Traffic Sign Recognition

Project #2

Self-Driving Car Nanodegree - UDACITY

Step 0: Load the data

At the 0^{th} step I loaded the train.p and test.p files. Because there is not any validation data set, I used train_test_split from sklearn library to split train data set to validation and train dataset. Split factor which I used was 20%.

Step1: Data set summary and exploration

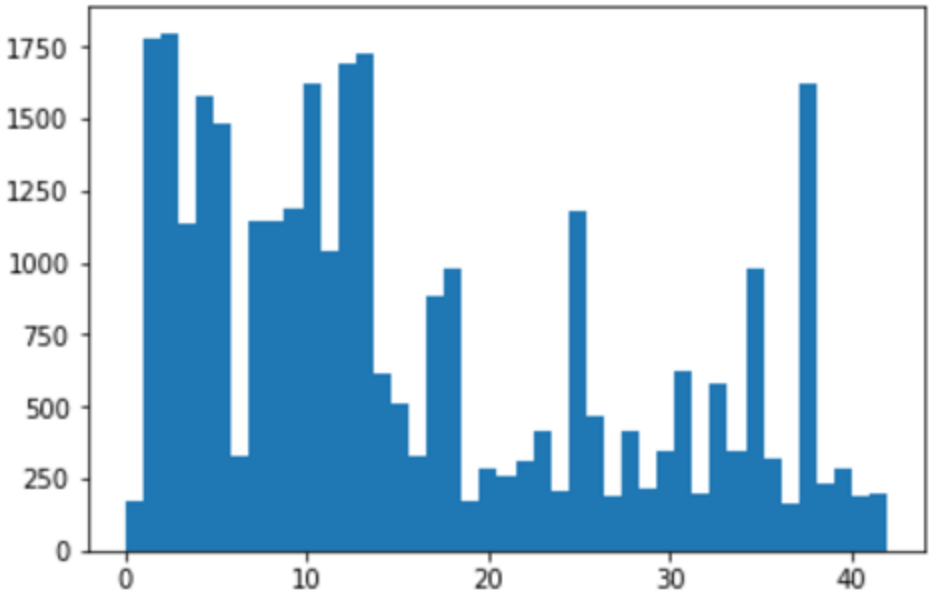
I used pandas library to load signnames.csv file and see the traffic signs description.

Here are some features of loaded data sets:

```
Number of training examples = 31367
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43
```

Then I plotted some randome traffic signs as follow:

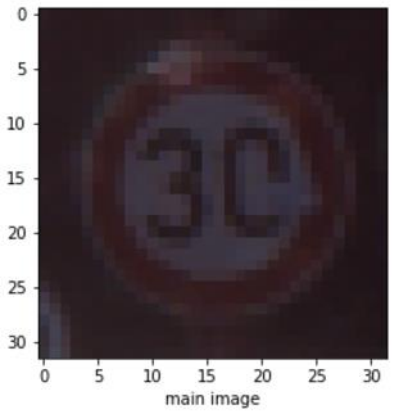
I also plotted histogram of classes distribution:

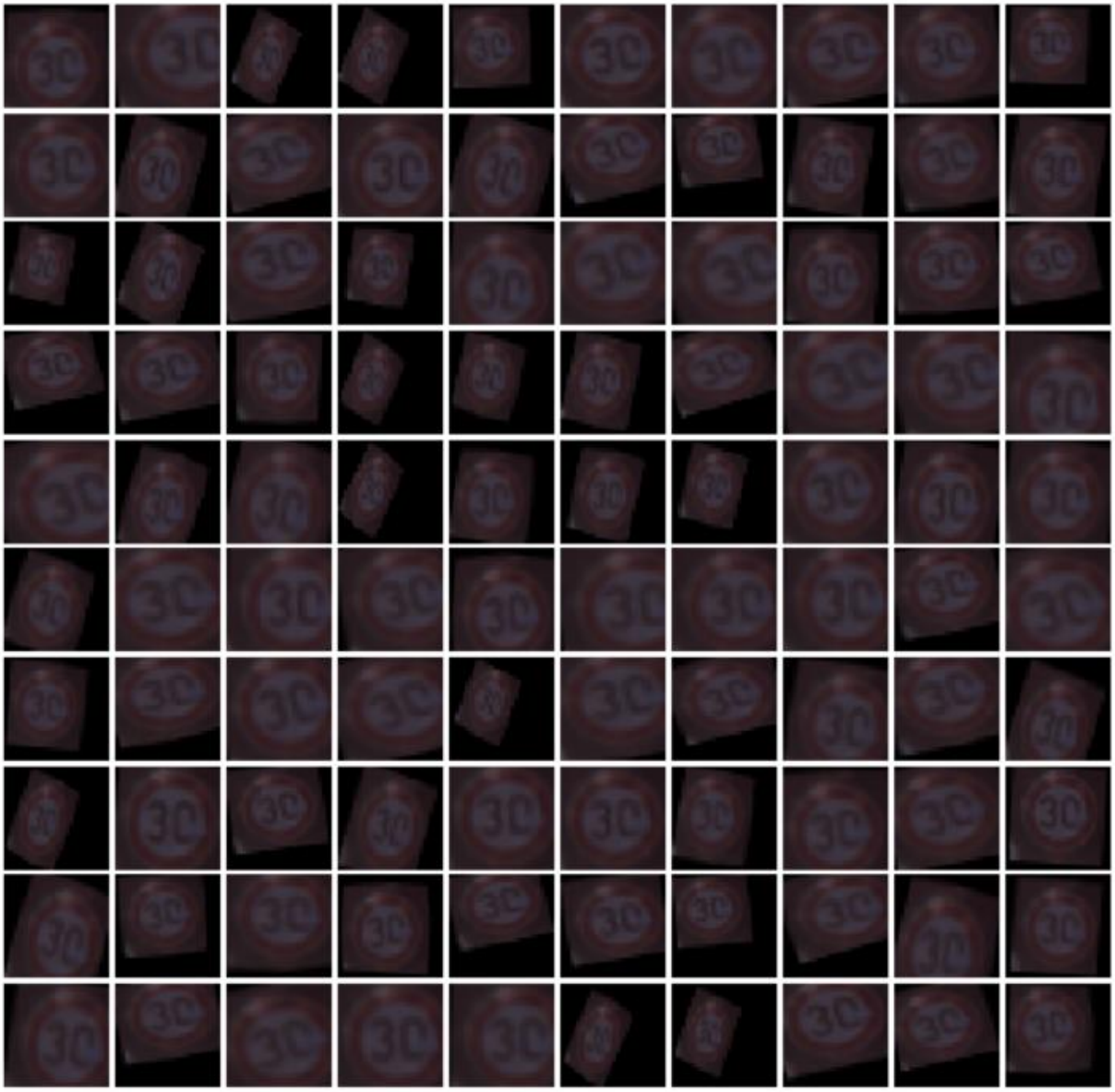


Step2: Design and Test a Model Architecture

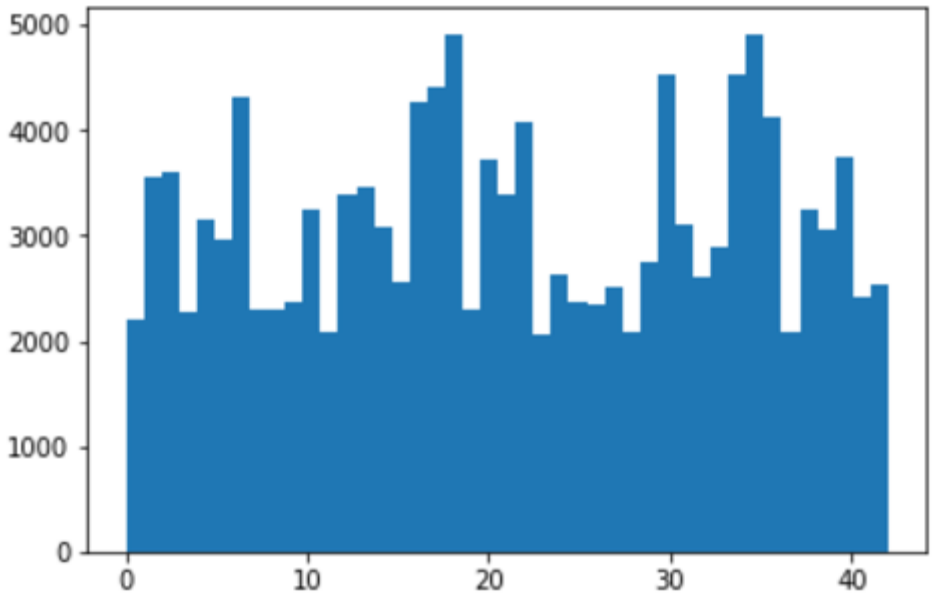
Pre-process the Data Set (normalization, grayscale, etc.)

You can see that number of images in some classes is fewer than others and this will affect the performance of network. I did some data augmentation such as rotation, translation, shear and brightness to generate more images. We determined a treshold of 2000 images per class and used data augmentation function. Results for one image are as follow:





After applying data augmentation on all train images, the histogram of new data set is as follow:

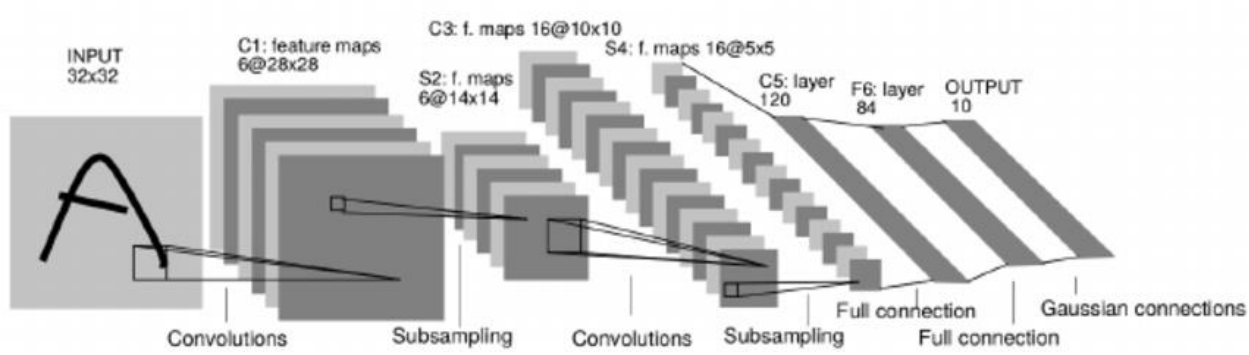


The balance of number of images per class now is better than previous state.

Then I used some preprocessing techniques such as grayscaling(because I think color is not important for this task), local contrast maximization(to improve contrast of images) and normalization on images. Here are some result images:

Model architecture

I used tf.contrib.slim to design model architecture. I implemented LeNet architecture with tuned hyperparameters for this problem as follow:



Layer	Description
input	32x32x1 grayscale image
Convolution 5x5	Stride1, 20 outputs,kernel 5x5, same padding, relu activation
Max_pool2d	Stride2, kernel 2x2, valid padding
Convolution 5x5	Stride1, 50 outputs,kernel 5x5, same padding, relu activation
Max_pool2d	Stride2, kernel 2x2, valid padding
dropout	Keep_prob = 0.5
Flatten	
Fully connected	500 nodes, relu activation
Fully connected	43 nodes, no activation function

I used dropout with keep_prob=0.5 to prevent overfitting in the network.

Then I trained the network for 100 epochs and used bach size of 128 and learning rate of 0.001.

I trained the network on lenovo Y700 with NVIDIA GTX980 GPU.

Final results are as follow:

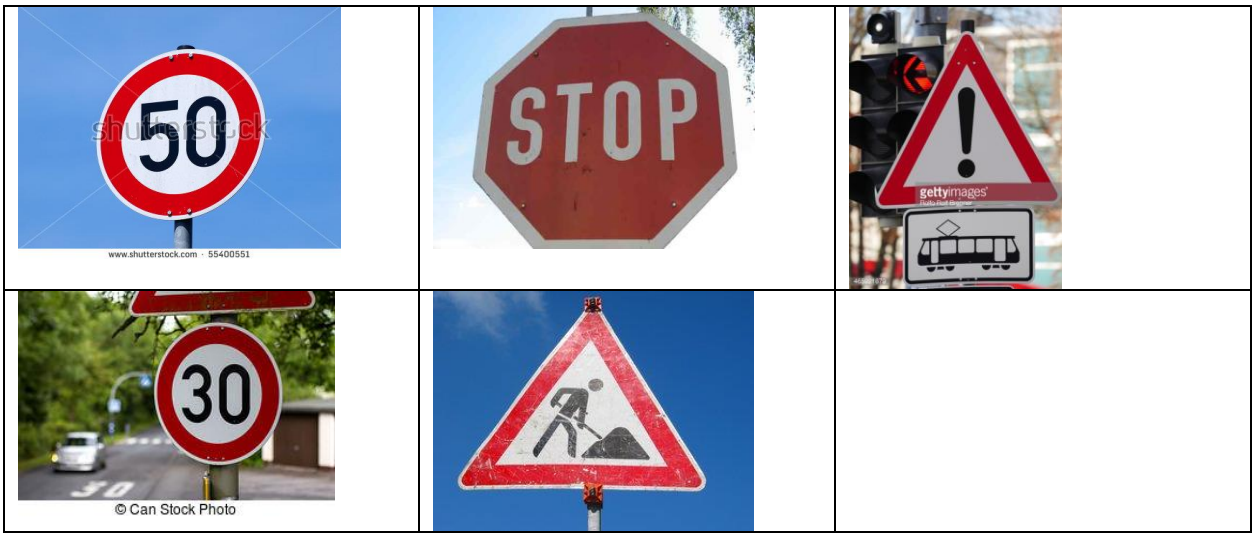
```
EPOCH 90 ...
Validation Accuracy = 0.993

EPOCH 91 ...
Validation Accuracy = 0.991

EPOCH 92 ...
Validation Accuracy = 0.992

EPOCH 93 ...
Validation Accuracy = 0.990

EPOCH 94 ...
Validation Accuracy = 0.992
```

```
EPOCH 95 ...
Validation Accuracy = 0.990

EPOCH 96 ...
Validation Accuracy = 0.993

EPOCH 97 ...
Validation Accuracy = 0.991

EPOCH 98 ...
Validation Accuracy = 0.992

EPOCH 99 ...
Validation Accuracy = 0.993

EPOCH 100 ...
Validation Accuracy = 0.991
```

The validation accuracy is mor than 99 percent. The test accuracy is 94.5%.

Step3: Test model on new images

I downloaded 5 images and resized them to (32,32,3) shape.



Then preprocessed them as same as original images. Then feeded them to network. I also built a label vector based on signnames.csv file for downloaded images. Results of one run are as follow:

```
predicted labels: [ 2 14 18 38 25]
actual labels: [ 2 14 18  1 25]
```

Here are the results of the prediction:

Image	**predicted**
Speed limit (50km/h)	Speed limit (50km/h)
Stop	Stop
General caution	General caution
Speed limit (30km/h)	Keep right
Road work	Road work

Results are different at different run. Some times accuracy is 80% and sometimes 100%. In this run 4th image prediction is wrong and predicted as "Kepp right" instead of "Speed limit (30km/h)". I think It is because of position of sign in the image.

After that I used tf.nn.top_k function to print top 5 probabilities for each image:

```
top 5 prob for new imagse: TopKV2(values=array([[  1.00000000e+00,   2.274
55039e-20,   5.64435779e-21,
          8.89585291e-22,   1.22554058e-27],
       [  1.00000000e+00,   3.21243704e-16,   4.02707944e-19,
          2.63649098e-21,   1.10620104e-25],
       [  9.17005539e-01,   8.29943120e-02,   8.85464218e-08,
          1.02486037e-08,   9.81100978e-09],
       [  9.99384165e-01,   6.15860336e-04,   2.02325434e-13,
          9.21509573e-14,   5.96382413e-14],
       [  1.00000000e+00,   2.11556304e-15,   7.07986864e-17,
          2.27761648e-19,   1.29456189e-20]], dtype=float32),
indices=array([[ 2,  1, 38,  8,  0],
       [14, 36, 33, 35, 17],
       [28, 29, 18, 23, 40],
       [25, 21, 10,  2, 37],
       [25, 19, 31, 22, 21]]))
```

Probability of first predicted label for each image is about 1. The network misclassified 4th image. I think It is because of position of sign in the image. It is not at the center.