M3 Report

1. Current State of Project

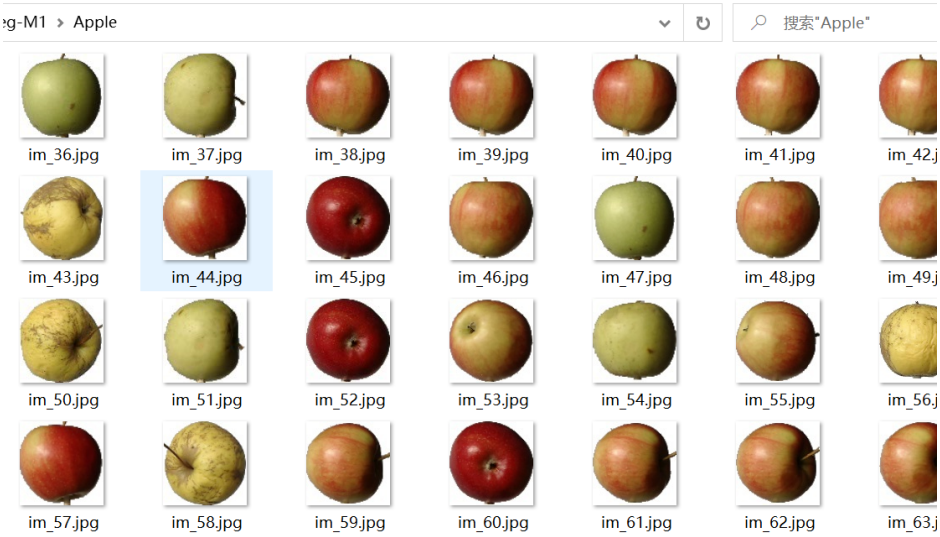
The web app has been deployed and thoroughly tested, and our machine learning model works as expected (issues will be indicated in the "Current Challenges"), but we will continue to make the page look cleaner over the following week. In the end, we decided to use Ting's model for the web app. Additionally, the android app for sending images from a camera is still a work in progress. We will also begin preparing the presentation slides and demo in the following week as well.

2. Changes to Proposal

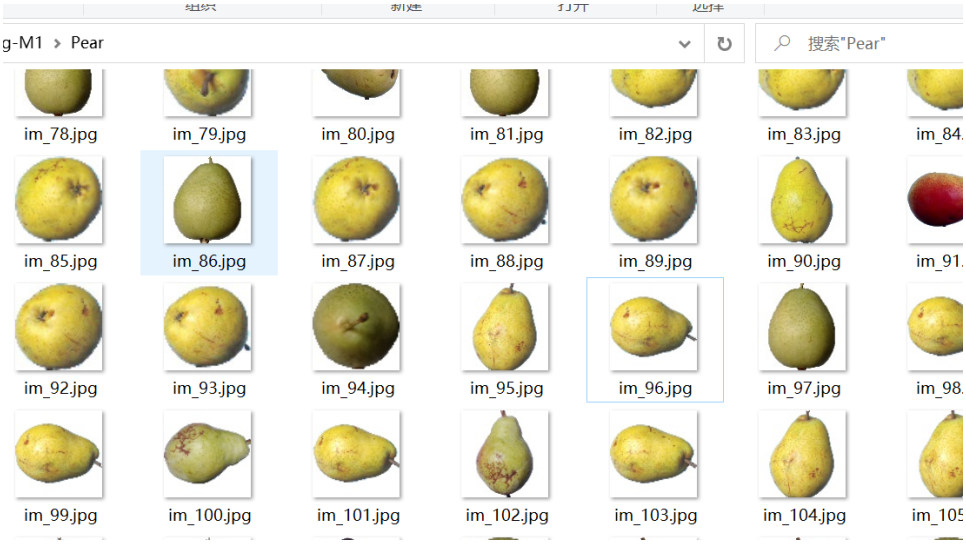
The web-app has been built in ReactJS instead of Angular/Vue because our project was small and did not require heavy routing, scaling, end-to-end testing. React has a lightweight framework and hence better matched our needs. We also decided to remove the brief introduction of the fruit or vegetable since it would've made the web app look cramped.

3. Current Challenges

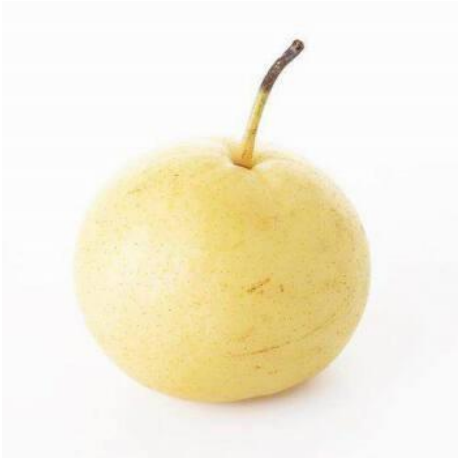
1. The recognition rate for certain fruits with a similar appearance is lower than expected. Here are some examples:
    a. In the dataset, one of the apple species looks very similar to one of the pear types (IMG 1 and IMG2), so the model often confuses the two.



IMG1



IMG2



IMG3

b. IMG3 is an image of a pear, but the model often thinks it's an apple. Similar mispredictions happen in many places, but other problems arise when we try to reduce the likelihood of these mispredictions. We've tried increasing the dataset, but it takes too long to train the model (more than three days), and some types of fruits are difficult for even human eyes' to recognize.

2. Model.pht requires a specific torchvision version, so we've built a virtual environment and written a corresponding requirements.txt to let group members load the model. However, when moving the server to google cloud, setting up the environment might not be necessarily smooth.

3. Spoonacular API sometimes gives recipes without ingredients, or instructions or both. There is also a limit on the number of calls that can be made by spoonacular. (150 requests/day : Free-tier)

4. Team Contributions

Aryan was responsible for building the web-app using ReactJS and integrating APIs — Spoonacular and ML model — into the app. Deployed the entire project onto an AWS machine and tested thoroughly for bugs.

Jeremy has worked on further training his model, but it wasn't picked for the final iteration of the project. He also wrote up a part of the report.

Rushil ported the current frontend react app to android (in order to have image input from camera) - also considering possibilities of ios.

Ting embedded the trained machine learning model into the flask framework and applied a very simple html frontend for testing to see if this combination works and also built a virtual environment for this part.