Team: Jonathon Lin

App Title: Apple Of My Eye

Description: This app allows users to snap a pick at an unknown apple to figure out what type of apple it is. The app will also suggests recipes that can be made with that specific type of apple

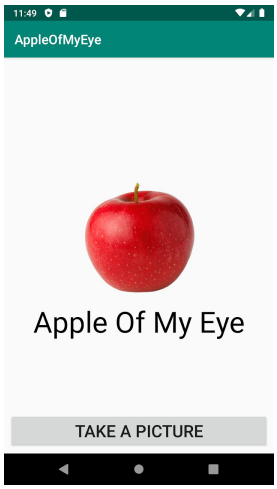
API: EDAMAM Recipe Search API

Third Party:

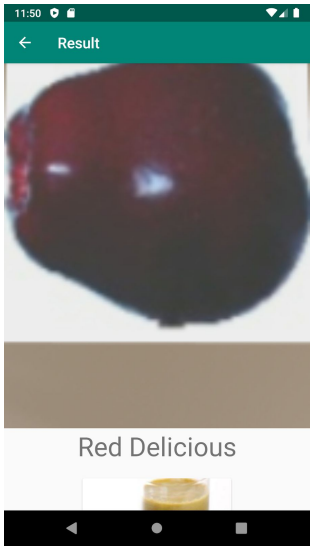
- Pytorch: I used pytorch to script my model so it could be used in Android Studio for the image classification portion of my app. While the importing of the model was easy to do, I noticed that the scripted version does not perform as well as the original in terms of correctly classifying an image. The scripted version has difficulties with apples that are green or yellow and I was not able to figure out why. If I had more time I would like to go back and figure out what's causing this issue.
- Retrofit: I used retrofit to help me make a GET request for the recipe portion of the app. I went with retrofit because I was familiar with it and it's relatively easy to set up. What I did have difficulties with is remembering how to set up the data classes so I get the response from the GET call I made and display it in the app
- Glide: Glide is used to display the images that were returned from the API call. I also chose this because I was familiar with it.
- Google Colab: Due to the hardware restrictions of my machine, I had to use Google Colab to train my model. There is also the added bonus of using CUDA instead of trying to train the model with my machine. Link

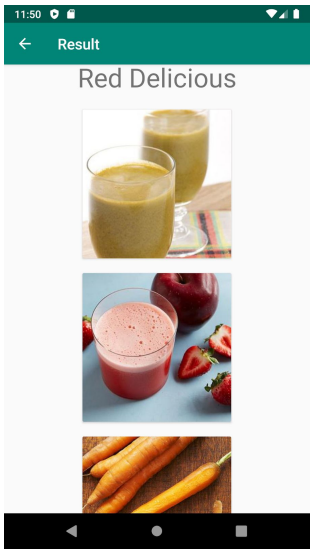
Discussion:

- Users will start out at the home screen which has a button that'll take the user to their camera



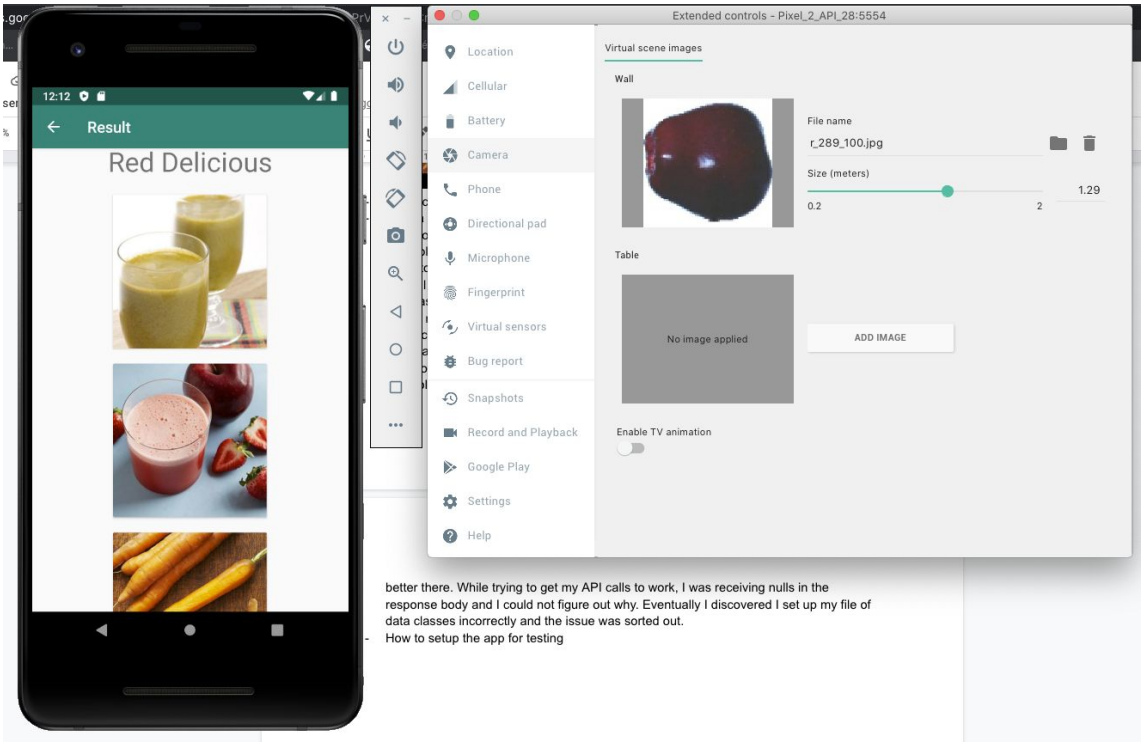
- After the user takes a picture of an apple, they will be taken to the result screen which will display the name of the apple and recipes they can make with that specific type of apple





- Users can click on one of the images and it'll link them to the recipe
- I had a lot of fun creating the model, importing it, and tweaking the classification code. What took me some time to figure out was how to go from camera to classifying but I was able to figure it out once I found out about bitmapToFloat32Tensor. While I was trying to figure out why my model was performing so poorly, I discovered that I made several mistakes. The first mistake was that my labels in the app weren't in the same order as the labels I used to train so my classifier was picking the wrong index. Another issue I ran into was that I did not realize that there was a .DS_STORE file in my data which caused some more indexing issues. Another issue I ran into while testing my app was that I had no efficient way of getting a picture in for the app to classify. I was able to figure out a working solution after doing some digging into the emulator. In the future I'll probably keep the model off of the phone and on the computer since it'll perform a lot better there. While trying to get my API calls to work, I was receiving nulls in the response body and I could not figure out why. Eventually I discovered I set up my file of data classes incorrectly and the issue was sorted out.

- How to setup the app for testing:
    1. Obtain a picture of an apple
    2. Upload picture of apple in the emulators Camera Extended Control



    3. Once you get to the camera, follow the navigation instructions to go through the wall opposite the TV. You'll eventually see the dining room with your picture of an apple on the wall. Try to zoom in as much as possible using the navigation and the Extended Control settings.
- Code breakdown:

Language	files	blank	comment	code
XML	15	31	16	451
Kotlin	9	40	0	240
SUM:	24	71	16	691

I wrote about 691 files according to cloc. If I take away boilerplate lines, external libraries, etc I counted about 640 lines.