Snap-n-Snack: a Food Image Recognition Application

Stephen Hoff  
*Santa Clara University, shoff@scu.edu*

Patterson Jaffurs  
*Santa Clara University, pjaffurs@scu.edu*

Michael Enriquez  
*Santa Clara University, menriquez@scu.edu*

Quintin Wilde  
*Santa Clara University, qwilde@scu.edu*

Follow this and additional works at: [https://scholarcommons.scu.edu/cseng_senior](https://scholarcommons.scu.edu/cseng_senior)

Part of the [Computer Engineering Commons](https://scholarcommons.scu.edu/cseng_senior)

**Recommended Citation**  
Hoff, Stephen; Jaffurs, Patterson; Enriquez, Michael; and Wilde, Quintin, "Snap-n-Snack: a Food Image Recognition Application" (2018). *Computer Engineering Senior Theses*. 121.  
[https://scholarcommons.scu.edu/cseng_senior/121](https://scholarcommons.scu.edu/cseng_senior/121)
SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: June 12, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Stephen Hoff
Patterson Jaffurs
Michael Enriquez
Quintin Wilde

ENTITLED

Snap-n-Snack:
a Food Image Recognition Application

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Thesis Advisor

Department Chair
Snap-n-Snack:
a Food Image Recognition Application

by

Stephen Hoff
Patterson Jaffurs
Michael Enriquez
Quintin Wilde

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 13, 2018
Snap-n-Snack:  
a Food Image Recognition Application

Stephen Hoff
Patterson Jaffurs
Michael Enriquez
Quintin Wilde

Department of Computer Engineering
Santa Clara University
June 13, 2018

ABSTRACT

Many people desire to be informed about the nutritional specifics of the food they consume. Current popular dietary tracking methods are too slow and tedious for a lot of consumers due to requiring manual data entry for everything eaten. We propose a system that will take advantage of image recognition and the internal camera of Android phones to identify food based off of a picture of a user’s plate. Over the course the last year, we trained an object detection model with images of different types of food, built a mobile application around it, and tested their integration and performance. We believe that our program meets the requirements we set out for it at its conception and delivers a simple, fast, and efficient way of tracking one’s diet.
# Table of Contents

1 Introduction .......................................................... 1

2 Requirements .......................................................... 2
   2.1 Functional .......................................................... 2
   2.2 Non-Functional ..................................................... 2
   2.3 Design Constraints ............................................... 2

3 Use Cases .................................................................. 3

4 Activity Diagram ....................................................... 6

5 User Interface .......................................................... 7
   5.1 Create Account/Login ............................................... 8
   5.2 Main Menu ............................................................. 9
   5.3 Initial Picture Screen ............................................... 10
   5.4 Camera Preview ..................................................... 11
   5.5 After Submitting Picture .......................................... 12
   5.6 After Clicking on Food Indicator Button ...................... 13
   5.7 Food Journal Day Selection ..................................... 14
   5.8 Food Journal Food Per Day ...................................... 15

6 Technologies .......................................................... 16

7 Architecture ................................................................ 17

8 Design Rationale ....................................................... 18
   8.1 GUI ................................................................. 18
   8.2 Technologies Used ................................................ 18

9 Risk Analysis .......................................................... 20

10 Development Timeline ............................................... 21

11 Image Recognition Model .......................................... 24
   11.1 Image Gathering and Preprocessing ......................... 24
   11.2 Model Training .................................................... 25
   11.3 Iteration ............................................................ 26
   11.4 Model Design ...................................................... 27

12 Test Results .......................................................... 29
   12.1 Preliminary Unrefined Testing ................................. 29
   12.2 Secondary Refined Testing ................................... 31
      12.2.1 Analysis of Results ...................................... 34
13 Conclusion
   13.1 Lessons Learned ......................................................... 35
   13.2 Future Work ................................................................. 36

Appendices
   A User Manual ...................................................................... 38
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Potential actions available to both users</td>
<td>3</td>
</tr>
<tr>
<td>4.1</td>
<td>Activity diagram showing the logical options a user can follow when interacting with the system.</td>
<td>6</td>
</tr>
<tr>
<td>5.1</td>
<td>Sign in screen with options for both standard email/password login as well as login via Google account.</td>
<td>8</td>
</tr>
<tr>
<td>5.2</td>
<td>The main menu of the application with options to go to either the camera or food journal portions of the application.</td>
<td>9</td>
</tr>
<tr>
<td>5.3</td>
<td>The view the user sees after selecting the camera button.</td>
<td>10</td>
</tr>
<tr>
<td>5.4</td>
<td>Camera activity of the phone</td>
<td>11</td>
</tr>
<tr>
<td>5.5</td>
<td>After submitting the picture, the page will display a food indicator on top of any recognized foods on the plate.</td>
<td>12</td>
</tr>
<tr>
<td>5.6</td>
<td>Results are displayed on a separate page after selecting one of the food indicator circles.</td>
<td>13</td>
</tr>
<tr>
<td>5.7</td>
<td>Page after clicking on the Food Journal button from Figure 5.2.</td>
<td>14</td>
</tr>
<tr>
<td>5.8</td>
<td>List of food for a given day.</td>
<td>15</td>
</tr>
<tr>
<td>7.1</td>
<td>Client-Server architecture representing the data stored on each and how they interact</td>
<td>17</td>
</tr>
<tr>
<td>10.1</td>
<td>The projected development timeline for our project during the Fall and Winter Quarters.</td>
<td>22</td>
</tr>
<tr>
<td>10.2</td>
<td>The projected development timeline for our project during the Spring Quarter.</td>
<td>23</td>
</tr>
<tr>
<td>11.1</td>
<td>Depiction of a tagged image of green beans using LabelImg.</td>
<td>25</td>
</tr>
<tr>
<td>11.2</td>
<td>Total loss for our first model for recognizing only baby carrots.</td>
<td>26</td>
</tr>
<tr>
<td>11.3</td>
<td>Total loss of our final model, able to recognize baby carrots, green beans, and chicken legs.</td>
<td>27</td>
</tr>
<tr>
<td>12.1</td>
<td>Average confidence values for carrots on a plate with no other food.</td>
<td>30</td>
</tr>
<tr>
<td>12.2</td>
<td>Average confidence values for carrots on one half of a plate that is also occupied by green beans.</td>
<td>30</td>
</tr>
<tr>
<td>12.3</td>
<td>Average confidence values for green beans on a plate with no other food.</td>
<td>30</td>
</tr>
<tr>
<td>12.4</td>
<td>Average confidence values for green beans on one half of a plate that is also occupied by baby carrots.</td>
<td>31</td>
</tr>
<tr>
<td>12.5</td>
<td>Second round of testing with carrots being the only food on the plate.</td>
<td>31</td>
</tr>
<tr>
<td>12.6</td>
<td>Confidence scores of green beans isolated on a plate.</td>
<td>32</td>
</tr>
<tr>
<td>12.7</td>
<td>Confidence values for four chicken legs on a plate.</td>
<td>32</td>
</tr>
<tr>
<td>12.8</td>
<td>Average confidence values of each food category when on a single plate all together captured from a birds-eye-view.</td>
<td>33</td>
</tr>
</tbody>
</table>
List of Tables

9.1 Table of significant potential risks to the project and mitigation strategies . . . . . . . . . . . . . . . . 20
Chapter 1

Introduction

Dietary tracking is an increasingly popular practice for maintaining a healthy lifestyle. Current products track everything from weight and serving size to caloric breakdowns in fats, proteins, and carbohydrates. However, to get these accurate readings, people need to identify the breakdown of their recipes into the separate ingredients and enter them into the application. This tedious data input from the user is unappealing to modern consumers, and many users stop using these products due to time constraints.

There are several common options in the app marketplace right now that assist people with dietary information, however these options are slow and frustrating, typically ending in the user deleting the app. These problems stem from an overly complex system of entering food data for precise results, such as the weight of the meat or vegetables or the volume of liquid consumed. To actually get these numbers, people would have to use a scale or check the bag for details if their food is from a package. This process is made easier by companies allowing for barcodes to be scanned, however this doesn't work for recipes and homemade foods. On the market today there exists at least one product that attempts to solve this issue, however, that product struggles with efficiency and has so many features that users may find complicated and unnecessary.

We propose an application that will be able to relay basic dietary information about food to the user by taking advantage of their phones internal camera alongside image recognition software. The user simply takes a picture of their meal, and our application will identify the food on their plate and provide basic dietary information about it. Additionally, the system will be able to identify the approximate serving size of each item on the users plate, so they can have a more complete picture of the details behind their meal. Our solution is meant to emphasize convenience and speed, as well as being lightweight so as not to slow down the user's phone.

Food and technology are important parts of everyone's lives, and establishing a common ground between them could make managing health and diet easier than ever. The ability to take a picture of food and identify its dietary content on the go would be an important asset to athletes, bodybuilders, or the average health-conscious individual from around the world. Our solution serves to bring simplicity and convenience to the ever-important task of staying healthy.
Chapter 2

Requirements

In this section we discuss the requirements and constraints that our application was to meet during implementation. The Functional requirements detail what precisely our system does, while the Non-Functional requirements deal with how it performs. To illustrate each requirement’s priority, all are enumerated within their distinct sections. We also discuss the Design Constraints that served as limiting factors on the implementation of this project.

2.1 Functional

1. The application will make use of its device’s camera to take photos for analysis.
2. The application will correctly identify foods in a given image.
3. The application will calories per serving of identified food.
4. The application will store users’ food history and associated data in a Food Journal.
5. The application will allow users to make accounts and securely sign in.

2.2 Non-Functional

1. The application will be accurate in its predictions.
2. The application will be easy to use.
3. The application will be efficient and lightweight.
4. The application will return results within a reasonable time-frame.

2.3 Design Constraints

1. The application will run on Android mobile devices.
2. The application will only function while connected to the Internet.
Chapter 3

Use Cases

There are two actors for this application. The first is the user, who has a set of very specific actions they can perform as shown in Figure 3.1. Most of the actions they can take are only possible after the user has taken a picture first. The second actor is an admin/developer who is able to tweak the image training model and the server.

Figure 3.1: Potential actions available to both users
1. **Register**
   - Actors: User
   - Goal: User makes an account for the application.
   - Precondition: User has installed the application.
   - Postcondition: User added to user list.
   - Exception: Entering in a currently used username will cause an error and force the user to choose a new one.

2. **Sign In**
   - Actors: User
   - Goal: User enters username and password into application to gain access to their account.
   - Precondition: User must have an account.
   - Postcondition: User gains access to account information and application functionality.
   - Exception: If the user enters the wrong username or password, an error will be returned that either the username or password was incorrect.

3. **Take Picture**
   - Actors: User
   - Goal: User takes a picture of their plate using native phone camera.
   - Precondition: User is registered and signed in to their account.
   - Postcondition: Detection results will be shown to the user after analysis from the server.
   - Exception: If the user does not allow camera access, an error will be sent to the user with a direction to allow access to the camera.

4. **Add Food Item to Journal**
   - Actors: User
   - Goal: User adds the detected food item to their food journal.
   - Precondition: The picture has been returned to the user and labeled by the database.
   - Postcondition: The food journal will have been updated with the new value on the date the entry was added.
   - Exception: If no items were detected the user will be unable to add anything.

5. **Review Food Journal**
   - Actors: Admin
   - Goal: User can review entries from current/previous days they have entered data.
   - Precondition: User must have added data to the food journal before hand.
   - Postcondition: N/A.
   - Exception: N/A.

6. **Train Model**
   - Actors: Admin
   - Goal: Admin will create a new image recognition model.
   - Precondition: Admin must have the technologies and training data necessary to create a new image recognition model (discussed in Technologies Used).
   - Postcondition: Admin will have a new image recognition model they can add to the server.
   - Exception: N/A.

7. **Update Model Being Used**
   - Actors: Admin
   - Goal: Swap current image recognition model running on the server with new model.
   - Precondition: Server has a model already in place, and Admin has a new model to replace it.
Postcondition: Server will use the new model.
Exception: N/A.

8. **Restart Server With New Model**
   Actors: Admin
   Goal: Server will be up and running with a new image recognition model.
   Precondition: Image recognition model was updated on the server beforehand.
   Postcondition: Server will be actively running with new model and available to connect to.
   Exception: N/A.

9. **Configure Server**
   Actors: Admin
   Goal: Set up and maintain server.
   Precondition: Need access to a server which can run a python file that implements an image recognition model.
   Postcondition: Server will be set up to receive input from users, and will send back JSON data to be interpreted by the application.
   Exception: Server needs to have a static stable IP and a port available for the application to connect to.
Chapter 4

Activity Diagram

The activity diagram shows the different paths that a user can follow when interacting with the application. The main paths include using the food journal, the camera portion of the application, login/account creation, and exiting the application.

Figure 4.1: Activity diagram showing the logical options a user can follow when interacting with the system.
Chapter 5

User Interface

This section contains screenshots of our application’s Graphical User Interface (GUI). Each section will showcase a different page in the system and the details that follow it describe its purpose and functionality. The design of each page is mainly focused on the core functionality as improvements to the actual look and feel of the interface can be improved later as a future work.
5.1 Create Account/Login

Figure 5.1: Sign in screen with options for both standard email/password login as well as login via Google account.

Upon launching the application the user will see the form shown in Figure 5.1. If they have not previously created an account the user will be asked to create one. Once logged in, the application will skip past this page unless the user chooses to sign out.
5.2 Main Menu

Figure 5.2: The main menu of the application with options to go to either the camera or food journal portions of the application.

The main menu shown in Figure 5.2 will be displayed after login. For now the menu only has two options, the camera and food journal button. Pressing either will take the user to the specified portions of the application.
5.3 Initial Picture Screen

Figure 5.3: The view the user sees after selecting the camera button.

Figure 5.3 shows instructions for how users should frame their food plate when taking their food picture to achieve optimal results. Besides that, the view also includes a button for the user to press once they have read the instructions fully and are ready to take the picture.
5.4 Camera Preview

Figure 5.4: Camera activity of the phone

The image shown in Figure 5.4 shows a camera preview from the native camera activity on a phone. This is specific to the model of the android phone so this portion of the application will differ from user to user. After framing the image the user is able to click the OK/check-mark button and the application will proceed to the next portion of the image recognition process.
5.5 After Submitting Picture

After submitting the picture, the page will display a food indicator on top of any recognized foods on the plate.

Figure 5.5: After submitting the picture, the page will display a food indicator on top of any recognized foods on the plate.

After submitting the picture the system will attempt to determine the content of the plate and place food indicators on top of the picture which are shown in Figure 5.5 as blue circles. Clicking on the circles will take the user to another page to view details about what food was found and how confident that recognition is. If no food items are detected the user will have to return to the main menu and try again.
5.6 After Clicking on Food Indicator Button

Figure 5.6: Results are displayed on a separate page after selecting one of the food indicator circles.

After selecting one of the food indicator circles if any are displayed, a separate page will be opened which names the food item the image recognition software has the highest confidence in as well as the degree of confidence. The user is also given the option to add the recognized item to their food journal before being returned to select any other food items that were identified in the submitted image.
5.7 Food Journal Day Selection

Figure 5.7: Page after clicking on the Food Journal button from Figure 5.2.

The first part of using the food journal. User is presented a list of all days that they have entered data previously. Selecting any of the days listed will allow the user to view the various items they added on that given day.
5.8 Food Journal Food Per Day

A page is displayed to the user based on the day selected from Figure 5.7. This page will show any food items that the user added to their journal through the camera functionality for that given day.

Figure 5.8: List of food for a given day.
Chapter 6

Technologies

As this is a mobile application that makes use of databases and cloud computing, we had to draw from a wealth of technologies before deciding which we wanted to use. We knew our program would be an Android app and that many machine learning frameworks use Python, but needed to find the library that would give us the best results within a good time frame. We also wanted to ensure that the components of the project accessed from the Internet would have a reasonable response time. What follows is a short list of the programs that we decided to use.

- Android: A popular open-source cellphone operating system, known for its ease of use and development.
- Java: An object-oriented programming language and the primary language used for Android application development.
- Python: A high-level scripting language widely-used in data science applications.
- Google Compute Engine (GCE): An on-demand cloud computing platform provider.
- JSON: A data format often used when dealing with databases and machine learning.
- Firebase: A mobile application development and testing platform.
Chapter 7

Architecture

Our system uses a client-server architecture because we wanted to separate the functionalities of the client program and the server database. The client side has the user take images and sends them to the server for processing. The server then examines the image using the image recognition model and returns the suite of information related to the identified foods so they can be displayed back to the client.

The machine learning model itself was developed and trained on our personal computers, separate from this system. The server uses this model to analyze the images sent to it. Additionally, user data will be stored on a Google Firebase data instance tied to each individual account. The diagram representing our chosen architecture can be found in Figure 7.1.

Figure 7.1: Client-Server architecture representing the data stored on each and how they interact
Chapter 8

Design Rationale

This section explains why we made certain choices in the conceptualization of the Graphical User Interface and why we chose to use certain technologies for this project. Its purpose is to make our decisions transparent to clients or users and to serve as a logic check for future decisions.

8.1 GUI

The GUI was made using Android Studio’s Layout Editor since it is a built-in feature for Android Studio and provides many useful tools for the creation of dynamic activities.

- The application makes use of the default camera intent on the user’s phone because it should be a familiar interface that the user understands how to use and gives them control over how to use their camera, although we do offer suggestions on how to achieve the best possible results.

- Food Indicator Selector: This is used to separate the foods for the user and distinguish between them in as clear a manner as possible, as well as allowing the user to select the food by simply tapping it. Also lets the user know which areas of the plate it considered associated with those food items.

- The food journal interface is fairly simple because the focus of the project was to accomplish food-based image recognition based on pictures taken of a plate by the user. Although we wanted to make sure the user had something they could do with the data that was found by our system and be able to add the discovered items to a food journal, the actual functionality and design of this feature did not require significant development besides base functionality.

8.2 Technologies Used

- Android: Because Android is an open-source platform, it will be easier for us to develop an application on it, due to the wealth of libraries and resources available. It is also widely-used, so it will be easier to find people willing to test our application in the future.

- Java: Since we are developing on Android, Java is really the best choice for application development, as it is the main language for doing so. Additionally, several of our team members are proficient with the language either through experience or the language’s syntactical similarity to C++.
- **Python**: We used Python because it is extremely useful for data science applications, as well as easy to read and write. The language also has extensive libraries for machine learning and image recognition, both of which are instrumental to our project.

- **TensorFlow**: TensorFlow is one of the aforementioned machine learning Python libraries and was invaluable to us for its ability to make use of the computer’s Graphics Processing Unit to speed up model training. Tensorfow’s Image Recognition API was the primary subset of the library that we used, thanks to its variety of starting model templates and optimization options.

- **Google Compute Engine (GCE)**: As we desired our application to be reasonably fast and lightweight, we needed our machine learning computations to be done outside of the phone. Thus, we needed a cloud server to analyze images and send results back to the user, while maintaining 100 percent uptime and scalability.

- **JSON**: We used JSON files as our preferred format for data packets sent back from GCE, because it is easily readable across programming languages. The “dictionary” style formatting of JSON makes it very helpful for sending unordered coordinate and image classification data.

- **Firebase**: Firebase was useful for integrating Google account authentication in our application so that we did not have to develop or store authentication services ourselves. We also used Firebase to store each user’s Food Journal activities, upload images, and keep track of information associated with each food our model could recognize.
Chapter 9

Risk Analysis

This section displays the most significant risks we had to potentially handle on the project and how we would deal with them, shown in Table 9.1. The most prominent error was bugs, which inevitably happen despite all programmers’ best efforts, and even the best mitigation strategy won’t stop them all. This means that bugs caused us to be delayed more than any other risk possible, since bugs can take anywhere from minutes to days to fix. The other most pressing risks have smaller probabilities of happening but more disastrous consequences if they happen, with the exception of "Miscommunicated Objectives" which was unlikely to happen given the proximity of the team members to each other and frequency of meetings to date.

Table 9.1: Table of significant potential risks to the project and mitigation strategies

<table>
<thead>
<tr>
<th>Event</th>
<th>Consequence</th>
<th>Probability</th>
<th>Severity</th>
<th>Impact</th>
<th>Mitigation Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bugs</td>
<td>Part or all of program doesn’t work</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>Understand code purpose before starting, perform proper testing at each stage</td>
</tr>
<tr>
<td>Project Unfeasible</td>
<td>Have to re-work project or turn in no working project</td>
<td>.2</td>
<td>10</td>
<td>5</td>
<td>Ensure work is doable, document actions, research similar projects</td>
</tr>
<tr>
<td>Run out of time</td>
<td>Unable to complete project before the due date</td>
<td>.4</td>
<td>9</td>
<td>3.6</td>
<td>Outline project thoroughly and communicate constantly</td>
</tr>
<tr>
<td>Code Loss</td>
<td>Forced to reproduce code, lose time</td>
<td>.3</td>
<td>7</td>
<td>2.1</td>
<td>Save code and upload to Github</td>
</tr>
</tbody>
</table>
Chapter 10

Development Timeline

Figures 10.1 and 10.2 show a Gantt Chart of our estimated development timeline. The tasks identified in the timeline were be completed by the group members specified by color. We managed to follow the timeline to an acceptable degree.
Figure 10.1: The projected development timeline for our project during the Fall and Winter Quarters.
Figure 10.2: The projected development timeline for our project during the Spring Quarter.

<table>
<thead>
<tr>
<th>Requirements Gathering</th>
<th>Spring Wk 1</th>
<th>Spring Wk 2</th>
<th>Spring Wk 3</th>
<th>Spring Wk 4</th>
<th>Spring Wk 5</th>
<th>Spring Wk 6</th>
<th>Spring Wk 7</th>
<th>Spring Wk 8</th>
<th>Spring Wk 9</th>
<th>Spring Wk 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual Phase and Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camera Functionality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Menu Design and Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frontend Backend Communication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype Testing and Debugging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Training</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image Annotation and Model Training</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Retraining</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Deployment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backend Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nutritional Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debugging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In Members</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stephen and Michael</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quintin and Patterson</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem Statement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Prototype</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Draft Comprehensive Project Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design Conference</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complete Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 11

Image Recognition Model

In this section, we will describe the method we used to create and train a machine learning object detection model to recognize food items. In broad terms, this process had several steps: gathering relevant images, tagging locations of food in said images, creating .record files from image data, and passing those files into our chosen model for training. With each iteration, we added either a new food type or more images to pre-existing food classes. At the end, we will also describe how our model itself recognizes food items.

11.1 Image Gathering and Preprocessing

In this stage, we manually scraped the web for images containing the food that we wanted our model to be able to recognize. In doing this, we generally searched for one food at a time, making sure to get a variety of images. For each food, we wanted images that represented many ways of depicting said food, so a range of camera angles, a mixture of other foods on the plate, and multiple ways of preparing the same food were all musts. However, we made sure to only use images where the food in question was fully visible and identifiable with the naked eye. Each food had roughly two hundred images apiece, with extra images containing multiple objects, so as to establish a solid dataset.

Once our batch of images was downloaded, we used a program called LabelImg to tag the locations of all food objects present in our model in each of the images. This was done by drawing a box over that location, the coordinates of which were then stored in a .xml file. These tags inform the model of the ground truth location of food items and so are very similar in appearance to our model’s predictions. An image of this is shown in Figure 11.1.
Next, we split the cache of images into training and testing sets, of ninety and ten percent respectively. The training set is used to teach the model the underlying patterns of the dataset, while the test set is used to measure its effectiveness. For each set, we combined all .xml files for their images into two .csv files for easier processing, before transferring all of the data there plus raw image data into a pair of .record files required by Tensorflow. These records store things like image dimensions, image classes, and formats.

### 11.2 Model Training

Most of what the model actually does is found within a .config file specifying what it is to do and how it is to do it, in accordance with the Object Detection API. This can be modified as necessary before training proper, but will be discussed in greater detail in Section 12.4. Here, we simply run a Python script with the configuration file and the two record files and training will begin.

Training this model is effectively a very long series of iterations, as it uncovers the underlying patterns of the dataset and learns to recognize food objects. Using Tensorflow’s monitoring tools, Tensorboard, we were able to observe the progress of our model graphically. Chiefly, we watched the total loss, or how far the current model was from finding the correct location and class of food in each image in the dataset. A representation of this for our first model can be seen in Figure 11.2.
Choosing when to stop training is a key part of this process; too early and you may not capture the data’s pattern, too late and you may overtrain, preventing the model from recognizing anything besides what it knows. In general, you want to stop when the model reaches an average loss of 1.0, which indicates that the loss function has converged and the model should understand the pattern. In our first case, convergence took about four hours and 1.7 thousand steps, although this was the first of many iterations.

11.3 Iteration

The next stage of training is building off of a previous model to improve on its weaknesses and broaden its capabilities. Overall, we went through five iterations of our image recognition model, each time adding more and better pictures to our dataset. Each passing iteration increases its complexity and with it the time and steps needed to converge. For the sake of brevity, we will show only our most recent model in comparison to our first, shown in Figure 11.3.
This model took more than fifteen hours to converge in twenty-nine thousand steps, a massive increase over the first version. A cursory look at the graph shows far more peaks and valleys than the first, owing mostly to its increased complexity. Being able to identify three food types is, of course, more difficult than only one, especially considering the much larger dataset present in this case. Even still, it reached an average total loss of about 1.0 towards the end of training and is quite accurate, as viewable in our Test Results section.

11.4 Model Design

The last thing to discuss in this section is the functionality of this model. It is, at its core, a modified version of the SSD Mobilenet V1 object detection model. This was provided with the API in the form of several files, including the configuration file that specifies its function. In short, it creates a Convolutional Neural Network (CNN) that images pass through only a single time, to improve efficiency. The CNN uses regression to localize a series of bounding boxes to the location it believes a food item is located. It then classifies the objects inside such boxes to one of the food types it can identify, alongside a confidence value. If this confidence exceeds a certain threshold, both location and class are used as the model’s answers for that image. The total loss presented in previous images is the L2-Norm Loss, or the Euclidean Distance between the model’s guess and the ground truth for each image. Errors in classification, confidence, and locational accuracy all have a component in this total loss.

Our modifications to this model were mostly for performance. For example, we reduced the number of images examined at a given time, or the batch size, to fix some memory issues that prevented training from occurring. Few changes were made for optimization purposes, as the model already fit very well with what we intended to do with
it, as it was made for high-speed object detection with mobile device support. However, with greater knowledge of machine learning and Tensorflow, it would be very possible to further tweak parameters or even totally change model functionality with some changes to the configuration file.
Chapter 12

Test Results

Over the course of implementing our application various sections of the project needed to be tested. This section will focus on the trial runs we ran through after system integration was initially completed as well as the second round of testing which was done after an update to our model.

Camera Variations

- Ideal Position: From our intuition we believed that the “ideal” angle for our application would be for the user to hold the camera above the plate of food pointed down giving a “birds-eye-view.” Additionally, the user should frame the camera so as much of the camera preview is occupied by the plate as possible. All other tests are variations of the Ideal Position.

- Angled Camera: Rather than framing the plate from directly above the user alters the angle by somewhere between thirty to forty-five degrees.

- Low Lighting: Uses the birds-eye-view but picture is taken with low ambient light. Enough light is left that the food is recognizable to the naked eye, but color is significantly darkened.

- Distance: Rather than fully occupy the camera preview with the plate, the distance variation increases the height of the camera by roughly an additional foot compared to where it would be for the Ideal Position.

12.1 Preliminary Unrefined Testing

The following results were done with an actual plate of food, namely being green beans and baby carrots. Up until this point the majority of testing was done by pointing the camera at a computer screen that had baby carrots, green beans, or chicken legs as it seemed unfeasible to do live tests with actual food every time a minor adjustment was made. Once we were satisfied with our preliminary implementation and system integration a basic live test was done.

For the majority of the results shown in Figures 12.1, 12.2, 12.3, and 12.4, our model performed fairly well. Many of the tests scored higher than 90% confidence. It should be noted however that for these initial test results some trials performed too poor to display to the user. Our system only sends results to the user that our model had higher than a 50% confidence score, anything lower than that we deemed not high enough to be reliable. Another potential issue with this first test phase was that the plate being used was a green ceramic plate and we believe that may have impacted the results for the distance and low lighting tests for green beans given the color similarity.
Figure 12.1: Average confidence values for carrots on a plate with no other food.

Figure 12.2: Average confidence values for carrots on one half of a plate that is also occupied by green beans.
12.2 Secondary Refined Testing

Our second phase of testing largely brought improvements to our results mainly due to some minor issues being addressed, such as replacing the green plate from phase one with a white plate, and our image recognition model being updated and improved. Another difference between the first and second phase of testing was the inclusion of chicken legs alongside the baby carrots and green beans already previously tested.
When testing each food category on the plate individually we used all four of the camera situations mentioned above, but we only used the ideal position for testing all three types of food on the plate at one time. When attempting the other camera situations it quickly became clear that the model was unable to consistently recognize all three items on the plate for any given test that was not the ideal situation previously described. As such, we decided not to include these results and focus on how our system performed when recognizing three food categories on a single plate of food using the ideal position.

Figure 12.5: Second round of testing with carrots being the only food on the plate.

Figure 12.6: Confidence scores of green beans isolated on a plate.
Figure 12.7: Confidence values for four chicken legs on a plate.

Figure 12.8: Average confidence values of each food category when on a single plate all together captured from a birds-eye-view.
12.2.1 Analysis of Results

Between the first and second phases of testing our results for Ideal and Angled were largely similar with some slight improvements favoring the second phase of testing. Although we expected Ideal to perform well, the Angled method caught us initially off guard for how well our system performed (at least for a single food category on a plate). After consideration we believe that the high confidence values for Angled (sometimes even outperforming Ideal) actually make sense. Due to the method by which we trained our image recognition model, where we scraped the Internet for images of the different food categories as our training data, it became evident to us that our model would favor an angled approach for a single food category on a plate because the majority of the data set images were pictures taken at such an angle.

For low lighting and distance tests our system had very high scores across all food categories, which can be mainly attributed to the improved model (and having a white plate this time). As mentioned earlier, we initially tried to perform all four camera setups for a plate containing all three categories of food, but our system almost never detected all three categories with above 50% for the other three tests besides the Ideal. This decrease in accuracy makes sense when considering each of the three situations on their own. For the Angled test, having multiple items of food on the plate means that at least one food category will be in the back and harder for our system to see just due to the nature of how depth enters the equation when framing an image in that manner.

The main issue for the Low-Lighting and Distance tests was the same issue present in the Ideal test which gave us overall slightly lowered confidence compared to single food categories by themselves. In order to frame all three groups of food on the plate and fit them all in the frame at the same time, we naturally had to increase the distance away from the plate in order to accommodate all three groups taking up more space. By doing so, we essentially were performing the distance test already, which as can be seen from the test result figures usually had the highest impact on confidence across the board. As a consequence of this our system failed to recognize all three categories for any test that was not using the ideal camera framing, and even in that case our confidence was lower compared to single food items on a plate. This result can also be partially attributed to the fact that our model was not trained with any images that had all three items of food on a plate.
Chapter 13

Conclusion

Our objective was to create an android image recognition application that could identify a few different foods as a proof of concept. In this regard, we succeeded, as we made an application that fulfilled these expectations and passed the tests we set out for the system. This section details what we learned from this project, as well as what we think would be substantial improvements to the application that would make it successful in the real world.

13.1 Lessons Learned

There are several useful things we learned from our time building this application that allowed us to better coordinate with each other and accomplish our goals faster.

1. **Coordinate deadlines constantly with other group members.** The project required that members of the group communicate what each person was doing and when we hoped to have done by certain dates. When we didn’t do this as a group, progress stalled, leading to the project as a whole taking more time than it should have.

2. **The current task always takes longer than expected.** No task of significance takes less time than expected, and deadlines end up being looser approximations than the group would hope. Especially when each part is being worked on by only one person, this also means the group’s individual schedules can drag down productivity and lead to stalls in the schedule, forcing adjustments from the team as a whole. Deadlines should always have some leeway to account for the extra time needed for complications, especially in a project where each member has a section and might not be able to fulfill their duties on time.

3. **Researching the task before starting cuts down on the time necessary to complete the objective.** Looking up the best approach to a problem often saved time for our group, since finding a technology or method that accomplished what we needed quickly and taking the time to integrate it into our application was both faster and simpler than trying to accomplish it ourselves.

4. **Training a model is laborious and finicky.** The model took a very long time to train, which could go for as long as 10-15 hours, and when training a model the developer has to constantly make sure the data is good won’t over train the model and make it unable to handle unusual input. The model still has to be able to identify surprises, and this fact makes training the model difficult and adds difficulty to the back end development.
13.2 Future Work

There are many future developments possible with this project that would make this application competitive in the market.

1. **Take feedback from users.** Currently, while users can reject photos they take and take new ones of their food, they cannot submit pictures of correctly identified food to improve the model. This data would allow for more accurate identification especially for multiple foods on one plate.

2. **Allow users to change to correct food in image.** This way they could use only one photo, provided they got identified foods in the photo, and anything that was incorrect in the image could be changed to the correct option. This could also play into taking feedback from users, since submitting the correct food could improve the model’s accuracy in that situation.

3. **Create a larger pool of identifiable foods.** Right now the prototype can only identify three different foods which isn’t enough for serious use. Adding more food would allow the system to be usable in situations not including chicken, green beans, and baby carrots.

4. **Add detailed dietary information for food in the system.** Currently the data for each food type in the system is limited to calories per serving, and expanding this to include more specific values like grams of protein, carbs, and fat would help make the data more comprehensive, as well as allowing users to track their consumption of these macro-nutrients. This would make the food more interesting as a whole and would allow the user to take greater interest in what there food has in it rather than just logging it for calories.

5. **Improve the capabilities and UI of the food journal.** The food journal could use many quality of life additions, such as making it a calendar instead of a list of days, adding different times of day where the user can add breakfast, lunch and dinner, and log how many portions of the food they had in that meal. The system could also have a health level tracker for the foods in question, which shows how healthy they are for the user based on nutritional content, vitamins present in the food, and other metrics. Not only that, but improving the UI look and feel would help bring it in line with the rest of the application in terms of looks.

6. **Move the model onto the phone.** This would cut down on processing time for the picture since the phone itself would analyze it without having to pass the image over an Internet connection first.

7. **Utilize Firebase ML kit to implement the model on a familiar and easily accessed platform.** This was a recent development since Firebase released a beta version of a machine learning kit that would allow for easy implementation of machine learning models on phones, which would make reproducing our project very easy and most likely more efficient.
Appendices
Appendix A

User Manual

Using the Android Application

1. Download application your android device using the .apk file

2. After Snap-N-Snack has been installed, find it in your app directory and select the icon.

3. After opening the app for the first time, you will be given two options. Connect your account to our database by either using a standard email/password combination or a connected Google account. Follow any additional instructions as they appear on screen.

4. Once signed in you will be on the main menu of the application and can select either the Camera or Food Journal buttons.

5. Selecting the Camera button will show you a list of instructions on how we suggest framing the image of your food when you take the picture to achieve the best possible results. Take a moment to read the instructions before pressing the button at the bottom of your screen to begin taking the picture.

6. Once you press that button you will be using the default camera intent on your phone and should do any steps you normally do when using your particular Android device. As this segment may vary from device to device instructions may or may not be accurate for the following segment.

7. General Instructions for Camera usage: Frame image following directions from previous page, focus image for clarity, and press OK/checkmark/camera icon. Confirm the image to be taken back to the main application.

8. Upon arriving back in the main application wait a moment until the application stops saying that it is analyzing the image. At this point there will be blue food indicator circles placed above any food items that were recognized in the image.

9. Selecting any of these circles will take you to a separate page that will tell you what the system believes that food item to be and with what degree of confidence it has in that answer. At this point you may select either yes or no at the bottom of the screen to add or not add that item to your food journal. Pressing either takes you back to the previous page to select any additional food items.

10. Once you are done reviewing the image press the button at the bottom of the screen to return to the main menu of the application.

11. From here selecting the Food Journal button will take you to a list of days that have had data previously entered. If you are visiting this section for the first time and have not added any data from the Camera portion of the application there will be nothing to select.
12. Selecting any of the days will open another page for you to review any of the food data entered for that given day.

13. On the main menu there is a three dot "expand options" icon in the top right corner that will present the option to sign out of the application if selected.