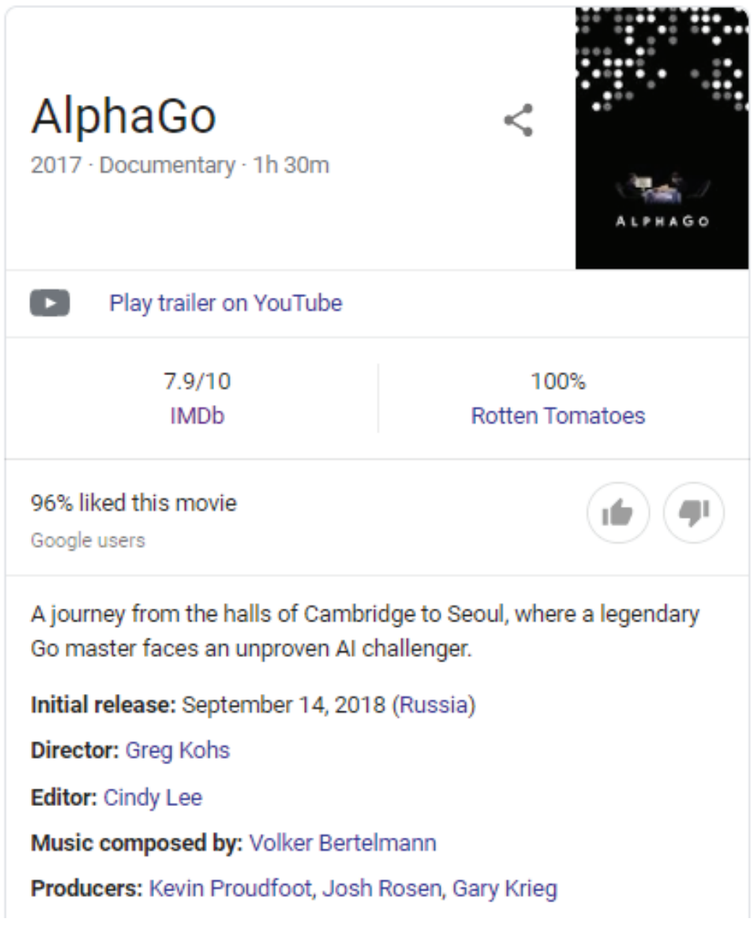
HOW WE DID IT

.CANs process documentation



01 The Inspiration

After being facinated by the AlphaGo documentary, we (Jin and Da) have always been interested in learning about Machine Learning (ML) and making a project using this advanced method.

The project week provided the perfect opportunity for this to happen. And here we want to document our process for future references.

The Making 02

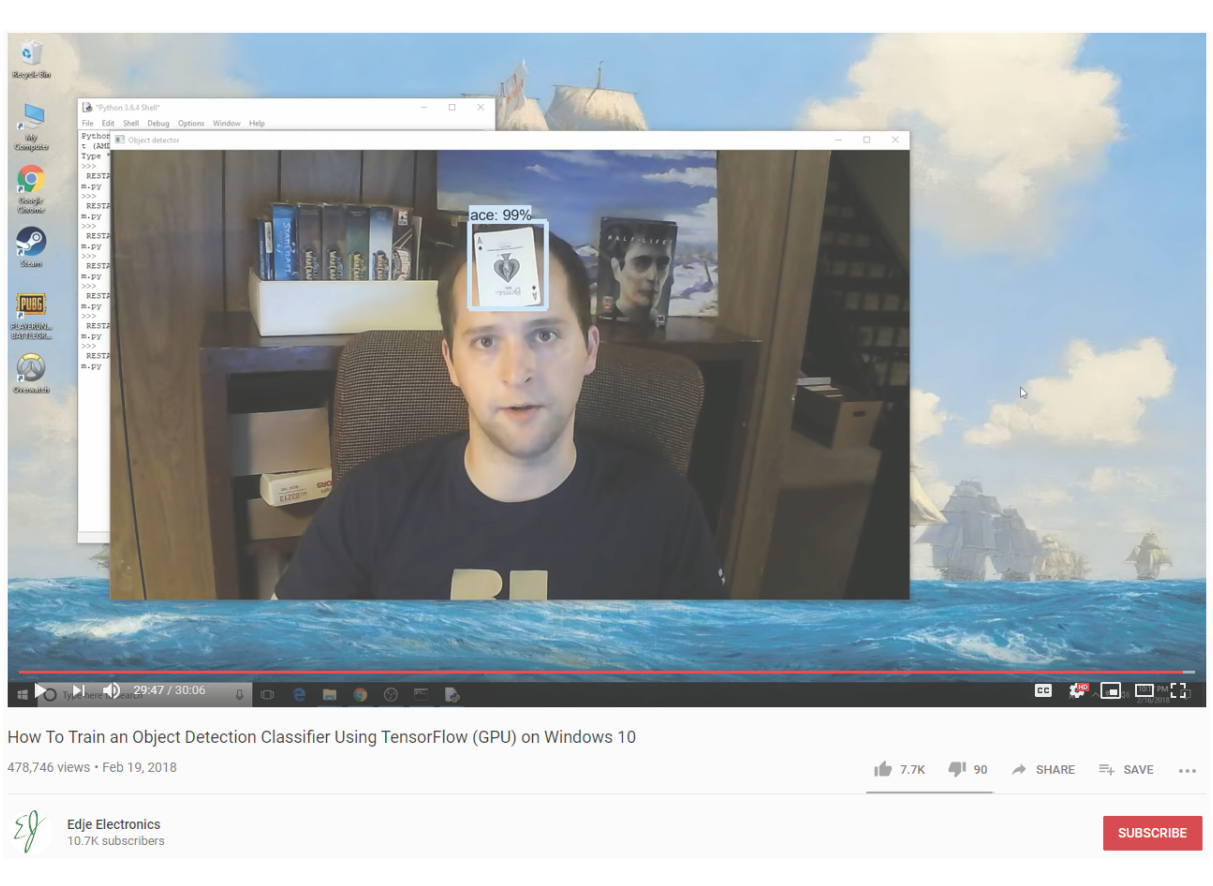
A. Deciding what we want to do with ML

Over the past few years, many types of ML projects have popped up all over the internet. With considerable amount of options, we spent the weekend researching for a project type that can be done within a week to meet the requirements of the CodingDojo's project week. Our final selection is object detection, which seemed to be a great learning opportunity and reasonable in difficulty for new comers.

B. Finding the right framework

Then, we want to narrow down which ML framework we want to use. Upon some research, we found Tensorflow just released their 2.0 version of the framework with exciting new features. With the intergration of Keras, the syntax are much more readable, and, as python programmers put it, "pythonic".

Source: https://dimensionless.in/wp-content/uploads/2019/01/tf-2.0-1080x675.jpg



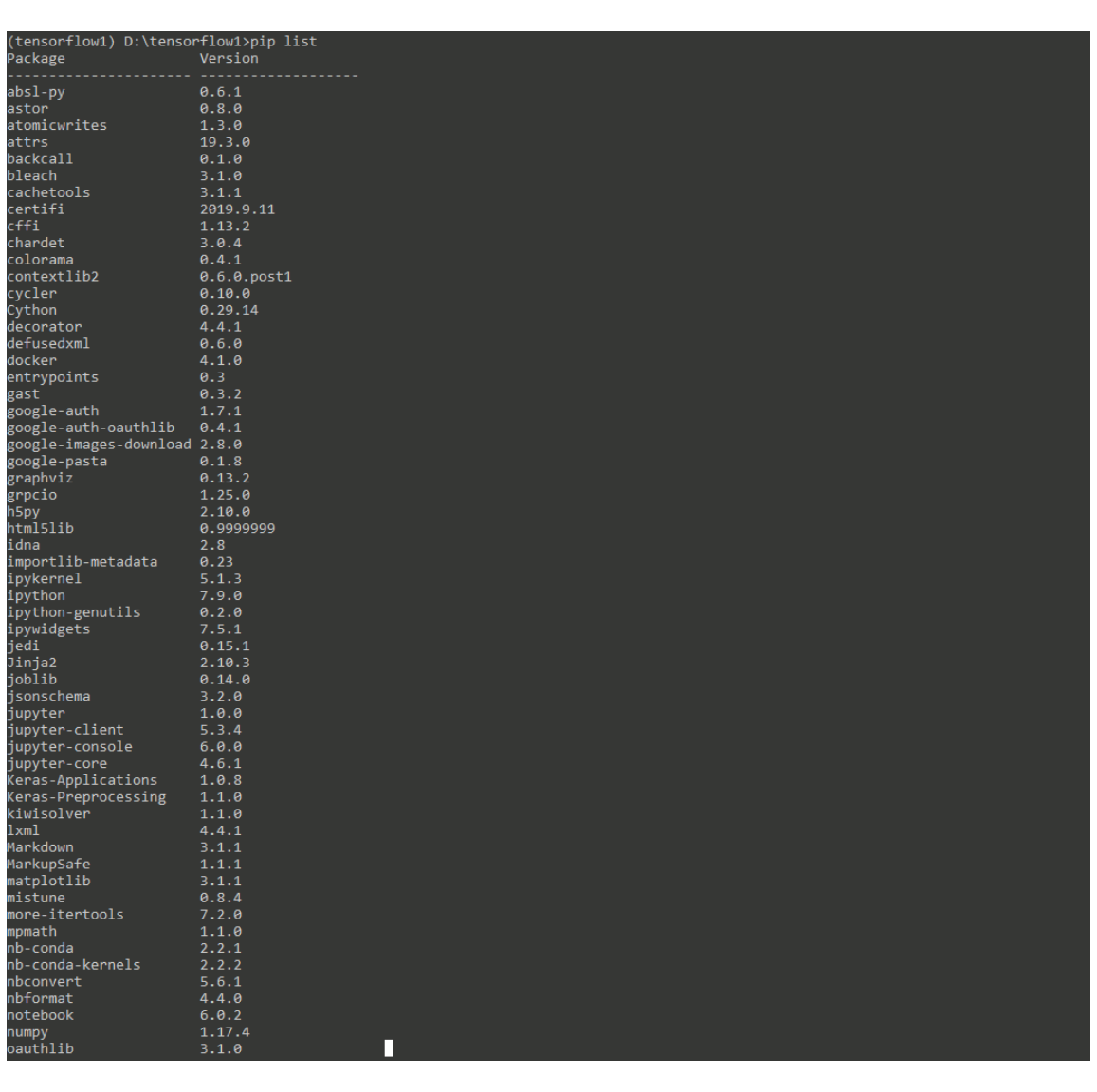
Source: https://www.youtube.com/watch?v=Rgpfk6eYxJA&t=1782s

C. Finding a good tutorial

With no prior knowledge in ML, it is quite crucial to find a good tutorial that can guide us from start to finish. In our case, credits to "Edje Electronics" on youtube, he made a very comprehensive guide to how he did his poker card classifier, and we have learned a great deal from his example.

D. Start our own project

After seeing the entire process of building an object detection model, we felt comfortable to start training our own model. We wanted to build a simple yet unique classifier, so we decided to just grab the 7oz soda cans scattered through our desks and make a can classifier.



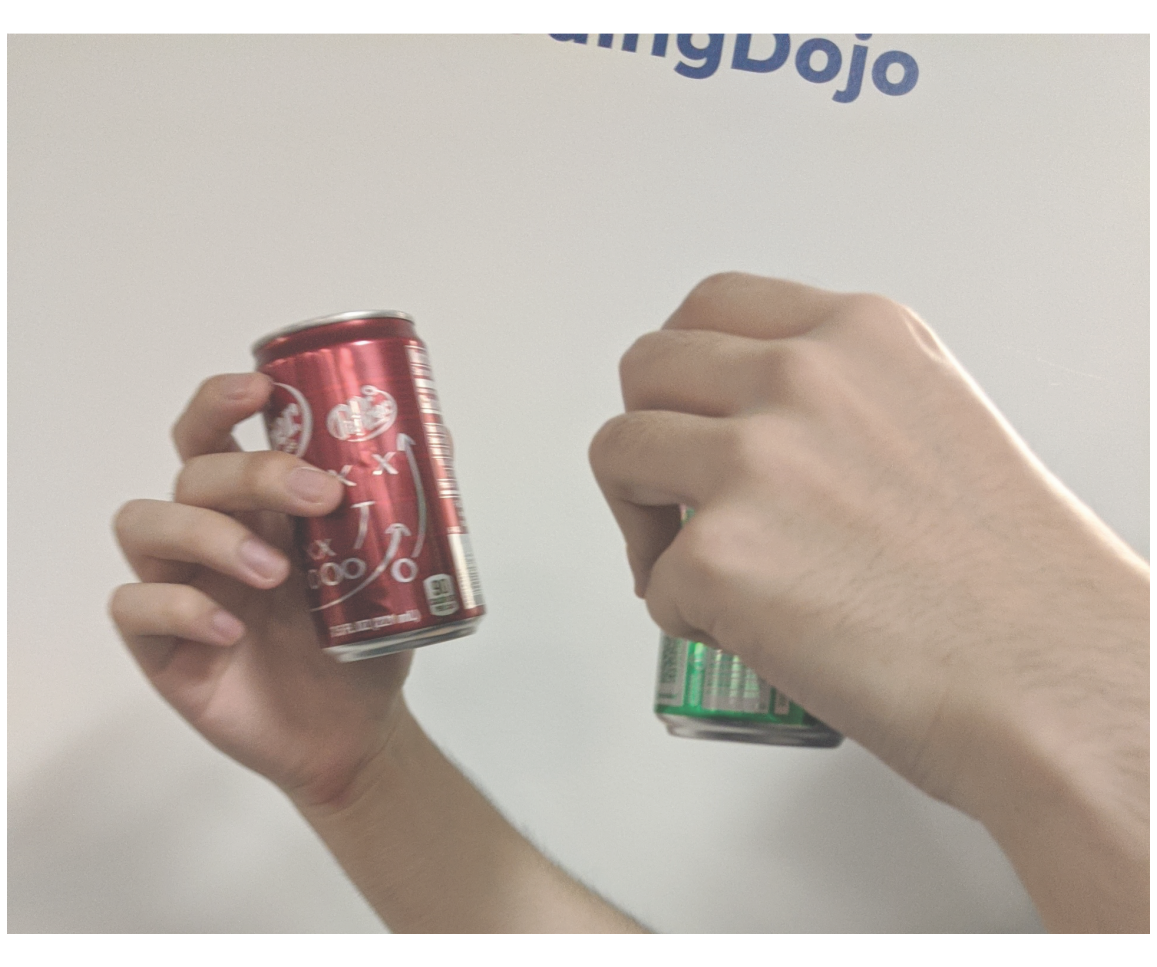
E. Setting up the virtual environment

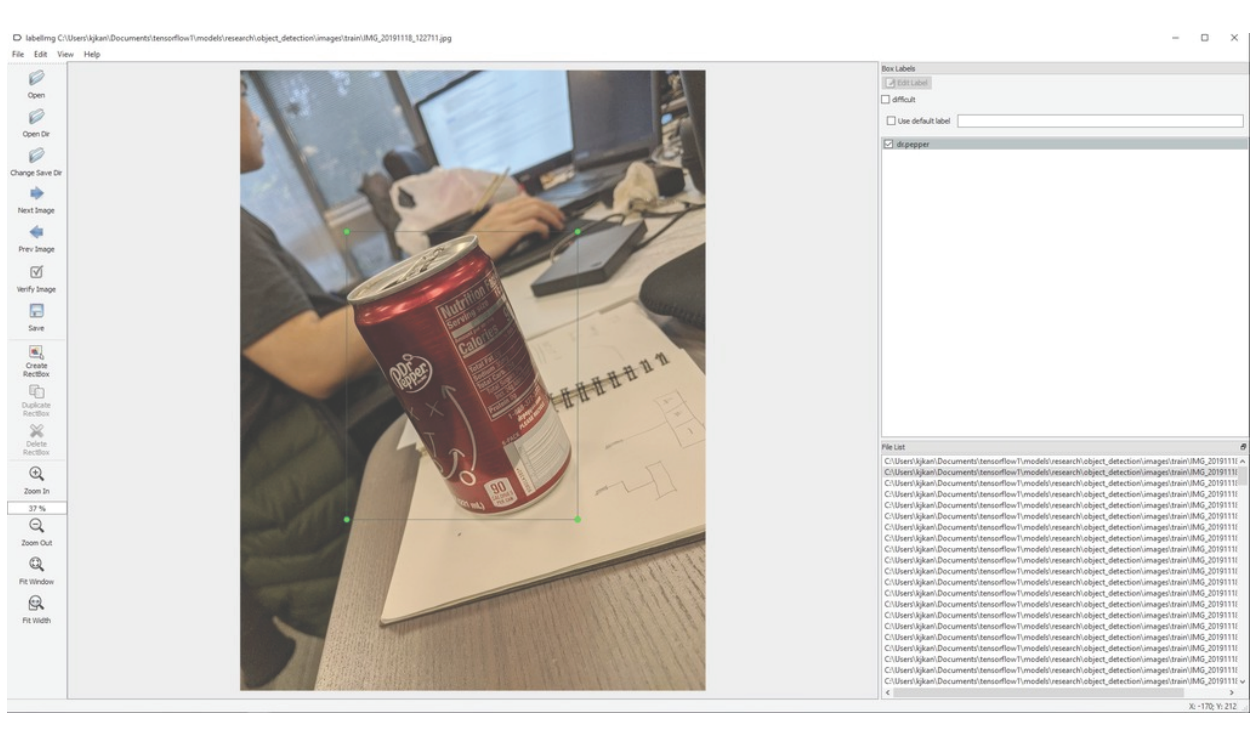
As a good practice, we started a <tensorflow1 > virtual environment for all 98 modules we used for this project, some were necessary for final product, some were for testing and experimentation.

*For the full setup guide, please refer to the video by "Edje Electronics" on youtube.
Source: https://www.youtube.com/watch?v=Rgpfk6eYxJA&t=1782s

F. Obtaining data

Perhaps the most important part of this project. At first we tried to use google images as our data source for pictures of cans, however, it turned out that many images online were not "original", meaning some images might be originally saved as PNG, but someone resaved the image as JPG and uploaded it. It caused the images to maintain a 4-channel RGB which was unreadable by the Object Detection API we were using. So we had to go around and take our own pictures, totaling 507 images of different cans in different backgrounds.



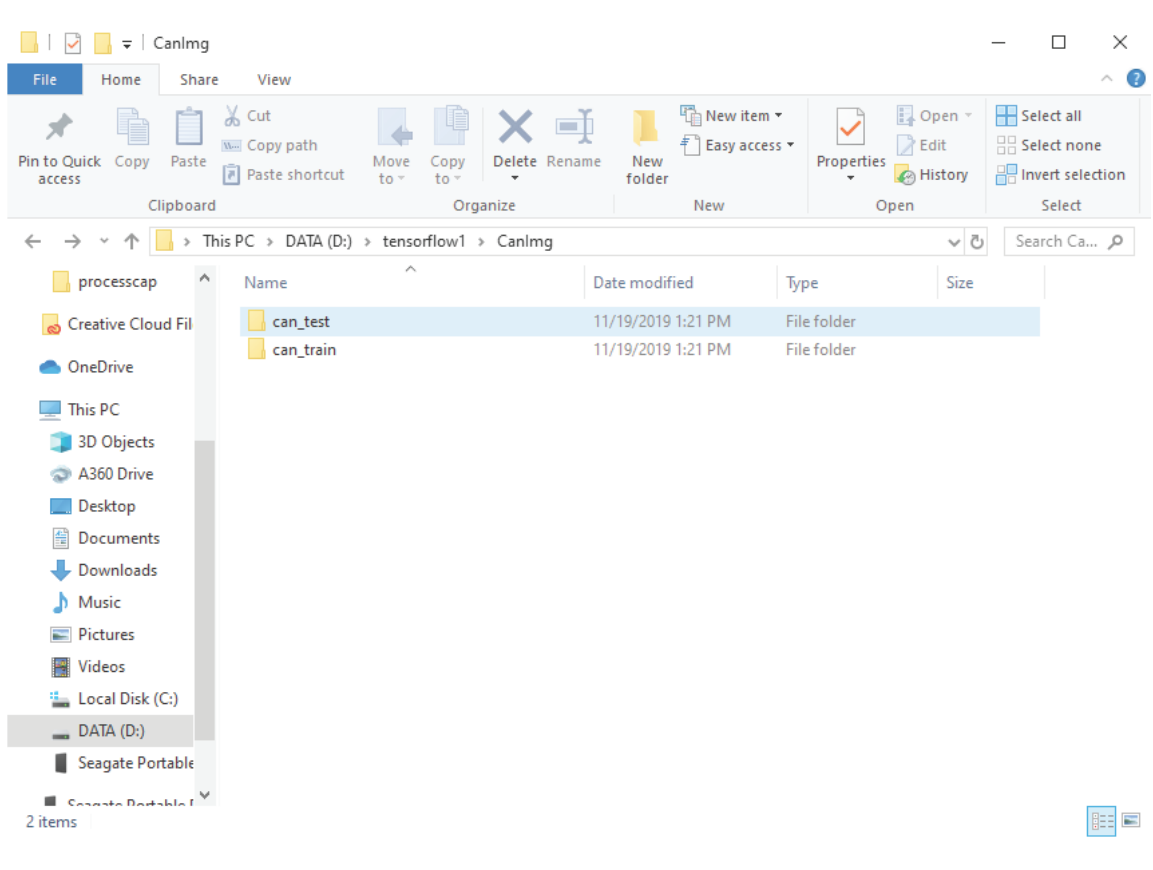


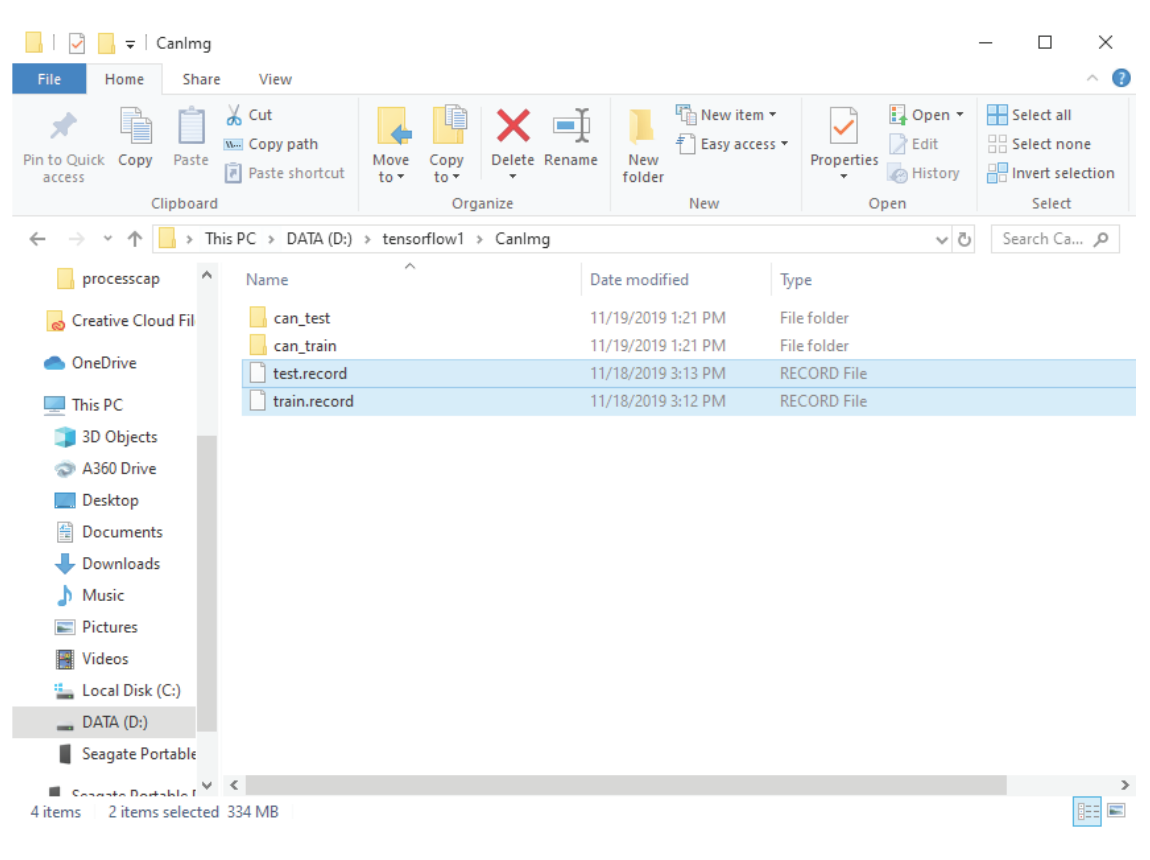
G. Processing data

In order for the model to start learning, we need to first label the training images and point out the cans for the algorithm to understand what consitutes a "can". This process was rather daunting and repetitive, but necessary. And later we discovered importance of this task and had to relabel all the images and retrain the model for better accuracy.

H. Split into training or testing datasets

It's not only important for the model to learn what a "can" is, it's also important for us to test its knowledge. As we have learned in various tutorials and forum responses, a good practice is to split the dataset in a 80/20 ratio - 80% for training, 20% for testing. There is another thing, we have to carefully pick which pictures to go into the testing so that the model doesn't just learn the pattern or order of which we feed the picture, but actually learning what a "can" is, as we wanted.



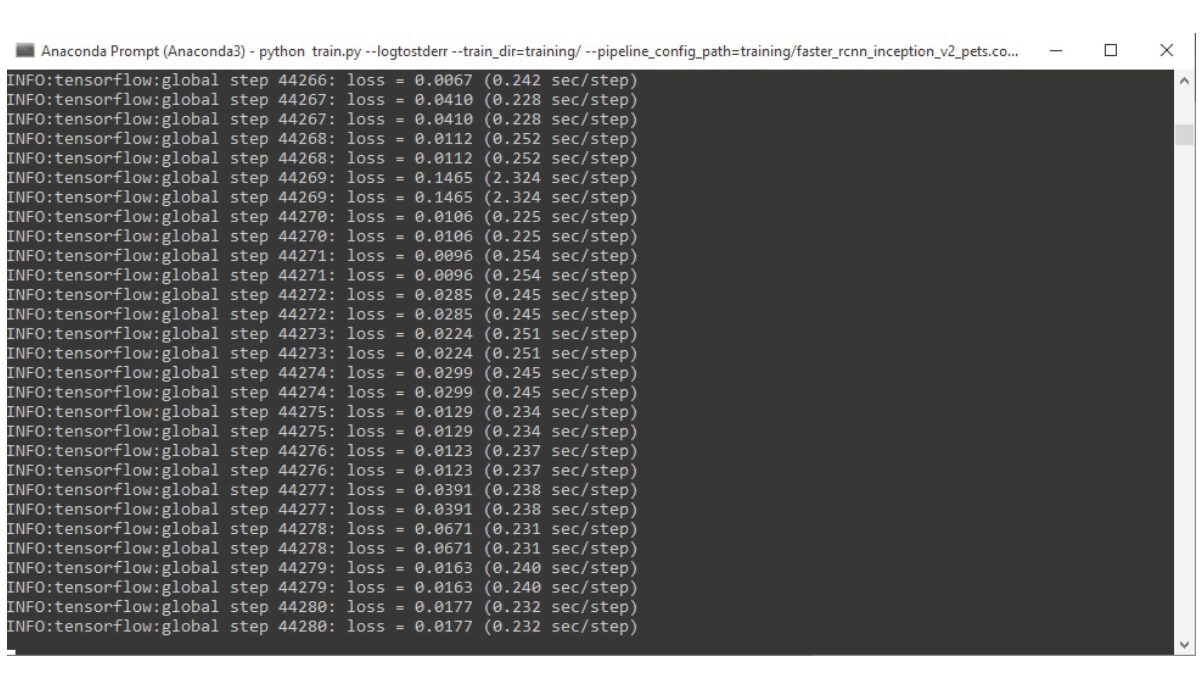


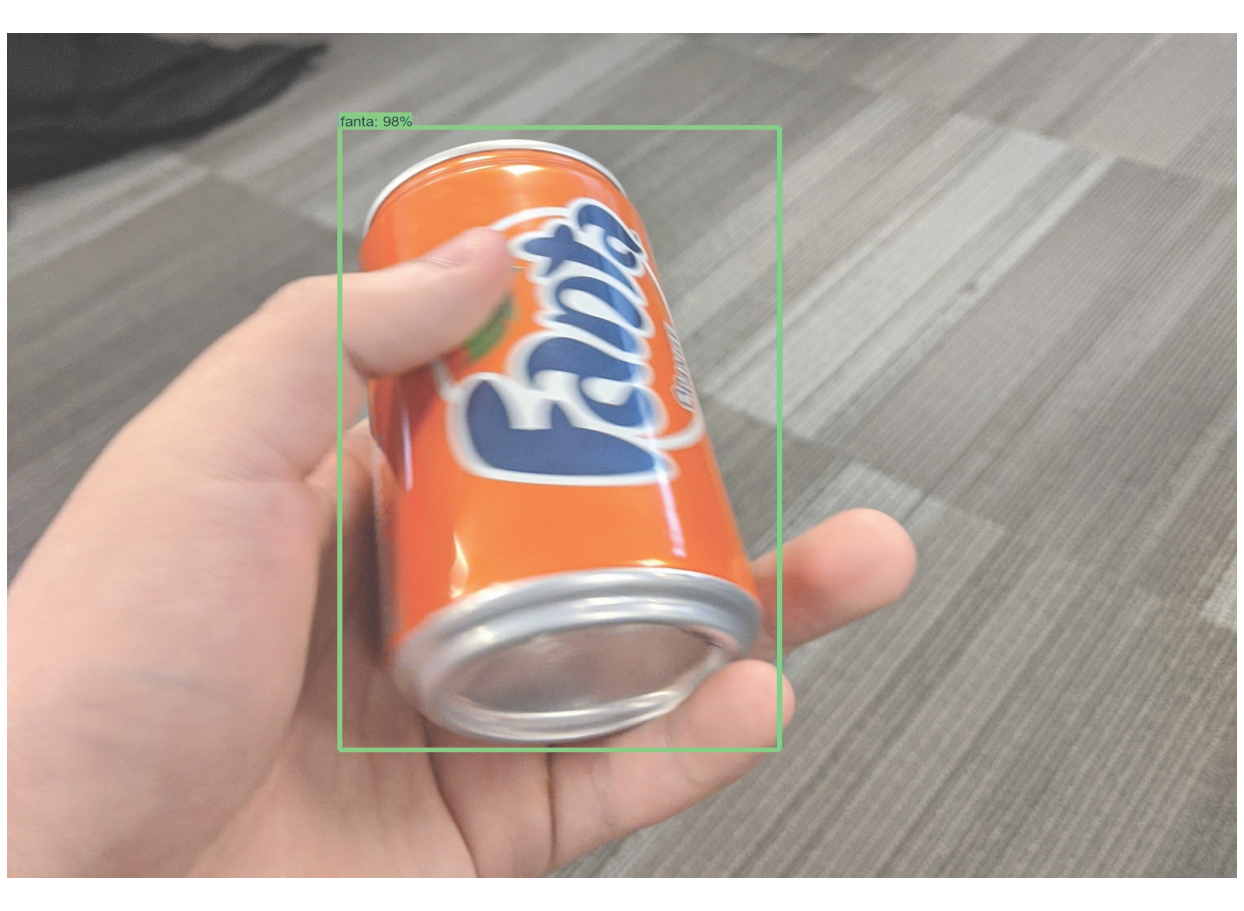
I. Generating necessary documents

After we have the input data ready, we need to convert the xml files from image labeling into excel sheets, which contains the coordinates of the four corners of the boxes we annotated on the images.

J. Training the model

Once the datasets and related files are ready, the training itself is quite easy thanks to the Tensorflow team. We only need to change a few directories and number of class we want to detect, we are done. The rest is to just wait and see the results. If the accuracy is not satisfatory, we have to alter the input data and retrain the model.



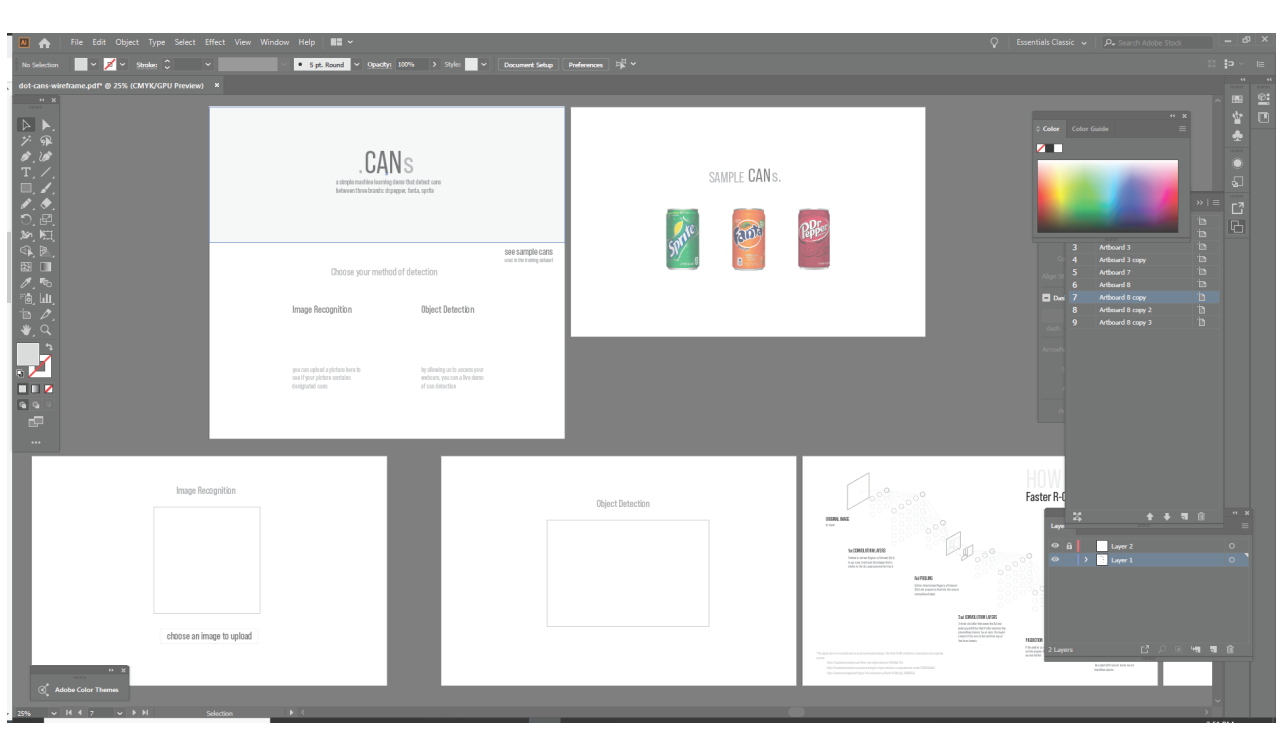


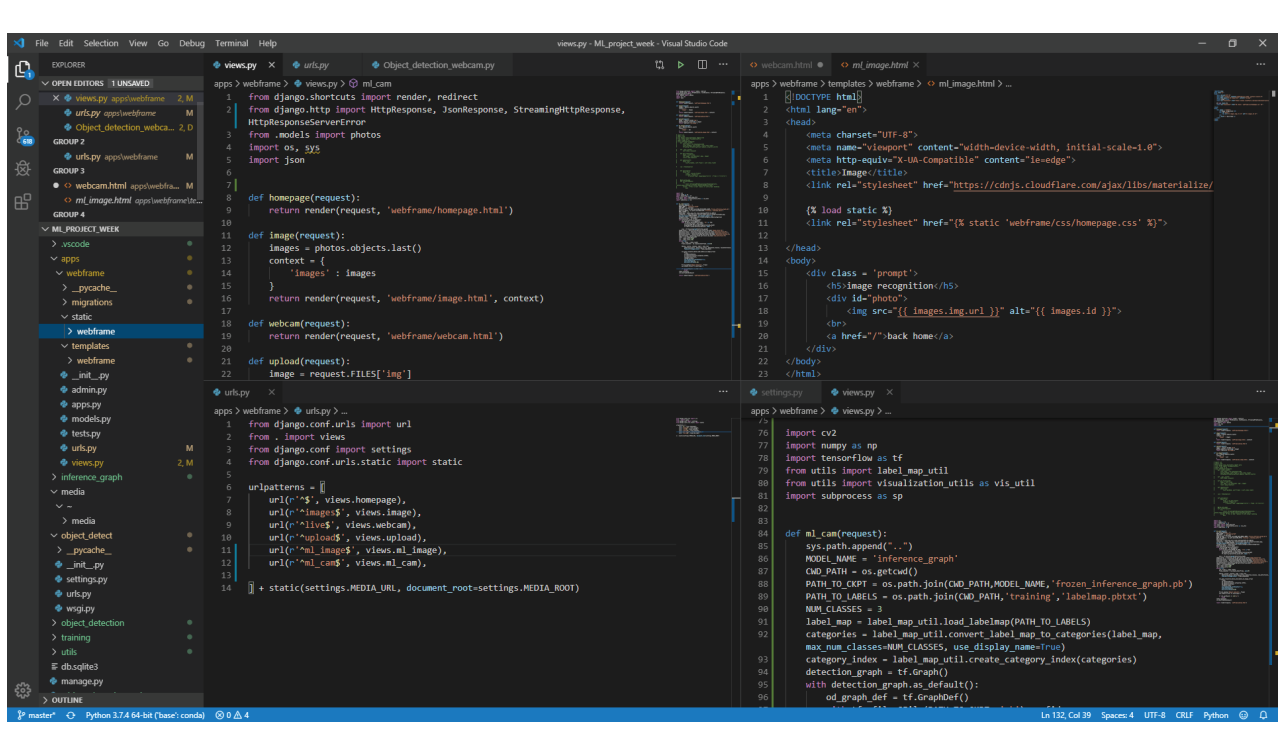
K. Seeing the result

We are always perturbed when it comes to "the moment of truth" --- if the accuracy isn't satisfactory, we have to spend a few more hours to train another one. The Tensorboard ML visualization helped a great deal.

L. Designing the website

With model trained, it's now time to move onto the next stage, where we integrate ML onto a functioning website using Django web framework. First we designed and planned the wireframe of our website in Adobe Illustrator.



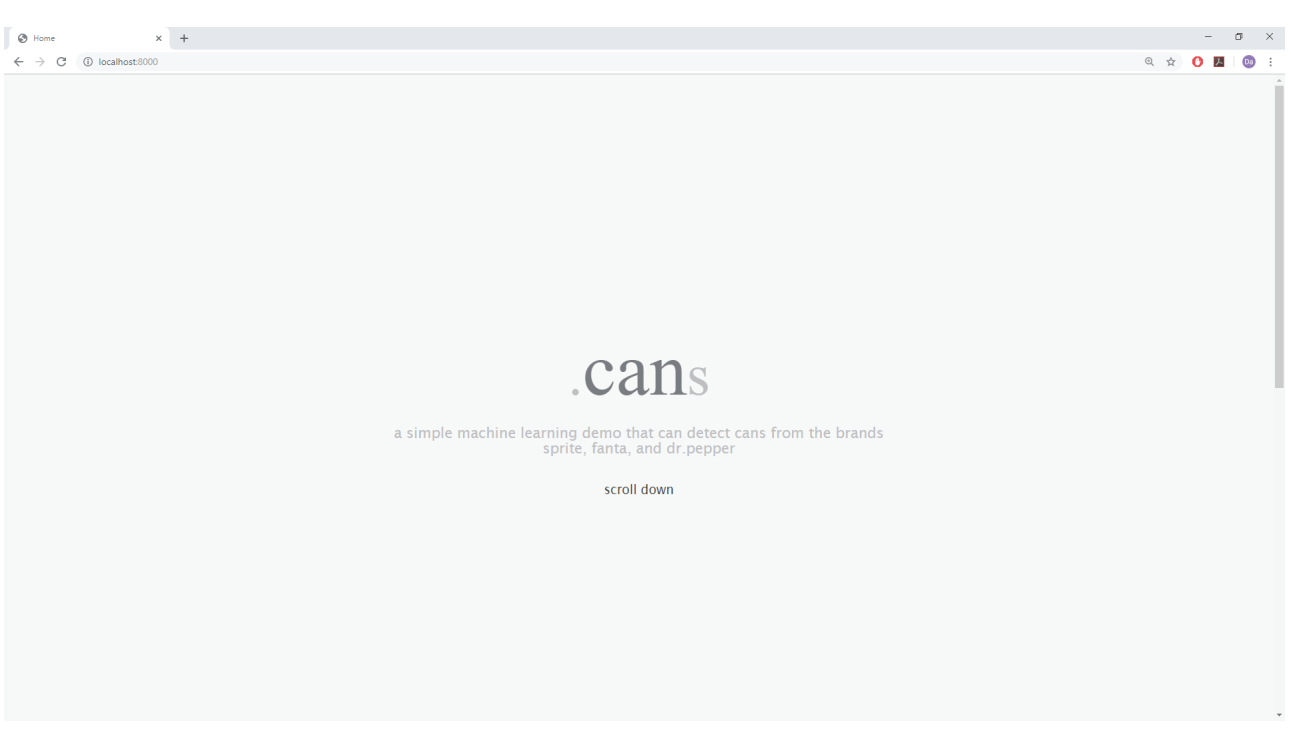


M. Building and testing the website

With frontend coded, it's time to move our trained ML model into the backend, where it can detect and classify if there is a 7oz soda can on the image a user uploads.

N. Finish

After everything seemed to work properly, we added more information on the website like how faster R-CNN architecture works, what cans did we use to train the model and our process. We believe a proper documentation of a project will be helpful not only to ourselves, but to anyone who visit the site and be inspired to start their own project.



The Reflection 03

What we did well:

Persistence

Like most things in life, the start of something is always hard, especially when you have a very limited time to learn something completely new and produce accept able result. We persisted through countless frustrations and never lost morale. We are proud of ourselves.

Adaptation

Many guides we found online were made years back. We didn't tunnel vision into one possible solution, and we learned a great deal by tweaking, altering and updating these snippets.

What we could do better:

Data Quality

The quality of our data can be improved tremendously. The images we current have don't cover enough scenari-os effectly, expecially regarding Fanta cans, as a result, the model often mistaken Fanta cans as Sprite Cans.

Live Detection

~~After spending the last two days bashing our heads against the wall trying to make it work, we eventually had to stop because of the time limit. Currently, there is a lack of information on the internet regarding porting openCV images to Django framework. However, the struggle has made us see where we were lacking in web development and how we could improve in the future.~~

Added and Fixed by using StreamingHttpResponse.