**Udacity Self-Driving Car Engineer**

**Report**

* **Project:** Traffic Sign Classifier
* **Name:**  Shao Hongxu

**Let me just use some brief sentence to explain the thing I did, basically these are 3 big parts here:**

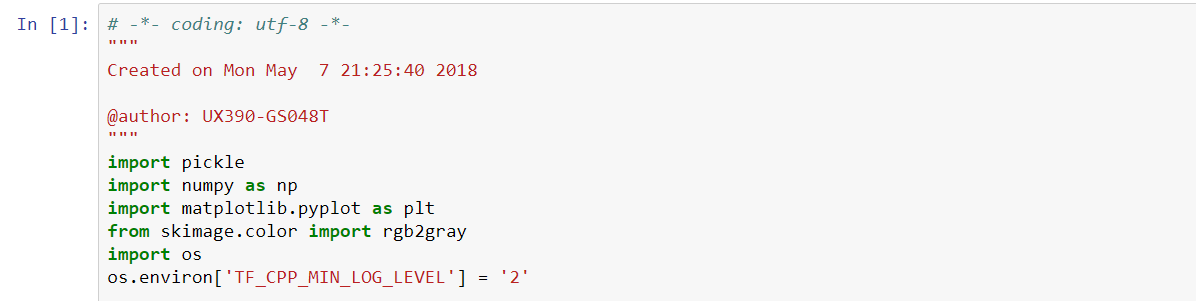
**Part 1:** import necessary lib

**Part 2:** identify Main function

**Part 3:** calculate model prediction accuracy

**To be specific as below, you will find some note on Script, key take away and Area could be improved:**

**Part 1:** import necessary lib:



**Part 2:** identify Main function:



To archive the goal of deep learning Model, below is what I have done under Main function:

**2. 1:** Import data of training, validation, testing set. Original data is within ‘.p’ format, here we need to pull them and save into list format for further usage:





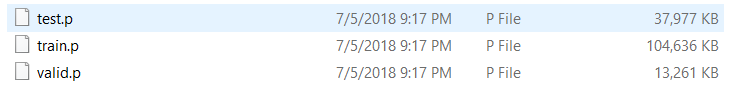
**-> Key take away:** while pulling original image data, it is necessary to convert colour image into grey image before using it for deep learning model. It will reduce the dimensions of input data. In the end, the benefit would be faster training time.

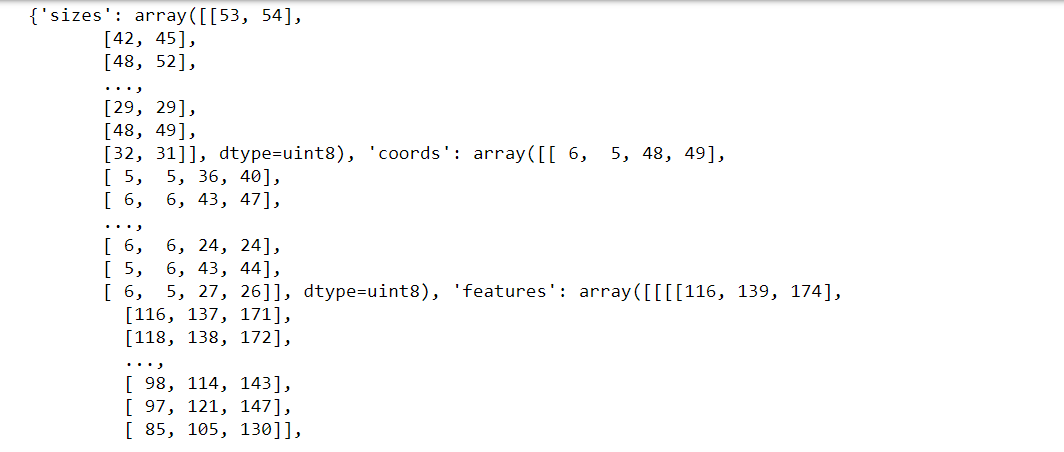
**-> Area could be improved:** not all the provided used in this code in the end, for those data not being used, related code could be commented to save computing Memory.

More importantly, if we could enlarge original data, it would be very helpful, like project spec recommended, rotation, translation, zoom, flips, and/or colour perturbation etc. will decide later whether to do it or not depends on the time & accuracy.

**-> Result:**

*Original data:*





*Data after pulling into python:*



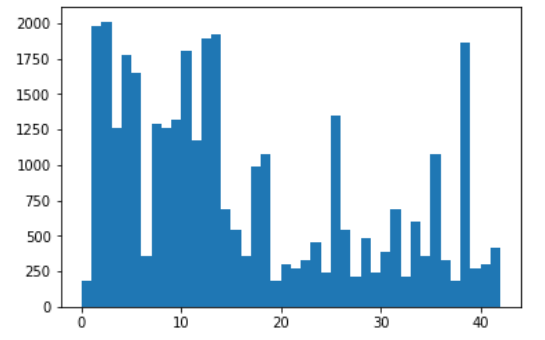
**2. 2:** Visualize the training data – image label



**-> Result:**

*Summarise the training label, realized in total there are 42 different label sets, so that 43 output classifiers needed for the classification model:*





**2. 3:** choose some random images and list down shape and min/max value from image data:



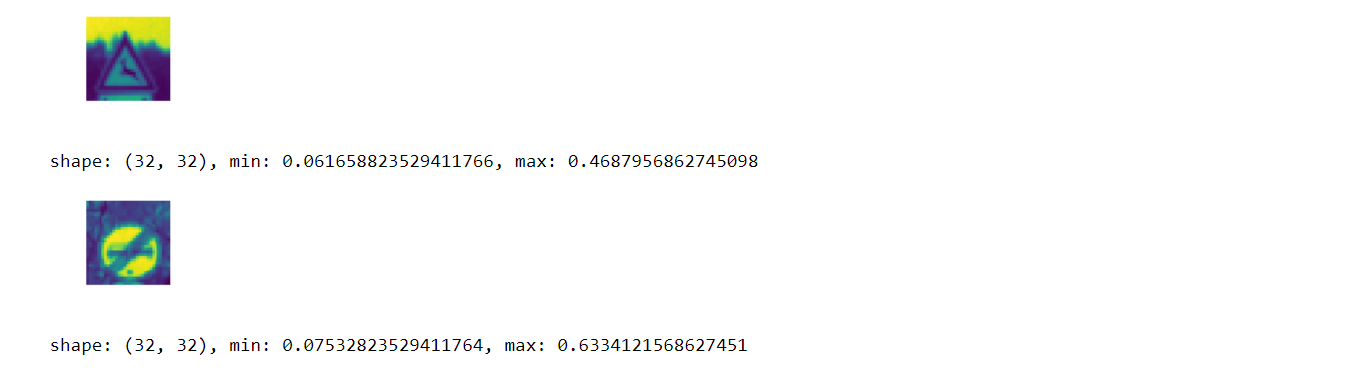
**-> Key take away:** as mentioned from project Spec, all images data provided have already been converted into 32\*32 shape.

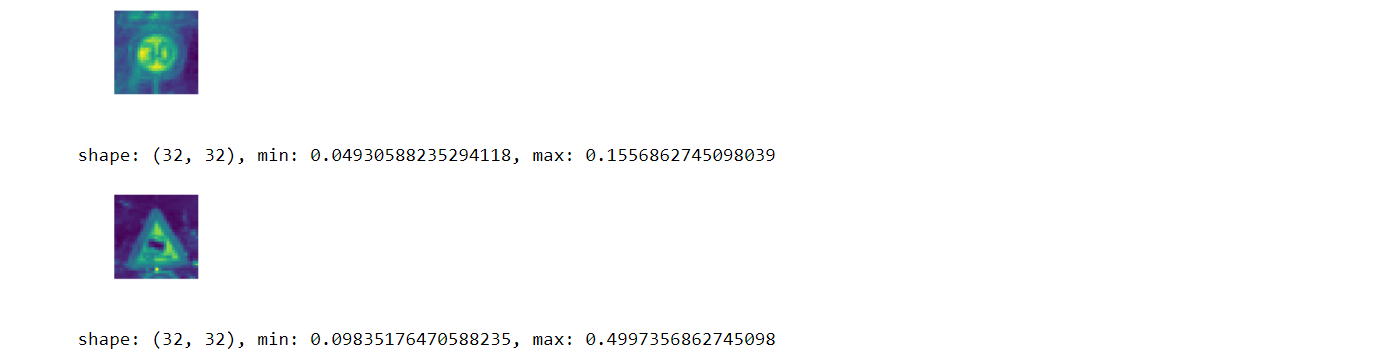
Still I just put a checking point here in case any other similar model has different shape of images, we can have a basic sense of how various the images’ shape are.

**-> Area could be improved:** here we can also draw a chart to show how many image there are in total etc.. Just from better understanding data perspective.

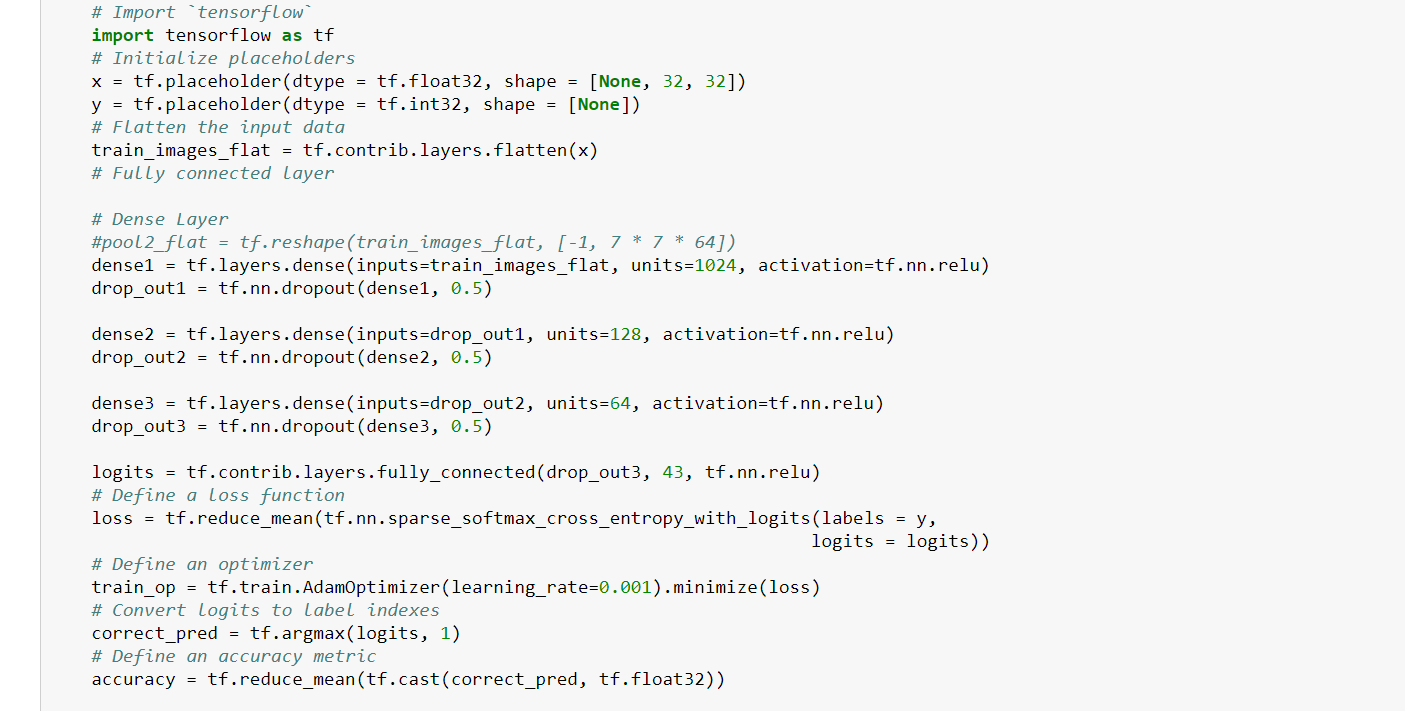
**-> Result:**

*4 sample Images:*





**2. 4:** to initiate a training model using TensorFlow:



**-> Key take away:**

This is the most important area apparently, architect here obviously will impact the final accuracy in the end. Let’s look at the model overview:

* Input layer: 32\*32
* 1st layer: flatten layer
* 2nd layer: 1024 units, with 0.5 dropout
* 3rd layer: 128 unit, with 0.5 dropout
* 4th layer: 64 unit, with 0.5 dropout
* Output layer: 43

From 32\*32, we use 5 layer levels to reduce level units into 43 classifiers in the end.

**-> Area could be improved:** here is the key area can be improved, as all other machine learning/ deep learning model, a bad model will not come out with any high accuracy prediction rate:

- how many levels of layer?

- how many units for each level of layer?

- dropout rate? here to be 0.5

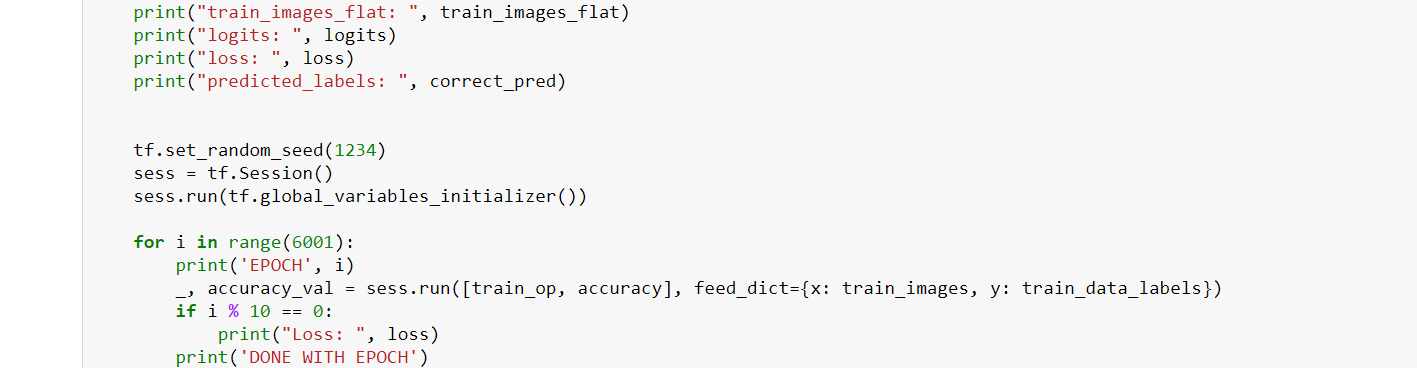
- etc.

This part is the core of Machine learning/ deep learning. Also GPU, TPU play an important role of training out of the better model accuracy.

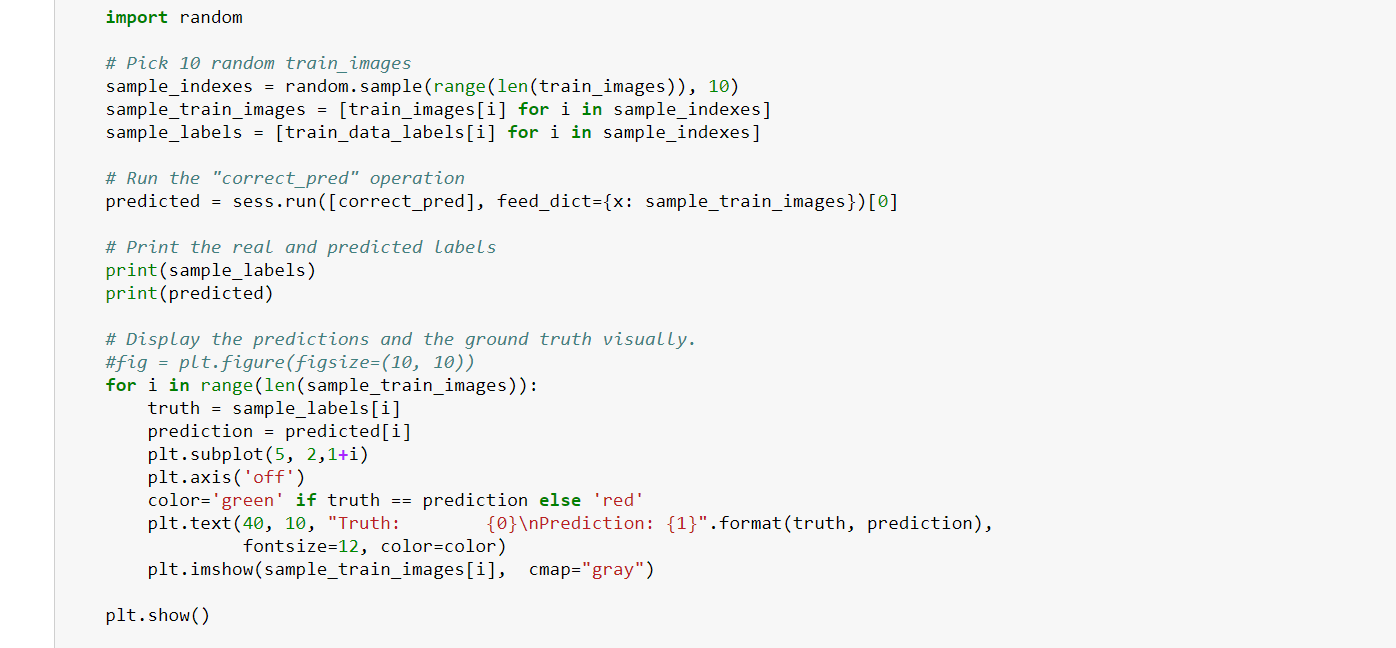
First time I was using laptop with 16GB CPU (no NAVIA GPU), 98% CPU usage stopped me doing that again since it may destroy the machine.

Second time I was using my desktop with NAVIA GPU 1080. It takes only 5min to complete 6000 echoes. And with not bad accuracy in the end, so the model architect mentioned above can work to a certain extend.

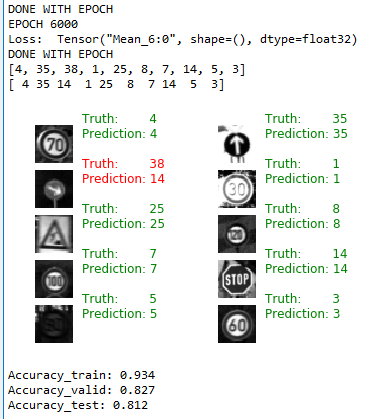
**2. 5:** Initiate model weight variables, and to set up run Epoch into 6000 times.



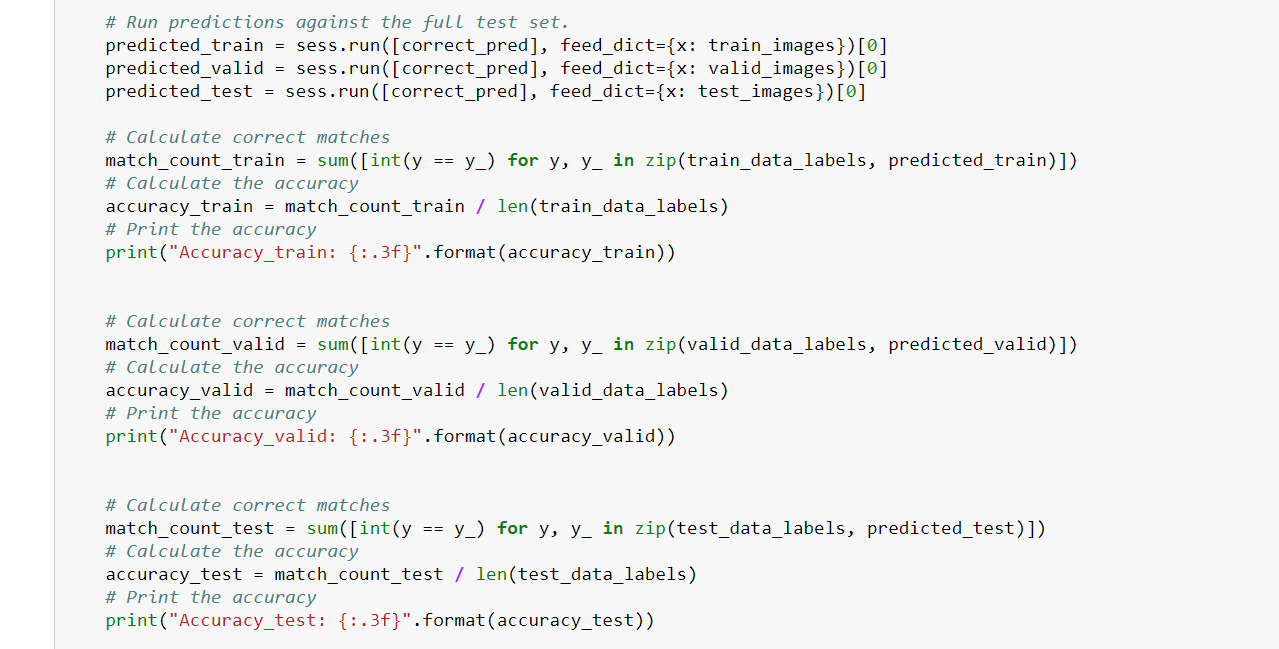
**2. 6:** once model training completed, to show the prediction of 10 random images



**-> Result:**

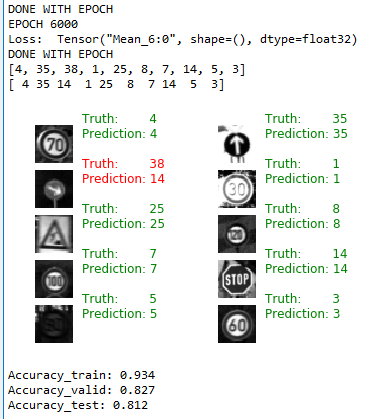


**Part 3:** calculate model prediction accuracy:



Result:

After 6000 echoes:



After 12000 echoes:



**Summary:**

These are the steps I could think of for project 2, as a base line. Still a lot of improvement needed to achieve a better accuracy.

The key area could be improved is:

1. Original data expansion

if we could enlarge original data, it would be very helpful, like project spec recommended, rotation, translation, zoom, flips, and/or colour perturbation etc.

1. Model architecture enhancement

- how many levels of layer?

- how many units for each level of layer?

- dropout rate? here to be 0.5

- etc.

1. After 12000 echoes, my computer GPU temperature become 81, should couldn’t try with more echoes with model. Think will think about the way to enhance model instead of running more echoes. Now training accuracy could reached 93%, but validation set still below 90% (target is 93%), will keep thinking about the way to improve and update here whenever there is a new idea!!