**Traffic Sign Recognition Classifier**

The goals / steps of this project are the following:

* Load the data set
* Explore, summarize and visualize the data set
* Design, train and test a model architecture
* Use the model to make predictions on new images
* Analyze the softmax probabilities of the new images

**Step 1:**

After loading the dataset, the first step was to calculate the summary statistics of the traffic signs data set. I used python’s len() function to the calculate the number of training, validation, and test data. To calculate number of unique classes, I used python’s set() function, which returns unique elements (duplicate elements are not allowed). I also calculated the image shape.

|  |  |
| --- | --- |
| Number of training examples | 34799 |
| Number of validation examples | 4410 |
| Number of test examples | 12630 |
| Image data shape | (32, 32, 3) |
| Number of classes | 43 |

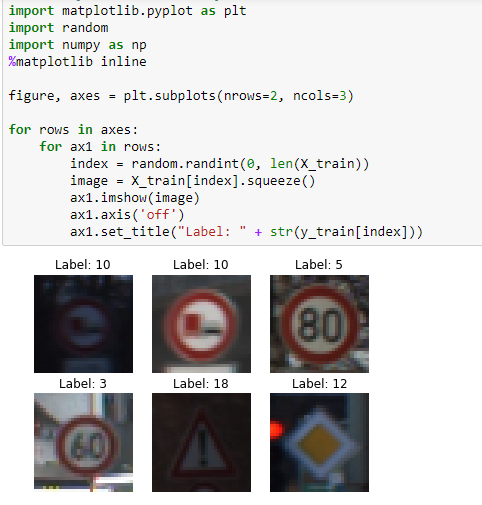






**Step 2:**

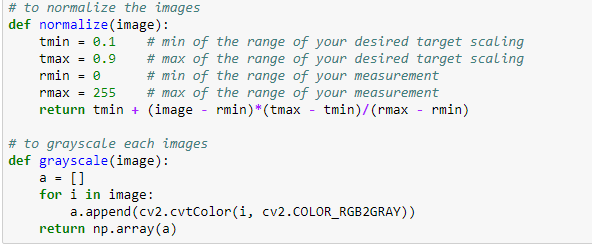
For visualization, I simply used matplotlib to plot random 6 training images and printing their labels (nothing fancy).

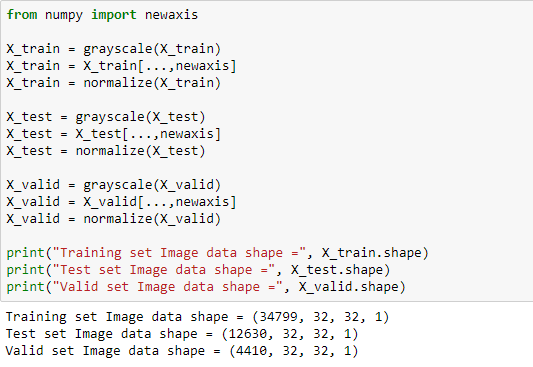


**Step 3:**

Preprocessing of the images, I tried – just normalizing (converting images to float and dividing by 255), and grayscaling images and normalizing (normalizing images within a particular range).

In the end, I chose the latter. I converted the RGB images to GRAYSCALE images. Used numpy’s newaxis function to increase the dimension of the image after the conversion. (After converting the images to grayscale, the shape of the image came out to be (32, 32), so I used newaxis to convert to (32, 32, 1)). Then normalized the grayscaled images to a particular range (0.1 – 0.9).





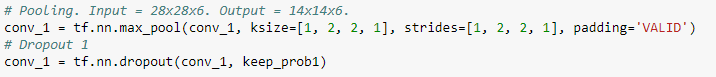
**Step 4:**

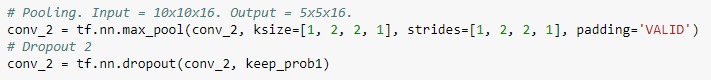
I used the LeNet architecture which was used for MNIST (LeNet Lab in TensorFlow). I added **dropout** after each max\_pooling layers and after 1st fully\_connected layer. Created placeholders for keep\_prob. Set the number of EPOCHS to 70 and BATCH\_SIZE to 128.

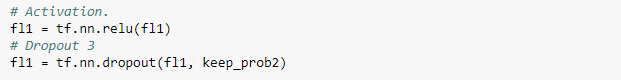




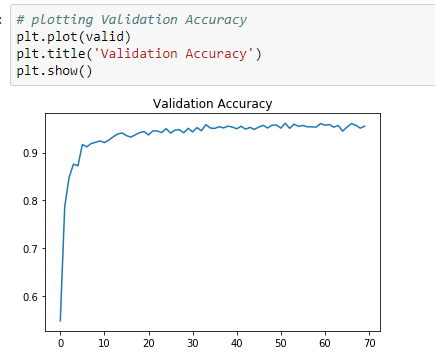
For the validation accuracy, I kept the dropout rate to 1, and for training, 0.8 and 0.7. I had to experiment on the dropout rates as to keep the validation accuracy about 0.93.



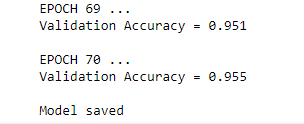




I created a list and appended all the validation accuracy in order to create a plot for Validation Accuracy.



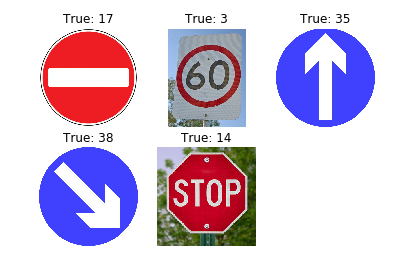
By adjusting the parameters, I was able to obtain a validation accuracy of 0.955 after 70 epochs.

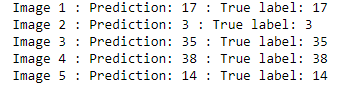




**Step 5:**

For testing new images, I collected 5 random test images, resized to (32, 32) for the model, preprocessed, and predicted the labels. I was able to get all the correct labels.







**Step 6:**

To analyze the softmax probabilities for each image, I used **tf.nn.top\_k** setting **k=5** to get top 5 probabilities.

