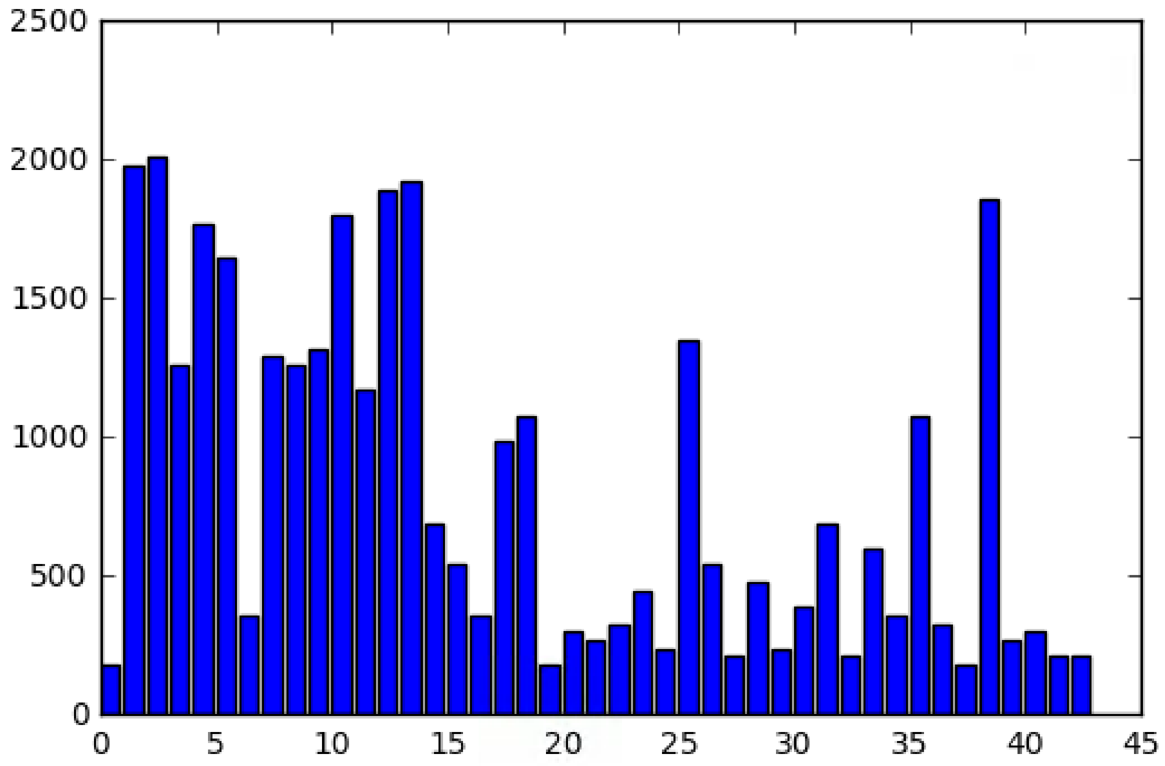
Data Set Summary and Exploration

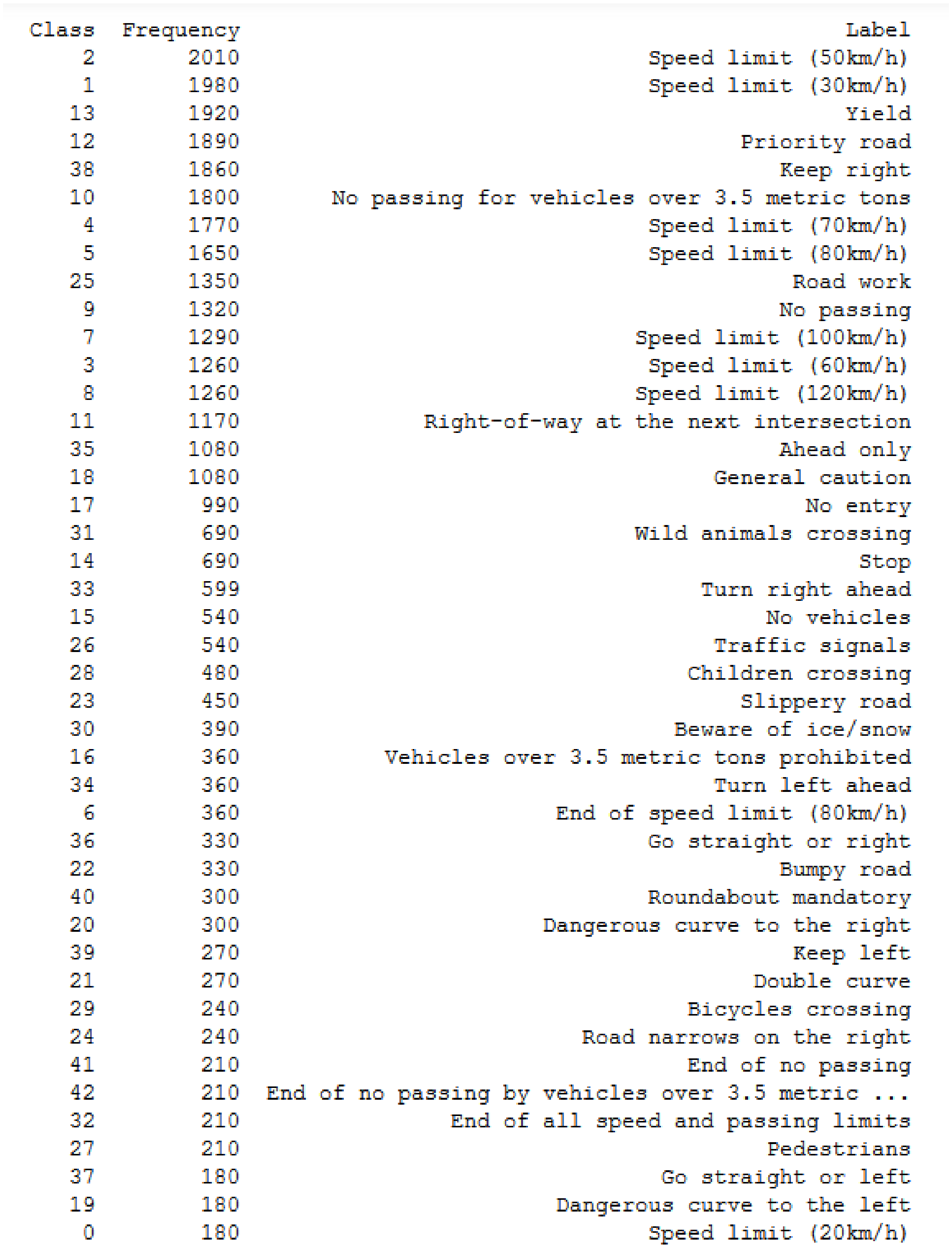
The code for these steps is contained on code cells #1 and #2 the IPython netbook. The dataset summary statistics are as below:

- Size of each image: 32 x 32 x 3 (32 pixel x 32 pixel RGB image)
- Training set size: 34799 x 32 x 32 x 3 (i.e. 34799 32 x 32 RGB images)
- Validation set size: 4410 x 32 x 32 x 3 (i.e. 4410 32 x 32 RGB images)
- Testing set shape: 12630 x 32 x 32 x 3 (i.e. 12630 32 x 32 RGB images)
- Unique number of classes in data set: 43

A random set of training images were visualized in 7x7 grid to understand the quality of data and to determine the pre-processing steps on them. The code for this is contained in code cell #3 of the IPython notebook. This resulted in identifying that images would need to be corrected brightness prior to training.

The distribution of various image classes (i.e. the histogram of number of images for a given class) was also visualized along with the values and class labels in a tabulated form. The code for this is contained in code cell #4 of the IPython notebook. This resulted in identifying that the data for all classes was not evenly distributed and the low distribution classes would need some "augmentation" prior to training





```
Class  Frequency                                          Label
    2       2010                         Speed limit (50km/h)
    1       1980                         Speed limit (30km/h)
   13       1920                                         Yield
   12       1890                                 Priority road
   38       1860                                    Keep right
   10       1800  No passing for vehicles over 3.5 metric tons
    4       1770                         Speed limit (70km/h)
    5       1650                         Speed limit (80km/h)
   25       1350                                     Road work
    9       1320                                    No passing
    7       1290                        Speed limit (100km/h)
    3       1260                         Speed limit (60km/h)
    8       1260                        Speed limit (120km/h)
   11       1170         Right-of-way at the next intersection
   35       1080                                    Ahead only
   18       1080                               General caution
   17        990                                      No entry
   31        690                         Wild animals crossing
   14        690                                          Stop
   33        599                              Turn right ahead
   15        540                                   No vehicles
   26        540                               Traffic signals
   28        480                             Children crossing
   23        450                                 Slippery road
   30        390                            Beware of ice/snow
   16        360    Vehicles over 3.5 metric tons prohibited
   34        360                               Turn left ahead
    6        360                   End of speed limit (80km/h)
   36        330                          Go straight or right
   22        330                                    Bumpy road
   40        300                          Roundabout mandatory
   20        300                 Dangerous curve to the right
   39        270                                     Keep left
   21        270                                  Double curve
   29        240                             Bicycles crossing
   24        240                     Road narrows on the right
   41        210                             End of no passing
   42        210 End of no passing by vehicles over 3.5 metric ...
   32        210             End of all speed and passing limits
   27        210                                   Pedestrians
   37        180                           Go straight or left
   19        180                   Dangerous curve to the left
    0        180                         Speed limit (20km/h)
```

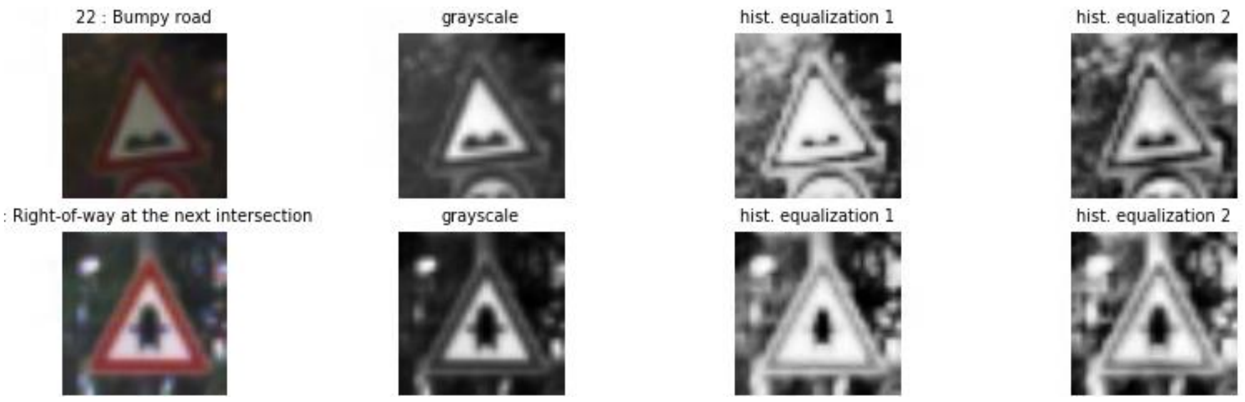
Model Architecture Design and Test

Data Pre-Processing

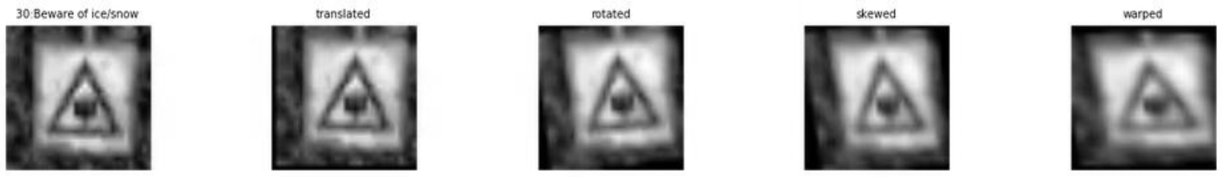
From the visualization of training images and the class distribution, the following pre-processing was performed on the image data:

- Convert to grayscale
- Adaptive Histogram Equalization using CLAHE
- Image augmentation (consisting of translation, rotation, affine and warp perspective transform) on classes identified in the earlier histogram

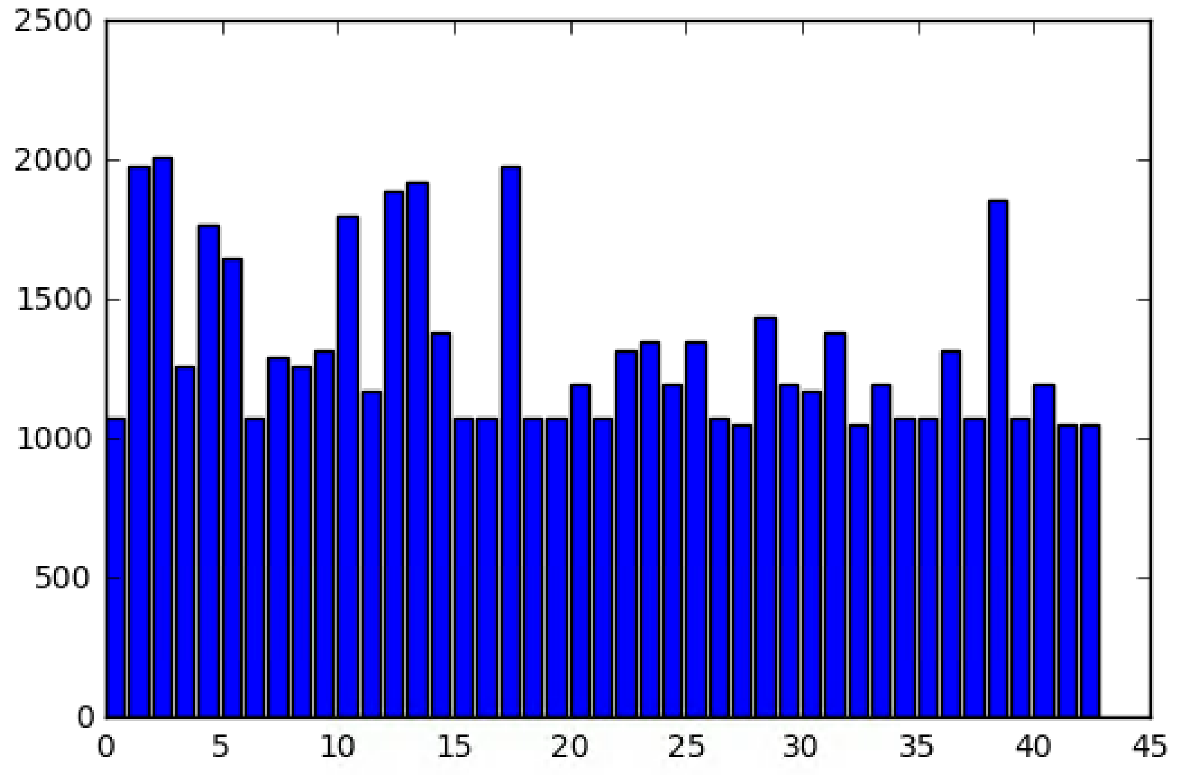
The code for grayscale conversion and histogram equalization is contained in code cell #5 of the IPython notebook. I decided to convert the images to grayscale for 2 primary reasons: it allowed me to get a LeNet model trained using the code from the labs in the lectures, the Multi-scale Conv. Neural Network for Traffic Sign classification paper by Pierre Sermanet and Yann LeCun (http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf) reports 99% accuracy using only grayscale images.
I experimented using standard histogram equalization as well as adaptive histogram equalization (CLAHE using different clip limit and grid size parameters) techniques. Analysis of randomly selected images after each transform resulted in selecting CLAHE histogram equalization with a ClipLimit of 11.0 and GridSize of 3x3 as it tended to retain the key image features more consistently. The below image shows the pre-processing sequence:



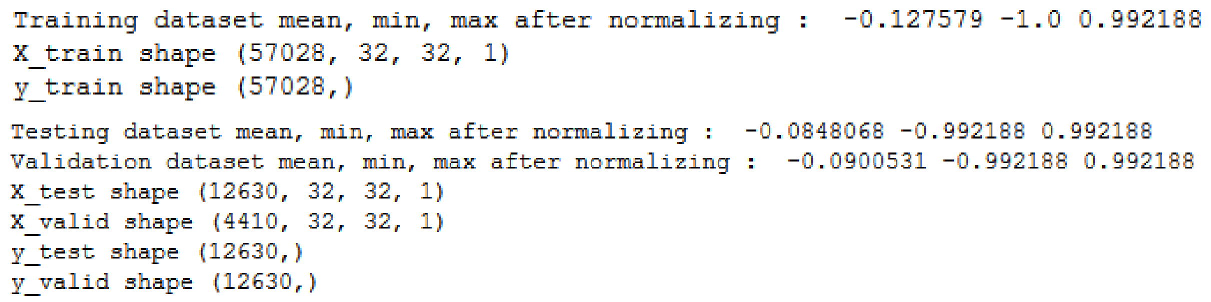
The code for image augmentation is contained in code cell #6 of the IPython notebook. In here an augmentation multiplier for each class is determined using cutoff value of 1000 minimum samples for each class. Then each image in the identified class is randomly applied a series of translation, rotation, affine and warp perspective transform using randomly selected parameters. The below image shows the results of the image augmentation:



The distribution of various image classes post image augmentation is visualized along with the values and class labels in a tabulated form in code cell #7 of the IPython notebook. As seen from the image below this now show more uniform distribution of training images across all the classes.



Finally, the training, validation and testing images are normalized between the range of -1 to +1 and the data reshaped to the format/layout expected by the Conv. Layers in Model/Network in code cell #8 of the IPython notebook. The below image shows the dataset sizes/shapes after the augmentation and normalization operations.



```
Training dataset mean, min, max after normalizing :  -0.127579 -1.0 0.992188
X_train shape (57028, 32, 32, 1)
y_train shape (57028,)

Testing dataset mean, min, max after normalizing :  -0.0848068 -0.992188 0.992188
Validation dataset mean, min, max after normalizing :  -0.0900531 -0.992188 0.992188
X_test shape (12630, 32, 32, 1)
X_valid shape (4410, 32, 32, 1)
y_test shape (12630,)
y_valid shape (12630,)
```

Since the ‘pickled’ data set already contained separate training, validation and testing data set, no separate split/shuffle was performed on these. The testing data set was kept separate from training and validation was evaluated only once the model architecture and parameter where finalized.

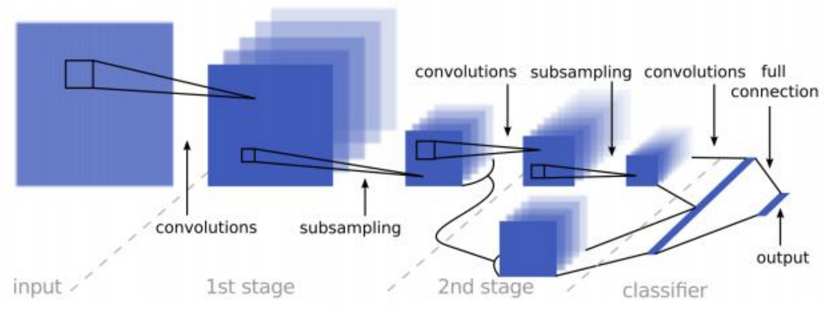
Model Architecture

An initial LeNet model from the labs in the lesson is implemented on code cell # 10 of the IPython notebook.

The code for final model is located on code cell # 10 of the IPython notebook. This model was based on the Multi-scale Conv. Neural Network for Traffic Sign classification paper by Pierre Sermanet and Yann LeCun (http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf) and consisted of the following layers:

Layer	**Description**
Input	32 x 32 x 1 Normalized Grayscale Image
Stage 1: Convolution 5x5	1x1 stride, outputs 28 x 28 x 32
Stage 1: Activation	RELU
Stage 1: Max Pooling	2 x 2 stride, outputs 14 x 14 x 32
Stage 2: Convolution 5x5	1 x 1 stride, outputs 10 x 10 x 64
Stage 2: Activation	RELU
Stage 2: Max Pooling	2 x 2 stride, outputs 5 x 5 x 64
Stage 3: Convolution 5x5	1x 1 stride, outputs 1 x 1 x 1600
Stage 3: Activation	RELU
Stage 4: Flatten Stage 2 Output and Stage 3 Output	Stage 2 output 5 x 5 x 64 -> 1 x 1 x 1600 Stage 3 output 1 x 1 x 1600
Stage 4: Concatenate flattened layer into single layer	Outputs 1 x 1 x 3200
Stage 4: Dropout Layer	Keep Prob. = 0.5
Stage 4: Fully Connected Layer	Outputs 43
Stage 4: Soft Max	

The below image shows the Multi-Scale Conv. Net Architecture:



Model Training

The code for training the model is located on code cell #11, #12, #13 of the IPython notebook
To train the model the following hyper tuning parameter where used:

- AdamOptimizer with default values
- Learning Rate 0.0009
  I experimented with different learning rate was from 0.0005 – 0.002 with best performance often being for values of 0.0009 and 0.001
- Batch Size 128
  I experimented with batch sizes of 128, 256 and 512 with a more consistent accuracy being achieved for batch size of 128
- Epochs 60
  I experimented with values of 10, 25, 50, 60, 75 and observed that accuracy improved and stayed consistently between 97.9% – 98.8% from Epoch 50 – 60 and remained relatively stable in high 98% afterwards.

.

Model Validation and Testing

The code for validation and testing of the model is located on code cell #14 of the IPython notebook. My final model results where

- Training/Validation set accuracy of 98.8%
- Testing set accuracy of 96.6%

I started with the basic LeNet architecture (along with the default hyper tuning parameters) from the labs in the lessons. With this I was able to achieve a validation accuracy of 95.5% and a testing accuracy of 93.5%.
I adapted the basic LeNet model to the Multi Scale Conv Network described in the paper by adding the following:

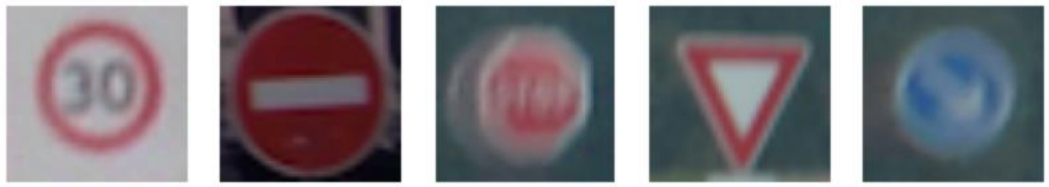
- an additional Convolution layer with output of 1 x 1 x 400
- combining the outputs from Stage 2 and Stage 3 conv. layers
- a drop layer with keep prob. of 0.75

For this iteration, I did not change the size of the output maps for Stage 1 and Stage 2 conv. layers. With this I was able to achieve a validation accuracy of 96% and a testing accuracy of 95%.
Finally, I increased the size of output maps for Stage 1 conv. layer (6 to 32) and Stage 2 (16 to 64) conv layer and updated the sizes of the subsequent stages. With this I was able to consistently achieve a validation accuracy of > 98.3% and a testing accuracy of > 96.3%.

I chose the Multi Scale Conv. Network model as it performs better compared to the basic LeNet model (95% vs 98.8% on training/validation set and 94% vs 96.6% on the testing set)

Test on New Images

I tested my model on the following signs I downloaded from the web:

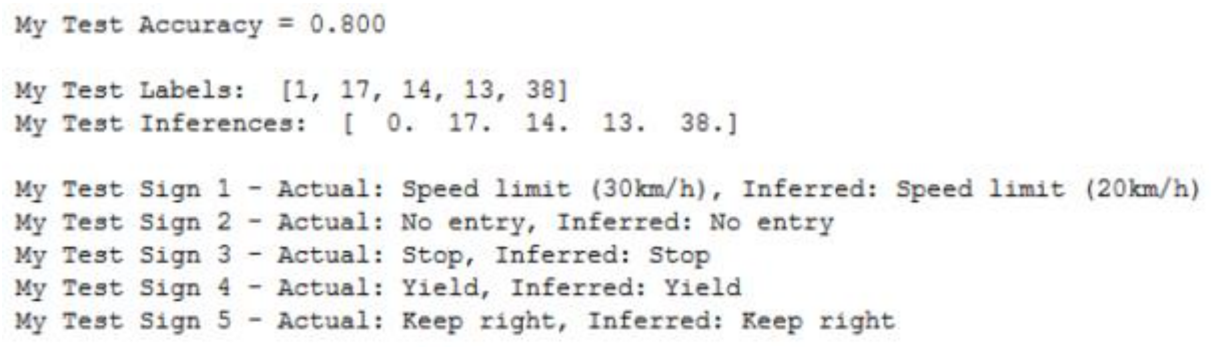


The first sign might be difficult to classify for the following reasons:

- its little different from the other 30 km/hr speed limit signs, specifically image has more whitespace (with a much smaller sign text) compared to other signs trained
- the confusion matrix from the earlier training/testing consistently shows class 1 (sign for 30 km/hr speed limit) as most frequent failing case

Additionally, we might have issues with image 3 and 5 as the images are extremely blurred.

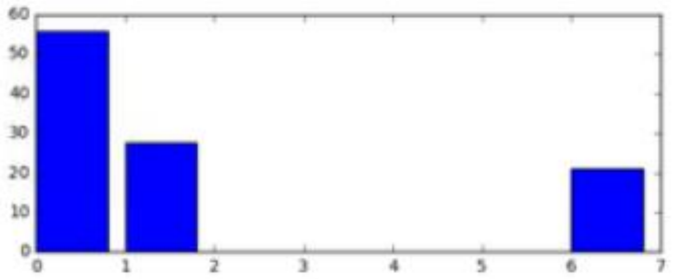
The code for making predictions on my final model is located in the code cell #17 of the IPython notebook. The results for the prediction are as below:

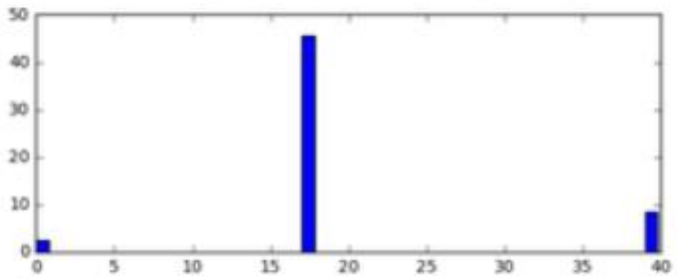


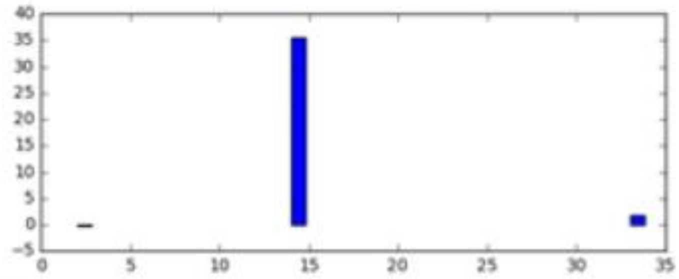
```
My Test Accuracy = 0.800

My Test Labels:  [1, 17, 14, 13, 38]
My Test Inferences:  [  0.  17.  14.  13.  38.]

My Test Sign 1 - Actual: Speed limit (30km/h), Inferred: Speed limit (20km/h)
My Test Sign 2 - Actual: No entry, Inferred: No entry
My Test Sign 3 - Actual: Stop, Inferred: Stop
My Test Sign 4 - Actual: Yield, Inferred: Yield
My Test Sign 5 - Actual: Keep right, Inferred: Keep right
```

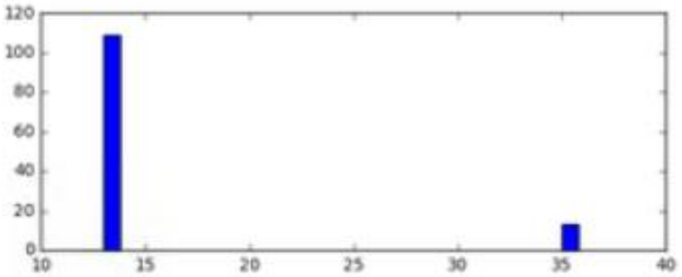
The code for analyzing the softmax predictions on my final model is located in the code cell #18 of the IPython notebook.
For image 1 the model probabilities are Class 0 (20 km/hr limit) 55%, Class 1 (30 km/hr speed limit) 30% and Class 6 (end of 80 km/hr speed limit) 20%.
For image 2 the model probabilities are Class 17 (No Entry) 45%, Class 40 (Roundabout Mandatory) 8% and Class 0 (20 km/hr speed limit) 2% and
For image 3 the model probabilities are Class 14 (Stop) 35%, Class 33 (Right Turn ahead) 2% and Class 3 (60 km/hr speed limit) 0.5%.
For image 4 the model probabilities are Class 13 (Yield) 100% and Class 36 (Go Straight or Left) 15%.
For image 5 the model probabilities are Class 38 (Keep Right) 35%, Class 14 (Stop) 10% and Class 41 (#nd of No Passing) 8%.

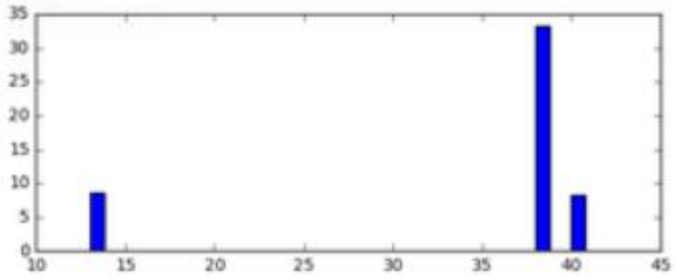
30


0 10 20 30 40 50 60
0 1 2 3 4 5 6 7


0 10 20 30 40 50
0 5 10 15 20 25 30 35 40


−5 0 5 10 15 20 25 30 35 40
0 5 10 15 20 25 30 35


0 20 40 60 80 100 120
10 15 20 25 30 35 40


0 5 10 15 20 25 30 35
10 15 20 25 30 35 40 45