Computer Vision Systems Programming: Project Report

Ondřej Peterka, 12229251

INTRODUCTION AND MOTIVATION

Calorie counting and nutrient tracking are crucial for sportspeople as they directly impact performance, recovery, and overall health. Additionally, tracking nutrients ensures athletes consume an adequate balance of carbohydrates, proteins, and fats, essential for muscle repair, energy production, and immune function. However, the process can be tedious, especially when dining out or eating pre-prepared meals where exact nutritional information is often unavailable. Calorie-counting apps can help with this task, yet they still rely heavily on user input and estimation, making it challenging to accurately measure intake. The necessity for improved nutritional labelling serves as an excellent use case for the latest advancements in OpenAI's models capable of processing vision inputs. The full code for this project is available on GitHub.

GOAL OF THE PROJECT

The goal of this project is to use the latest model of OpenAI with vision capabilities to serve as an automatic food detection system enhanced with an estimation of the calories and nutrients present in the food. This project consists of three high-level objectives

1. Obtain OpenAI API keys and find a suitable library/wrapper for communication with the API.
2. Use the GPT4-Vision model for processing food images, estimating their calories and nutritional values
3. Use standard GPT4 model for proposing alternatives/suggesting improvements to complete the user-specified calorie and nutrition targets.

Functional requirements

The system should fulfil the following list of functional requirements:

- provides capabilities to upload an image in a few standard formats (png, jpg)
- estimates calories and nutritional values based on the provided image
- displays estimated values coherently and has a graphical illustration of total values measured so far.
- on demand provides textual hints, where the user can make improvements to his eating habits and achieve goals.
- handles multiple users, their registration and login.
- keeps a history of entries for each user separately.
- estimates and suggests target values for all calories and nutrients based on user-specified data.
- handles invalid inputs (e.g. images with no food) and provides meaningful error messages.

Non-functional requiremets

The system should fulfil the following list of non-functional requirements:

- minimizes the latency wherever possible (login, food analysis, food alternative suggestion, storing into to database)
- is intuitive and easy to understand, provides a simple interface
- is scalable, allowing easy implementation of additional nutrients for tracking

Programming Language	UI Framework	Database	API communication
C#	WPF	MariaDB + Entity Framework	3rd party C# wrapper for OpenAI API communication

Table 1. Overview of all technologies used in the project.

Model	Input [1K Tokens]	Output [1K Tokens]
GPT-4	$0.03	$0.06
GPT-4 Vision	$0.01	$0.03
Image Input (1024x1024)	$0.00765	-
Image Generation (1024x1024)	-	$0.040
Text Embedding Large	$0.00013	-
Text-To-Speech HD	$0.030 [1K characters]	-
Speech-To-Text	$0.006 [1 minute]	-

Table 2. Pricing of different models provided by OpenAI. Models used in this project are in the upper part of the table.

IMPLEMENTATION DETAILS

This section describes details of the implementation, discusses and motivates the choices for technologies and provides insight into the system.

Technologies used

C# is an object-oriented programming language developed by Microsoft. It's known for its simplicity, type safety, and scalability, making it an ideal choice for building robust and efficient Windows applications. WPF (Windows Presentation Foundation) is a framework for building user interfaces in Windows applications. It provides a rich set of controls, layout options, and data-binding capabilities, allowing for the creation of highly interactive user experiences. Together, C# and WPF offer a powerful combination for building feature-rich and visually appealing Windows applications with ease. However, WPF, as the name suggests, is limited to only Windows machines.

Entity Framework simplifies database interactions by allowing developers to work with database entities as regular .NET objects, abstracting away the complexities of database operations.

OpenAI provides a Python library obtainable by pip. Developers can also communicate with the public OpenAI API by creating a JSON payload and directly sending it to the endpoint. As this project targets Windows machines and is developed in C#, a 3rd party NuGet package called OpenAI (OpenAI (2023a)) was used. This NuGet provides all communication capabilities and some higher-lever functions (uploading images, creating chat with history, etc.).

OpenAI API

Open AI provides a public API to enable end-users to integrate various models into their applications. Currently, OpenAI provides multiple models for Text Generation, Embedding (text encoding used in similarity comparison, clustering, anomaly detection, or recommendations), Text-To-Speech & Speech-To-Text, Image Generation, and Vision.

The API does not provide a free version. It is based on a pay-per-request model and is completely independent of the ChatGPT Plus. Each model has a different price point dependent on the number of tokens used. According to the OpenAI website (OpenAI (2023b)), 1000 tokens equal to approximately 750 words. All the API communications packages (the official by OpenAI and all the 3rd party ones) contain a functionality capable of computing the exact number of tokens in a given string. For some of the models, users have to pay for input and output tokens separately. It is also worth noting that even though new users get $5 for free credit to use, most of the advanced models will be unlocked after a minimum payment of $1 is made. Table 2 shows examples of different models and their costs. Models used in this project are at the top.

Basal Metabolic Rate (BMR) [kcal]	Male: $BMR = (10 \cdot weight) + (6.25 \cdot height) - (5 \cdot age) + 5$ Female: $BMR = (10 \cdot weight) + (6.25 \cdot height) - (5 \cdot age) - 161$
Calories [kcal]	$\{1.2, 1.375, 1.55, 1.725, 1.9\} \cdot BMR$
Proteins [g]	$\frac{Calories}{4} \cdot \frac{\{17.5,22.5,27.5\}}{100}$
Fats [g]	$\frac{Calories}{4} \cdot \frac{27.5}{100}$
Carbohydrates [g]	$\frac{Calories}{4} \cdot \frac{55}{100}$
Fibers [g]	Male: 38 Female: 25

Table 3. Overview of the equations used in the estimation of the target values for each user

Even though the pricing is very affordable and seemingly low, it is still worth thinking about this when considering the cost of development and runtime so that an appropriate business model is chosen. For instance, the development of this project (including the generation of the logo for the app) cost $0.74.

Equations for estimated values

NutriVision tracker estimated the correct amount of calories and all nutrients automatically. When the user is creating an account, he/she will be asked to provide the following information:

- Name, Password, Email (not used to estimate values)
- Age
- Height [kg]
- Height [cm]
- Gender (male/female)
- Activity Level (5 different options from Sedentary to Extremely Active)

From the following information, estimated target values are computed using the Harris-Benedict equation (Mifflin et al. (1990)) as follows:

In Table 3, equations using sets in the computations are introduced. Numbers from the set correspond to one/or more of the activity level categories. In the Calorie estimation, each number in the set corresponds to exactly one activity level, and in the Protein estimation, the first set number corresponds to the least active level and the last set number corresponds to the most active level.

EVALUATION

Accuracy of Estimation

This project aimed to estimate calories and nutrients in food pictures. As a part of the evaluation process, the nutrients of a few selected images were estimated and compared by hand-calculated values from the app Calorie Tables. It's worth noting that for the person-aided evaluation of the food images, estimating the weight of food correctly greatly affects the results. However, in many scenarios, this task might be very difficult and as a result,t the human-estimated values might not reflect the exact reality. Three selected images are depicted in Figure 4 and their corresponding human- and computer-computed values are shown in Tables 5, 6, and 7.

Looking and Test Images 1 and 2 we can observe that OpenAI's model has problems with misclassification. In the first case, potato dumplings were classified as an uncommon banana type and in the second case, fried cheese was estimated to be a fried fish. This behaviour is not unexpected, the NLP model has no previous experience with this food, whereas the person knows that potato dumplings are the usual side for spinach and pork chops or that a square-shaped fried object will most likely be fried cheese. In this case, the model underestimates the majority of the values. This might also be caused by the incorrect weight estimation.

The third test image was specifically chosen in a way that there's no ambiguity in the contents. Here we observe correct food detection and estimated nutrition values are closer to the human estimation.

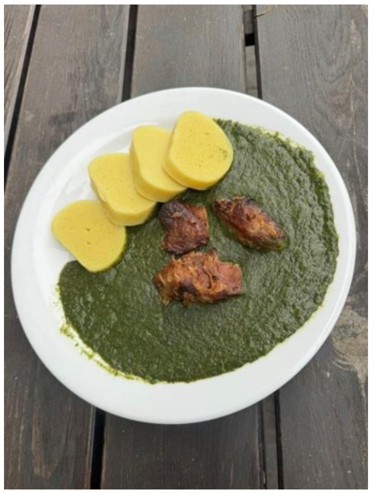
Image 1

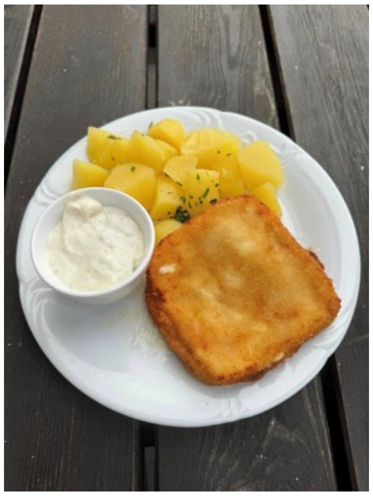
Image 2

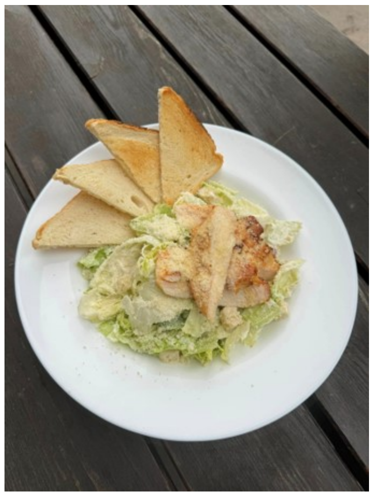
Image 3

Table 4. Selected images for evaluation.

Test Image 1	Human Estimation	NutriVision Tracker
Food	Spinach sauce with pork chops and **potato dumplings**	Spinach sauce, **Fried Plantains**, Grilled Meat
Calories [kcal]	800	600
Proteins [g]	20	45
Carbohydrates [g]	37	30
Fats [g]	54	35
Fibers [g]	5	5

Table 5. Comparison of Human-computed and NutriVision Tracker estimated calories and nutrients for Test Image 1.

Test Image 2	Human Estimation	NutriVision Tracker
Food	Fried **cheese** with potatoes and mayonnaise	Breaded fried **fish** with boiled potatoes and tartar sauce
Calories [kcal]	1100	600
Proteins [g]	38	25
Carbohydrates [g]	66	60
Fats [g]	80	30
Fibers [g]	4	5

Table 6. Comparison of Human-computed and NutriVision Tracker estimated calories and nutrients for Test Image 2.

Test Image 3	Human Estimation	NutriVision Tracker
Food	Caesar Salad with Roasted chicken, parmesan and toast	Caesar Salad with CChickenand Toasted Bread
Calories [kcal]	700	560
Proteins [g]	42	36
Carbohydrates [g]	43	28
Fats [g]	37	42
Fibers [g]	4	5

Table 7. Comparison of Human-computed and NutriVision Tracker estimated calories and nutrients for Test Image 3.

Image	Suggestion
Test Image 1	• Replace fried plantain with boiled plantain to reduce fat content. • Use lean cuts for grilled meat to lower fat and increase protein proportion. • Add a variety of other leafy greens to increase fibre without significantly raising calorie count.
Test Image 2	• Swap breaded fish for grilled fish to decrease fats. • Replace boiled potatoes with a mixed vegetable side to lower carbs and increase fibre. • Use a yoghurt-based sauce instead of tartar to cut down on fats.
Test Image 3	• Include more vegetables like cucumbers and tomatoes to boost fibre while keeping calories in check. • Use whole-grain bread to increase fibre and nutrients. • Opt for a lighter dressing or use lemon juice and spices instead of Caesar dressing to reduce fat.

Table 8. Suggested alternatives/Improvements for the three Test Images.

As mentioned with Test Images 1 and 2, the model has problems distinguishing "hidden" contents of the food (e.g. fried food) and does not take into context how the individual food components complement each other (it's not very usual to eat fried bananas with spinach and pork). One idea how to improve this inaccuracy was to implement feedback for the model, where the model would be able to ask additional questions when unsure about the contents of the image. However, the model was generally overly confident and was certain in all tested images, even though the produced prediction was incorrect every time. This is currently one of the major weaknesses of the model — the inability to estimate uncertainty and ask for additional follow-up questions.

Improvement Suggestions

This section concerns the second task of this project — to let the user request healthier alternatives and improvements to their food to better reach their nutrition goals. For this evaluation, the same three Test Images as in the Accuracy Evaluation were used.

From Table 8 we can see the system making meaningful suggestions and improvements and reasons for them, given the model was able to successfully predict the contents of the image. The inputs to the suggestion model are only textual descriptions of the foods, their estimated nutrients and calories and the goals of the user. It is therefore not possible for the model to know that instead of, e.g., plantains in Test Image 1 it should be working with potato dumplings.

Speed

Systems with quick reaction time are nowadays a *de-facto* standard. Large delays will annoy users and might prevent the app from gaining more popularity. NutriVision tracker has three points where the user has to wait for some action: after logging in to load the data, when adding new food, and when requesting alternatives for better food choices.

In this project, all of those actions are implemented as async methods and, therefore, allow the user to manipulate the remainder of the app while waiting for the request to finish.

Delays of the API requests (adding new food and requesting alternatives) in this case are something one can not affect in any way. The first request after the startup of the app takes approximately 5-10 seconds. This corresponds to library initialization. Each successive request takes approximately 2-5 seconds.

CONCLUSION AND FUTURE WORK

A fully functional standalone system was developed. This system provides a multi-user environment, utilizes a database to save the data, and has an easy-to-use UI with graphic and interactive elements for the user to orientate better in his goals and the current progress.

OpenAI API was utilized with partial success. GPT-4-Vision was used to process images and estimate nutritional values and calories. GPT-4 was used to obtain sensible recommendations for food alternatives/upgrades. The DALL-E 3 model was used to generate the logo of the application. When compared to hand-calculated and estimated values, model-estimated values were off by approximately 20-30%. The biggest weakness of the tested models was their overly certain detections of the foods in images and their inability to estimate their uncertainty and ask for more specifications from the user.

The API correctly suggests improvements and reasons for the choice. However, the correct suggestion is conditioned by the correct detection of the food. As this functionality is purely text-based, it will provide suggestions based on the (possibly) incorrectly detected images' descriptions.

The major weakness of this project is the inability of the model to detect ambiguous foods, and tell, when the estimation is not certain (and potentially ask the user for additional information). This will be the main subject of the future development.

REFERENCES

Mifflin, M., St Jeor, S., Hill, L., Scott, B., Daugherty, S., and Koh, Y. (1990). A new predictive equation for resting energy expenditure in healthy individuals. *The American Journal of Clinical Nutrition*, 51(2):241–247.

OpenAI (2023a). OpenAI NuGet Package. `https://www.nuget.org/packages/OpenAI/`.

OpenAI (2023b). OpenAI Pricing. `https://openai.com/pricing`.