions. Maintenance: Updating the hard-coded CONVERSIONS dictionary and the procedure for generating calorie _ table _ processed.csv will be manual to add foods or correct inaccuracies.

10 Future Work

Improved Food Recognition: Explore possible other uses of object detection models (potentially also via Roboflow or other services) to detect several foods present on a single image. User Profiles: Save meals for users, keeping a history and enabling dietary goals. Refined UI/UX: Streamlining multi-step process by fast adding some most common extras or showing portion size in visuals.

11 Conclusion

CAMHEALTH is a revolutionary innovation in the provision of affordable, simple, and fun-to-use nutritional awareness tools. The project seeks to bring to fruition the promising novel solution in overcoming the time-consuming burden of manual calorie tracking. The user-centric workflow manifested by the application within the Streamlit interface succeeds in making what would otherwise have been a tedious chore into an interactive experience. Currently, the prototype utilizes curated datasets, and one-item classification; as a framework, it shows considerable promise and built-in scalability for future achievements. Dynamic unit conversions, backup data retrieval mechanisms and incremental stages of AI interactions are all demonstrated in the successful deployment.