Keyur Patel

Nutrition Calculator System Report

Section 1: Problem Background and Motivation

Critical to the maintenance of a healthy lifestyle is the diet one consumes. In fact, it can be said that it is the most important aspect of a person's well-being. The consumption of the proper amounts of various macronutrients, such as cholesterol, fats, carbohydrates, and proteins is imperative to ensuring good health. Without any sort of assistance, tracking these quantities manually each day can be an arduous task. A tool that has been a great help in this endeavor is the "Nutrition Facts" label, a piece of information located on nearly every container of commercially sold food that contains the amount of nutrients contained in a single serving size of the food itself. This project aims to ease the process of manually tracking calorie and nutrient consumption by taking advantage of the mostly universal format of these labels. It takes as input images of nutrition labels alongside amounts of serving sizes consumed, which the system then processes via two image processing libraries to extract the final numbers of each nutrient consumed. Finally, the system outputs to the user a summary of the user's nutrition for the day, consisting of a textual summary as well as a set of visualizations represented as graphs showing the proportion of the recommended daily value consumed, and some suggestions for changes the user may make to properly meet their nutrition goals.

For the development of this system, some assumptions about user need and behavior need to be made. To begin, the system will be built to focus on a specific subset of all possible information from nutrition labels, namely these ten quantities: calories, total fat, saturated fat, trans fat, cholesterol, sodium, total carbohydrates, dietary fiber, sugars, and protein. These quantities appear the most across labels on all types of food, and are generally considered the most important macronutrients for the body. Other quantities, such as Vitamin A or calcium, are important, but do not appear on all labels, and if they were to be tracked, the system would report a lot of values as missing due to their absence. Hence, as per the assumption that the aforementioned ten are the highest interest quantities, only they will be tracked. Another assumption made is that the images input to the system will be of containers that are either in the shape of a box or a slightly curved/rounded object, such as a bottle. These surfaces provide the most consistency in terms of the label surface texture, which ideally would be smooth for the best analysis possible. Less rigid containers, such as plastic bags, will be included in the testing image library, but are primarily examined for curiosity's sake, and will not have nearly as much sway in the development and fine-tuning of the system as boxes or rounded objects. A third assumption to be made is that the user is interested in their consumption compared to the standards set for daily nutrition by the US FDA, which are the same standards food manufacturers use to create nutrition labels. As such, the system will represent consumption of each macronutrient as a graph displaying the portion of the daily value consumed, most likely as a series of pie charts.

Section 2: Domain Engineering and Capture of Data

Domain engineering for the purposes of this project was straightforward: all images were captured in well-lit spaces, with the camera facing the nutrition label head on, and with the label being large and centered within the frame. The images were all captured via an iPhone 12 camera, and stored as .jpg files within a directory labeled 'Testing'. Within this directory, the images were further categorized in subdirectories based on shape, with the three shapes in consideration being boxes (such as cereal boxes or even roughly cube-shaped jars), rounded (such as round jars, bottles), and bags (such as candy bags). These three shape categorizations provide a good spread of how the majority of items that can be found at a grocery store are shaped, and the system's performance on all three were compared.

To get a sense of the system's performance for users other than the system's designer, other individuals were asked to capture their own images to add to the testing set. These individuals were given an explanation of the domain engineering, and followed as best as they could.

In total, the library consisted of 42 images, 24 images of boxes, 9 images of bags, and 10 images of rounded containers. These images are located in Appendix A of this report.

Section 3: System Approach

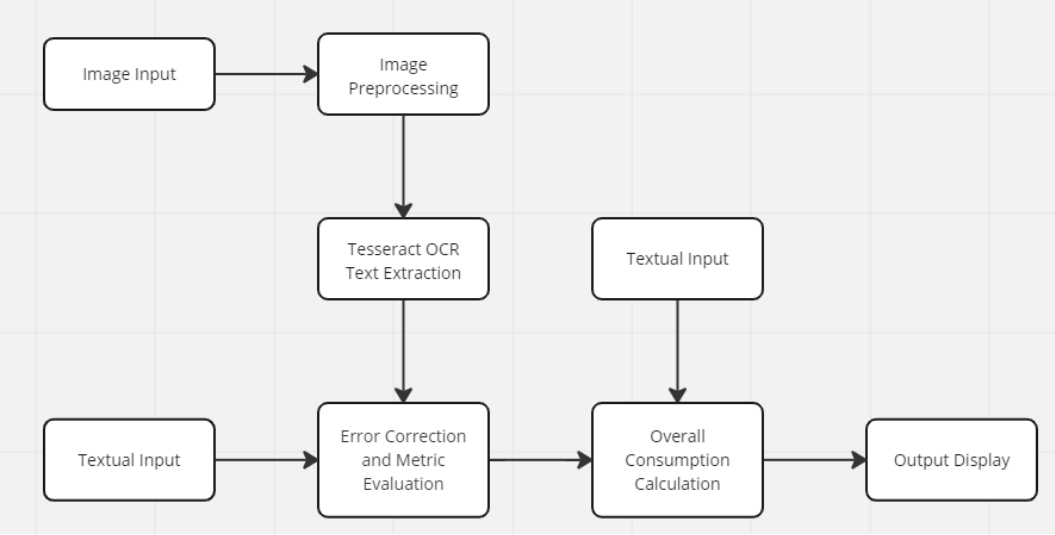


Figure 1: System Flowchart

The general flow of information through the system is detailed in the flowchart above. Input from the user is required in three areas, initial image inputs, text input for error correction, where the user is prompted to rectify any mistakes from Tesseract OCR, and text input for overall consumption calculations, where the user is prompted to enter how much of each quantity was consumed. Image inputs are taken in from a specified directory within the project folder, so all the user has to do is upload the images there and the system will analyze them all. Some assumptions about the system internals must be made as well. For instance, it is assumed that the

images will require some form of preprocessing prior to being submitted to Tesseract. Though the domain definition as described in the previous section provides a good outline for image taking, some sort of preprocessing to filter out unnecessary background information while placing the focus on the nutrition label will most likely be needed. As such, a proposal for the implementation of a threshold function when preprocessing images will be tested, with the hopes that it will improve system performance. Additionally, some tweaking of Tesseract may be necessary, whether it be a modification to parameters passed to the tool or a modification to the dictionary of words it is able to recognize. The error correction and metric evaluation stages are included as it is assumed that Tesseract will inevitably make mistakes when reading data, and the only proper means for correcting these errors is manual checking and rectification of incorrect values done by the user. Labels are stored as label objects, which have been designed to hold variables corresponding to each nutrient being tracked, initially populated with 'N/A' as the value for all.

Section 4: Evaluation Metric

The primary evaluation metric used for the scoring of the system is a manual check for accuracy of the values read in from a given image. As OCR is not a perfect system, there are bound to be errors, and the machine is typically unable to effectively detect whether the incorrect value has been read in or not. Hence, this evaluation metric is heavily dependent on the user of the system to provide input. In total, there are only ten nutrient values the system keeps track of, being calories, total fat, saturated fat, trans fat, cholesterol, sodium, total carbohydrates, dietary fiber, sugar, and protein. Therefore, the evaluation metric is scored out of 10, with a 1 being added to the score for every correct entry and a 0 for every incorrect entry. Though this requires a great deal of human input and thus takes a long time to determine scores, this was the only real means of providing a means of evaluating the performance of the machine. This provides a basic means of determining system accuracy that will assist in further experimentation and system building.

$$Score \ = \ (\#\ of\ correct\ entries)\ /\ 10$$

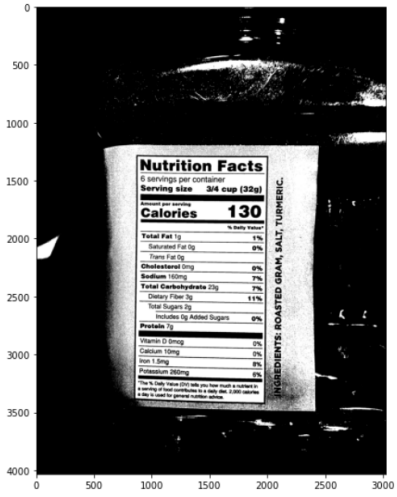
Section 5: Experimentation - Image Preprocessing

Image preprocessing was carried out primarily via openCV methods. Images are read in and stored via the cv2.imread() function, and passed through a threshold function to convert them into their binary representations, a suggestion taken from a previous paper covering the digitization of bank records. A threshold of 170 was determined to be an adequate value for the purposes of this system through experimentation with values in the 150 - 220 range. This range was chosen as the ideal output of the threshold function would single out the nutrition label as the largest object in the image, and due to the white background of the label, it would be resilient to a high threshold value. The addition of the threshold function was a decision made following a

great deal of testing and comparison of the evaluation metric of the system when handling colored versus binary inputs.

Figure 2: Comparison of OCR Output for Original vs Binary Image


```
Original Image:
----------------  ---
Name
Calories
Total Fat (g)      1
Saturated Fat (g)  0
Trans Fat (g)      N/A
Cholesterol (mg)   0
Sodium (mg)        160
Total Carbs (g)    23
Dietary Fiber (g)  3
Sugars (g)         2
Protein (g)        7
----------------  ---
```




```
Binary Image (threshold=170):
----------------  ---
Name
Calories
Total Fat (g)      1
Saturated Fat (g)  0
Trans Fat (g)      0
Cholesterol (mg)   0
Sodium (mg)        160
Total Carbs (g)    23
Dietary Fiber (g)  3
Sugars (g)         2
Protein (g)        7
----------------  ---
```

One example of testing is shown to the left - prior to binarization, the system was unable to successfully read the values for calories and trans fat, leaving the system with an accuracy of 80%. After thresholding, the system was able to accurately read the value for trans fat, showing an increase of 10% in the evaluation metric to 90%. Tests similar to these were run on a large subset of the entire image library, with the results shown in the next section. These tests were run with default parameters for Tesseract (explained in next section). An interesting observation from these experiments is that thresholding produced an increase in accuracy for a majority of the boxes and rounded objects, but severely ruined the performance of the system for bags. Upon analysis of the effect of thresholding on bags, it can be seen that nearly the entirety of the nutrition label is unrecognizable in the binary images. This is most likely due to the inconsistent texture of a bag, which could contain many wrinkles and imperfections that distort the lighting across the surface. As such, the system will suffer from performance on bags, as discussed in the limitations section.

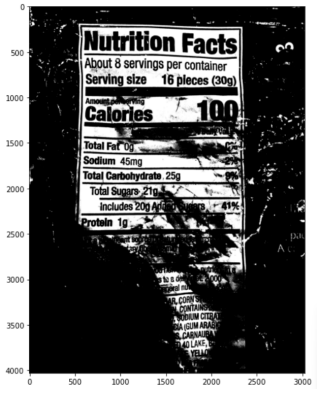


Figure 3: Comparison of Original and Binary Image for a bag

Section 6: Experimentation - OCR Page Segmentation Method

Tesseract OCR expects a full page of text by default when accepting input images. As such, it is not properly equipped to effectively extract text from less uniform images when run without any configuration set up. To handle varying inputs, Tesseract contains an option to specify a page segmentation mode, or pms.

The best page segmentation mode was straightforward and simple to determine and implement - page segmentation mode refers to the means by which Tesseract locates and segments the input image to read text. Therefore, which mode can be considered the best is dependent upon the image library, namely the layout of the text of interest. As all nutrition labels follow a standard format, and tend to be aligned vertically in a column, the page segmentation mode of "assume a single column of text of variable sizes" (noted as –psm=4) could be optimal for this system. This mode works best for receipt processing according to Tesseract documentation, and as nutrition labels follow a similar format, it has the potential to work well for them as well. The performance of the system with this change in psm was evaluated for both colored and binary images, much like in the previous section, with the results displayed below:

For more information on Tesseract page segmentation modes and a list of available ones, see: https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html#page-segmentation-method

Item Category	Item Name	Default psm		psm=4	
		Original Image	Binary Image	Original Image	Binary Image
Boxes	Cumin	0.9	0.9	0.8	1
	Peanuts	0.8	0.9	0.6	0.9
	Soup	0.8	0.8	0.5	0.7
	Cheez-Its	0.9	0.8	0.4	0.7
	Chocolate	0.1	0.3	0.5	0.4
	Goldfish	0.9	0.8	0.7	0.8
	Frappuccino	0.8	0.8	0.9	0.9
	Cheesecake	0.4	0.5	0.6	0.6
	Reeses Puffs	0.8	0.7	0.9	0.8
	Cake Mix	0.9	0.9	0.8	1
	Average	0.73	0.74	0.67	0.78
Rounded	Oats	0.8	0.7	0.8	0.8
	BBQ Sauce	0.4	0.5	0.3	0.7
	Peanut Butter	0	0.5	0.2	0.6
	Syrup	0.7	0.7	0.7	0.6
	Coconut Water	0.3	0.5	0.5	0.6
	Average	0.44	0.58	0.5	0.66
Bag	M&Ms	0.8	0	0.8	0
	Nerds	0.9	0	0.9	0
	Walnuts	0.7	0	0.6	0
	Doritos	0.4	0.7	0.3	0.7
	Gummy Worms	0.5	0.8	0.4	0.6
	Average	0.66	0.3	0.6	0.26
Weighted Average		0.618	0.618	0.6	0.66

Figure 4: Experimentation Output for Image Preprocessing

The numerical values reported in each cell represent the evaluation metric as determined manually from reading system outputs. The need for manual input for each iteration of testing made it a time-consuming task, something which will be discussed in the limitations section. The overall performance metric in the last row, weighted average, was chosen to have a weighting as this system was built with the intention for users to submit primarily images of boxes and slightly curved objects, with bags being the lowest priority but still included for curiosity's sake. The weighting used was

$0.50 * Boxes + 0.35 * Rounded + 0.15 * Bags$, reflecting the order of importance of the three categories with which this system was designed.

From the results recorded in the table, it would appear that preprocessing the images into binary representations (with threshold=170) and passing them to Tesseract with the flag –psm=4 leads to the best average scores across the tested subsets of Boxes and Rounded, whereas performance suffered greatly when preprocessing Bags into binary images, due to the reasons mentioned in the previous section. Based on the outcome of experimentation, both preprocessing of images into their binary representations and utilizing the fourth page segmentation mode will be used by the final system.

Section 7: Custom Tesseract Dictionary

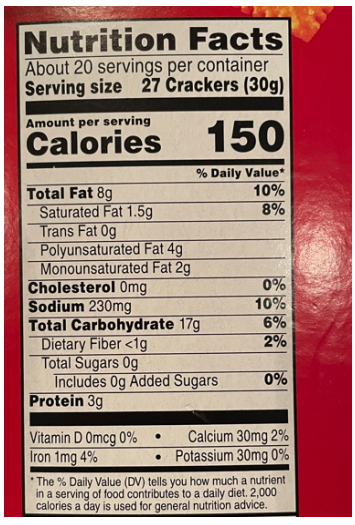
The implementation of a custom dictionary for Tesseract to recognize words from when parsing images was initially considered during the development of this project, in hopes of increasing the accuracy of Tesseract readings. Upon further research and some experimentation, however, it was discovered that this approach did not change the results of OCR conducted on the images. Tesseract works by reading text and attempting to match character sequences to those found within an internal dictionary in hopes of returning an actual word. The main purpose in disabling the Tesseract dictionary or adding custom words would be to help the system recognize non-dictionary words, such as names for instance. As all of the words that are being searched for in this project would reasonably exist within the dictionary, and the system has been observed to have an accuracy of approximately 70% when reading in values with the base dictionary, it was decided to no longer further pursue this change, as the potential benefits appear to be practically nonexistent. If Tesseract were to return misspellings of nutrient names, such as “Oaturated Fat” instead of “Saturated Fat”, this could potentially be handled with another process, detailed in the limitations section.

Section 8: OCR Output Parsing and Fixing

Figure 5: Example of Unfixed OCR Output

As Tesseract simply takes an image as input and returns a string of what was read, postprocessing of the returned string is necessary to extract the actual values of each nutrient. This parsing was fairly straightforward, and consisted of a linear scan of the string to locate the names of the desired fields. Once a desired field was

```
----------------  ----
Name
Calories          a
Total Fat (g)     89
Saturated Fat (g) 1.59
Trans Fat (g)     0g
Cholesterol (mg)  Omg
Sodium (mg)       N/A
Total Carbs (g)   1/9
Dietary Fiber (g) N/A
Sugars (g)        0g
Protein (g)       3g
----------------  ----
```



located, such as “cholesterol”, the value immediately following it in the string was taken as its corresponding value. Before this could be taken as a final value, however, some more processing of the value was required, as the outputs from Tesseract quite often contained errors. It tends to read in certain alphabetic characters as numbers and vice versa. For instance, as can be seen on the previous page, the system read in “8g” as “89”, mistaking the ‘g’ as a ‘9’, and “0g” as “Og”, mistaking the ‘0’ as an “O”. Tesseract is prone to making these errors as it attempts to read in whole words based on an internal dictionary rather than individual characters. As such, it seems to have a tendency to change characters to maintain a uniform word constitution (either all alphabetic or all numeric). To rectify this, simple substitutions were used to replace instances of the letter O with 0 and instances of the number 9 appearing at the end of a number with g. After this, non-numerical values were filtered out via a regex, which specified that all characters not belonging to the set of digits or ‘.’ (regex: [^0-9.]) be removed from the values, leaving only numbers. This solution was adapted from a paper discussing error correction when reading information with Tesseract. This leads to an easier postprocessing and calculation of daily macro values. The system was built with the assumption that values left with an ‘N/A’ after Tesseract output parsing can be set to 0, as the omission of a certain nutrient from a label can be an indicator that the item contains an insignificant amount of the nutrient. Additionally, 0 can be considered a ‘safe’ filler value as it is a value that tends to appear quite often across all nutrition labels, and replacing all unknown values with a 0 cannot decrease system accuracy, but has the potential to increase it for any given image. The above label after preprocessing can be seen to the left.

Figure 6: Fixed OCR Output

```
-----------------  ---
Name
Calories           0
Total Fat (g)      8
Saturated Fat (g)  1.5
Trans Fat (g)      0
Cholesterol (mg)   0
Sodium (mg)        0
Total Carbs (g)    1
Dietary Fiber (g)  0
Sugars (g)         0
Protein (g)        3
-----------------  ---
```

Section 9: Output Display

After OCR output parsing and correction, the user is prompted to manually correct errors with each label should they exist, with the previously described evaluation metric for the system being calculated once this error correction is finished. Following error correction, the user is asked to input how much of each item was consumed throughout the day. This input was initially planned to be in a universal metric, such as grams, and calculated using the serving size located on the nutrition labels. In practice, however, it was found

```
Name              hot_pocket
Calories          0
Total Fat (g)     26
Saturated Fat (g) 0
Trans Fat (g)     0
Cholesterol (mg)  120
Sodium (mg)       610
Total Carbs (g)   29
Dietary Fiber (g) 2
Sugars (g)        5
Protein (g)       13
----------------  ----------
Is there an error in the reading of hot_pocket (Y/N)? y
Please enter correct values for each incorrect nutrient.
Calories:         400
Total Fat (g):
Saturated Fat (g): 10
Trans Fat (g):
Cholesterol (mg):
Sodium (mg):
Total Carbs (g):
Dietary Fiber (g):
Sugars (g):
Protein (g):
2 inconsistencies noted.
System accuracy: 80.0%
```

Figure 7: Example of Error Correction

that Tesseract had a difficult time properly reading in serving sizes, often due to the space between “Serving Size” and the actual associated quantity found on labels. As such, users report quantities consumed in terms of the number of serving sizes consumed. A similar issue with reading calorie counts was made obvious as well. Both of these will be discussed further in the limitations section. The final macronutrient calculations are carried out and output is displayed to the user in both textual and graphical formats - the textual format displays nutrients and the amount consumed alongside the percentage of daily value consumed as compared to amounts set by the FDA, while the graphical format displays pie charts showing the amount of each nutrient consumed in terms of percentage of daily value. When determining the %DV, two quantities are left ignored, namely calories and sugars. Calorie quantity has no real set recommended value across all individuals, as everyone’s calorie intake requirements are unique, while sugar in general has not been assigned a standard by the FDA.

```
Nutrition Summary:
Nutrient             Amount    % Daily Value
-----------------  --------  ---------------
Calories:            1250
Total Fat (g):         52.5               67
Saturated Fat (g):     20.5              102
Trans Fat (g):          0                 0
Cholesterol (mg):     150                50
Sodium (mg):         1530                67
Total Carbs (g):      167                61
Dietary Fiber (g):     11                39
Sugars (g):            67
Protein (g):           30                60
```

```
Items Eaten Today:
bbq - consumed 1 serving size(s)
bread - consumed 1 serving size(s)
chocolate - consumed 1 serving size(s)
doritos - consumed 1 serving size(s)
hot_pockets - consumed 1 serving size(s)
ice_cream - consumed 1 serving size(s)
jelly - consumed 1 serving size(s)
pasta - consumed 1 serving size(s)
vegetable_stock - consumed 1 serving size(s)
```

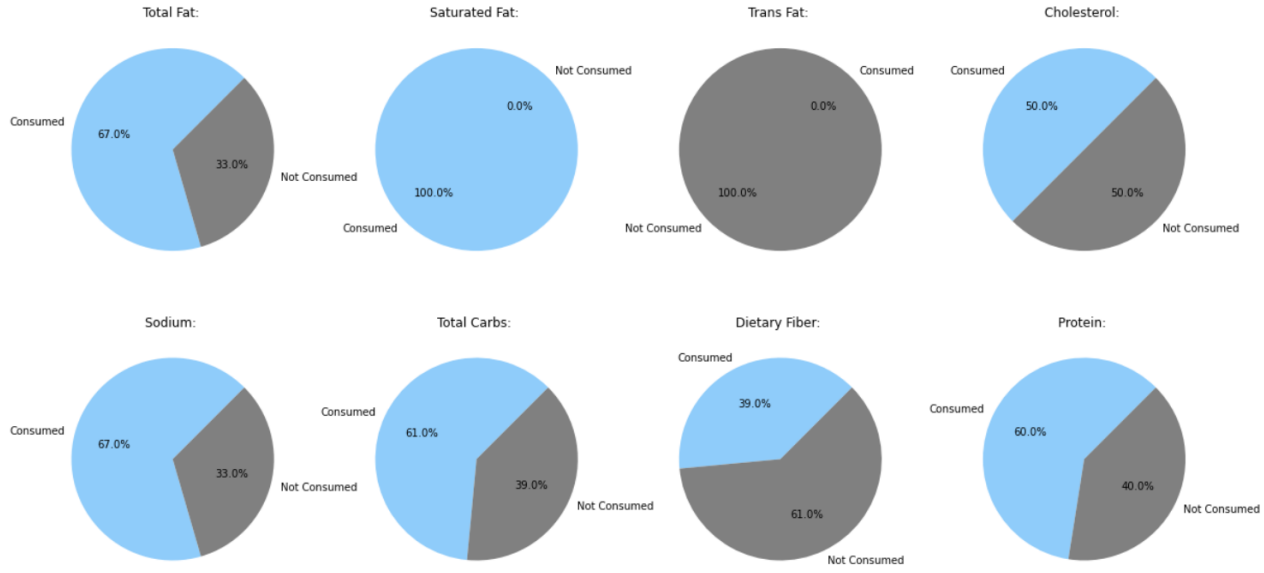


Figure 8: Example of Nutrition Summary Output

Section 10: Suggestions for User

The output section concerning suggestions to the user based on their current consumption is intended to provide ideas for the user to meet daily nutrition goals either by increasing or decreasing the intake of certain nutrients. It is currently a very basic system, which mentions to

the user which nutrients should be increased and which should be decreased in the daily diet to meet the FDA standards. For some nutrients, the daily value standard acts as a limit rather than a minimum recommendation, such as sodium and trans fat. The advice for these values have been adjusted appropriately, with a small range of values below the standard being considered as "good". In addition to mentioning the specific nutrients for which consumption should be altered, the system also advises the user on certain foods that would aid in making such a change, such as a suggestion to eat more lentils to increase fiber intake. Suggestions are reported as natural language sentences, which currently follow a basic fill-in-the-blanks template. The system has potential for future expansion into one that delivers more comprehensive advice to the user, perhaps through the addition of a database storing foods and what nutrients they are good for providing in a diet.

```
Your consumption of Total Fat is low, consider eating more daily.
Healthier foods to help you meet your total fat goal include avocados and eggs.
Ideal range for Total Fat is between 80% and 110% of the standard daily value.

Your consumption of Saturated Fat is high, consider eating less daily.
Consider reducing your intake of foods such as red meat and ice cream to lower saturated fat consumption.
Ideal range for Saturated Fat is less than 100% of the standard daily value.

Your consumption of Trans Fat is perfect, you are not exceeding the recommended limit!
Ideal range for Trans Fat is less than 100% of the standard daily value.

Your consumption of Cholesterol is low, consider eating more daily.
Healthier foods to help you meet your cholesterol goal include eggs and cheese.
Ideal range for Cholesterol is between 80% and 110% of the standard daily value.

Your consumption of Sodium is perfect, you are not exceeding the recommended limit!
Ideal range for Sodium is between 50% and 100% of the standard daily value.

Your consumption of Total Carbs is low, consider eating more daily.
Healthier foods to help you meet your cholesterol goal include oats and fruits.
Ideal range for Total Carbs is between 80% and 110% of the standard daily value.

Your consumption of Dietary Fiber is low, consider eating more daily.
Healthier foods to help you meet your protein goal include fruits and lentils.
Ideal range for Dietary Fiber is between 80% and 110% of the standard daily value.

Your consumption of Protein is low, consider eating more daily.
Healthier foods to help you meet your protein goal include eggs and white meat.
Ideal range for Protein is between 80% and 110% of the standard daily value.
```

Figure 9: Example of Suggestions for Nutrition Summary in Previous Section

Section 11: Observations About the Final System

The final system was tested across 6 groups of 9 labels chosen randomly from the image library - to keep in line with the assumption that the user would submit mostly images of boxes, 5 of these images were drawn from the set of all boxes, 2 from the set of rounded objects, and 2 from the set of bagged objects. While a majority of images were captured by the system developer, some were captured by third parties to mimic testing by other users. For each final test, the metric was evaluated after the execution of the entire system, including image preprocessing, Tesseract text extraction, and Tesseract output postprocessing. The metric for each individual label in the trial was then averaged twice, once normally and a second time with the weighted average scheme, for the final scores for the trial. The results are displayed below:

Trial	Box 1	Box 2	Box 3	Box 4	Box 5	Rounded 1	Rounded 2	Bag 1	Bag 2	Average	Weighted Average
1	1	0.6	0.8	0.9	0.6	0.7	0.9	0.2	0.3	0.67	0.7075
2	0.6	0.5	0.7	0.6	0.7	0.6	0.6	0.4	0.9	0.62	0.6175
3	0.7	0.9	0.8	0.7	0.6	0.7	0.9	0.8	0.3	0.71	0.7325
4	0.9	1	0.8	0.8	0.3	0.9	0.8	0.1	0.3	0.66	0.7075
5	0.6	0.2	0.9	0.7	0.6	0.9	0.9	0.4	0.5	0.63	0.6825
6	0.9	0.8	0.6	0.8	0.4	0.9	0.9	0.7	0.6	0.73	0.7625
									Final Averages:	0.67	0.7016666667

Figure 9: Final Testing Results

In total, across the 6 trials, the system was tested on 42 unique images, the entirety of the testing library. It can be seen from the data above that the system performance averaged around 70%, for both the unweighted and weighted averages. Some notable observations from the final testing are noted here below:

The system consistently worse on bags than the other two categories, as predicted. Upon analysis of the images of bags at various stages of processing, the reason is clear - bags do not have a rigid structure, and as such can exhibit varying textures and uneven lighting across their surfaces. These issues can greatly harm the effectiveness of the threshold function, thereby making OCR nearly impossible on the preprocessed image. Additionally, bags tend to be made of a shiny plastic that reflects light much more than cardboard boxes or paper labels on bottles.

Some boxes passed to the system performed considerably worse than others, with images for popcorn, milk cartons, granola bars, and chocolate exhibiting a poor system performance. Upon further analysis, this seems to stem from issues within the images themselves rather than the system. For instance, the popcorn box and milk carton were both captured in lighting that was poorer than that of the rest of the images. The image for chocolate was taken at an angle, and not head on, producing poor performance (see appendix for images). These issues were most likely due to the photographer not clearly understanding the domain engineering required for the images, and exhibit a limitation of the system that could cause problems for other users. For granola bars, the nutrition label followed a slightly different format than the rest of the images, which would have produced errors when passing the image through Tesseract.

Section 12: Limitations

Described below are some limitations the final system suffers from, as well as a recounting of some related assumptions.

Easily the biggest limitation of the system is the need to restrict inputs to images of boxes and rounded objects. In its current state, the system is unable to effectively binarize images of bags in a way which retains all of the important information while also filtering out the background. As mentioned before, the texture and material of many bags make them more susceptible to reflecting light in an uneven manner across the surface, messing with the threshold function and effectively ruining many images of bags after binarization. A potential future extension to the system could be to develop a means of automatically detecting the surface composition of a label via computer vision functions, and determining whether or not to threshold the image based on that, allowing for bags to pass through the system without thresholding.

A decent limitation of the system is the speed with which testing can be carried out. With a task like OCR, there is no real means of automating error checking that doesn't require a decent amount of manual labor on the part of the user. One potential solution would be to manually upload correct values and to have the system check what was read from an image

against these values, but that would defeat the purpose of the system as it is supposed to determine those values from the images alone.

A limitation involving the processing of Tesseract outputs is handling misspellings of desired words, such as the system reading “Saturated” as “Oaturated”. As such, it cannot accurately read in information for a label where the words may be partially scratched out or covered up by bright lighting. A potential solution to this limitation could be to consider the edit distance between what was read and the actual nutrient names, and accepting words that are within a threshold edit distance.

The formatting of a label itself can prove to be a source of error for the system as well. One common issue is the reading of calorie counts, which the system is unable to do accurately in nearly all cases. This is most likely due to the whitespace between the word “Calories” and the actual amount present on most labels, making it difficult for Tesseract to interpret these as consecutive terms. Another limitation comes from the overall format of a given label. In some cases, the labels have the english names for the nutrients immediately followed by the spanish names (“Total Fat/Grasa Total” for instance). In such cases, the system struggles to report correct values.

Section 13: Reflection

Overall, the development of this project has been a frustrating but interesting experience. I definitely have enjoyed being able to develop a system of my choosing with more emphasis put on the process rather than the numerical results, as it provided me a great opportunity to really experiment and learn more unique technologies I’ve never seen before. Going through Tesseract documentation and attempting to learn the workings of the system as well as how it can be modified and configured differently to better approach the problem I presented taught me a lot about how annoying it can be to try to learn about things for which only limited documentation exists, but I feel that this is a skill that will go far in a career as a developer. Additionally, image processing techniques not only learned through this project but the entire course have been a great asset for someone interested in studying computer vision in the future, as I currently am. In the end, I was not able to fully develop the system I initially envisioned, and had to make various changes to my approaches, but I am satisfied with the work I have completed, and feel it is a great groundwork for future expansions and building out into a full-stack application with a dedicated frontend and backend.

If I were to redo this project from scratch, there are a few things I would do differently throughout the entire process. For instance, I would definitely put more emphasis on and give myself more time for testing, as the manual testing that needs to be carried out has been very time-consuming. Additionally, there were some parts to the system I only thought of after it was too late to implement, as time and schedule constraints prohibited me from looking fully into them, such as an implementation of some sort of edit distance mechanism to determine how different two words are.

References:

- https://arxiv.org/pdf/1811.06193.pdf
- https://arxiv.org/pdf/2204.00052.pdf
- https://arxiv.org/pdf/2101.05214.pdf
- https://arxiv.org/ftp/arxiv/papers/2107/2107.03700.pdf
- https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html#page-segmentation-method
- https://www.fda.gov/food/new-nutrition-facts-label/daily-value-new-nutrition-and-supplement-facts-labels

Appendix A: Image Library

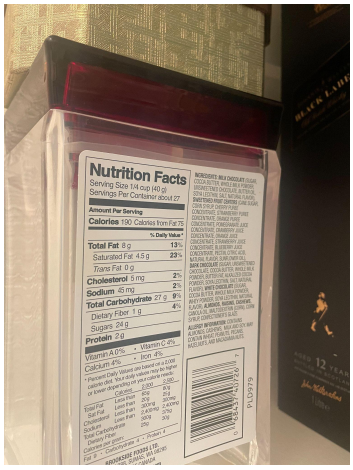
Boxes:

cake_mix.jpg

cheesecake.jpg

cheezits.jpg

chocolate.jpg



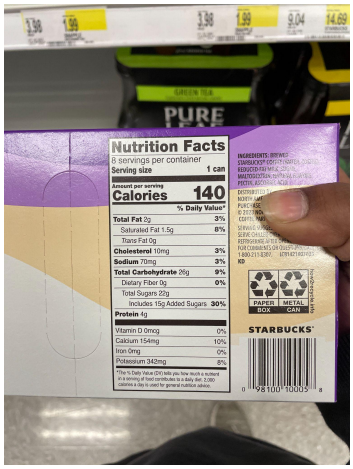
club_crackers.jpg

crackers.jpg

Keyur Patel

cumin.jpg

frappuccino.jpg



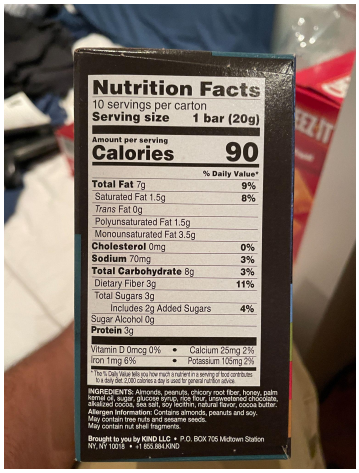
goldfish.jpg

granola_bars.jpg

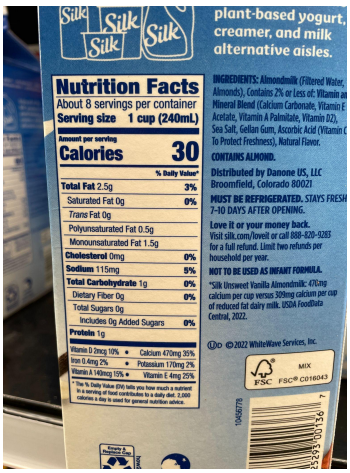
hot_pockets.jpg

ice_cream.jpg

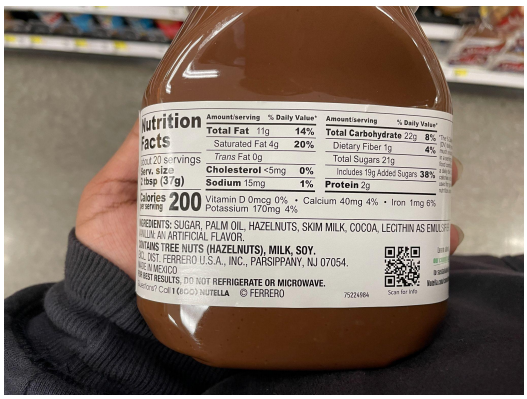
kind_bars.jpg



milk.jpg



nutella.jpg



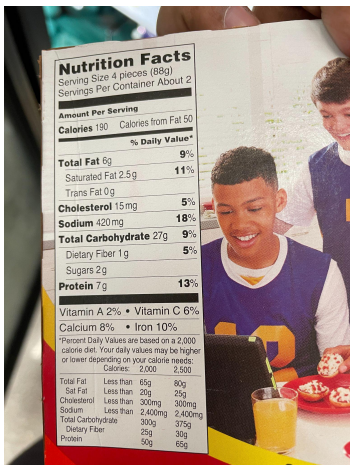
Keyur Patel

oreos.jpg

pasta.jpg

peanuts.jpg

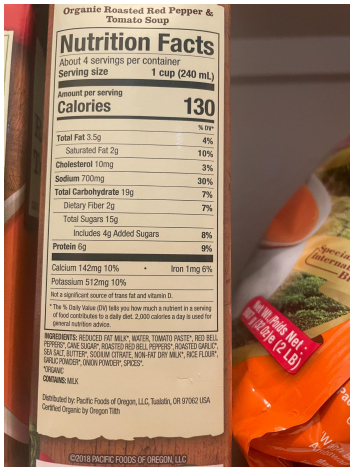
pizza_bagels.jpg



popcorn.jpg

reeses_puffs.jpg

soup.jpg

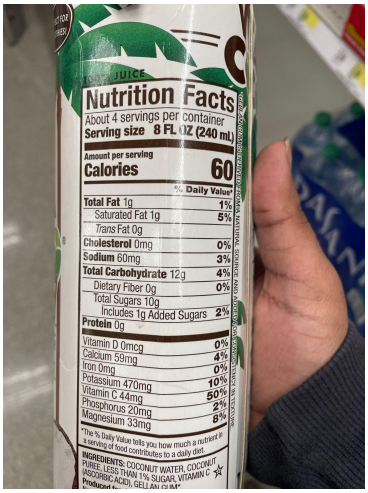


vegetable_stock.jpg

Rounded:

bbq.jpg

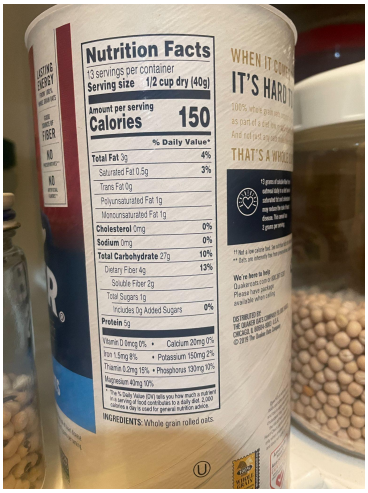
coconut_water.jpg



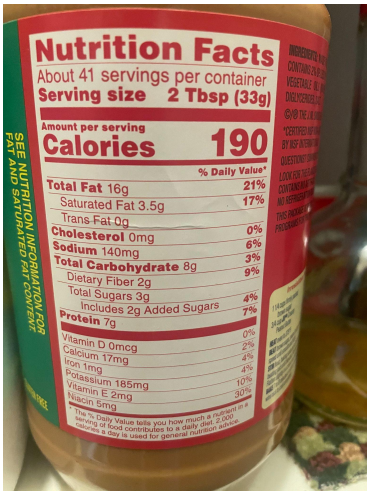
jelly.jpg

mayonnaise.jpg

oats.jpg

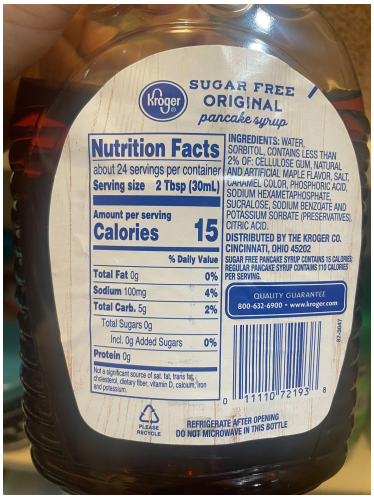


peanut_butter.jpg

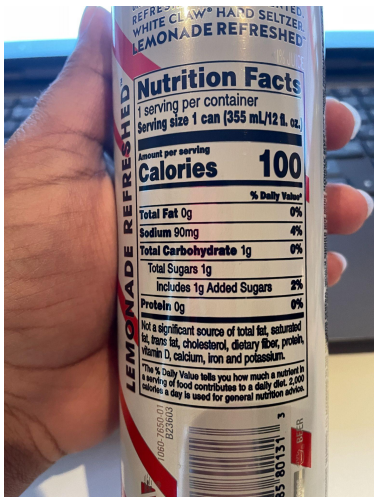


salad_dressing.jpg

syrup.jpg



whiteclaw.jpg



Keyur Patel

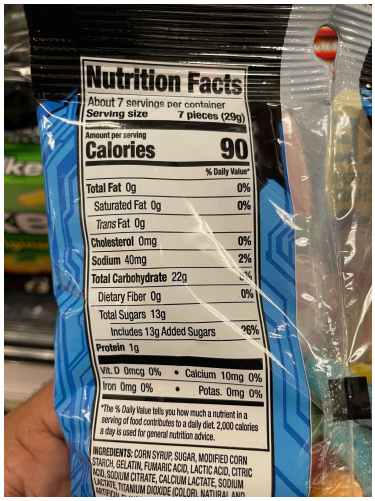
Bags:

almonds.jpg

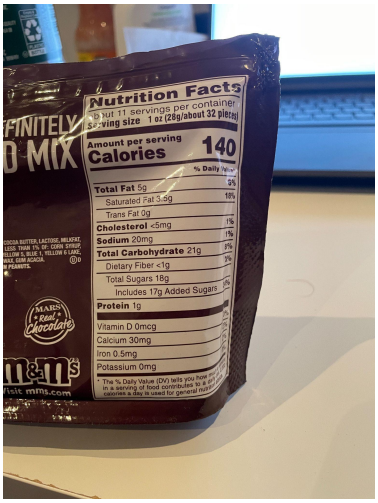
bread.jpg

doritos.jpg

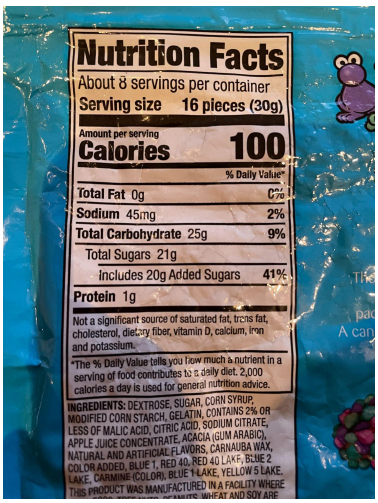
gummy_worms.jpg



m&ms.jpg



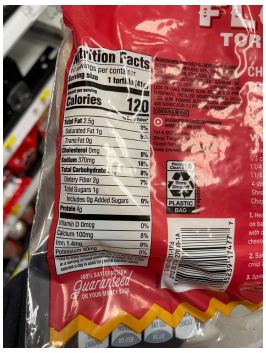
nerds.jpg



pretzel_chips.jpg

sun_chips.jpg

tortillas.jpg



walnuts.jpg