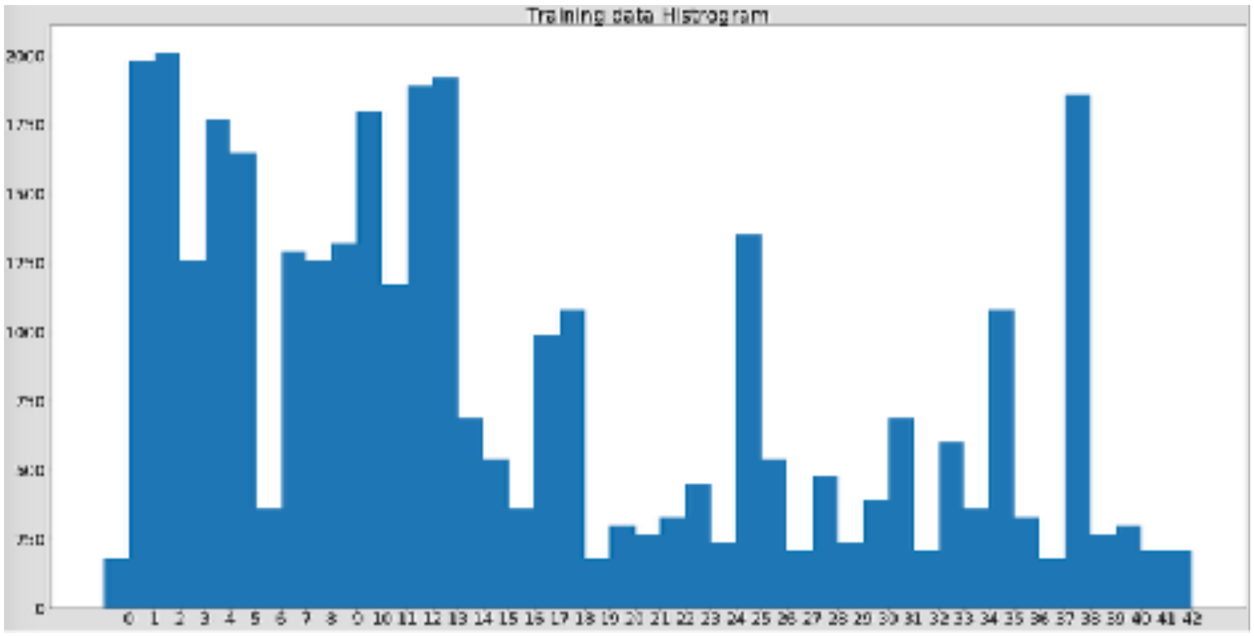
Traffic Sign Classifier Report

Provide a basic summary of the data set.

- The size of training set is 34799
- The size of the validation set is ?
- The size of test set is 12630
- The shape of a traffic sign image is 32*32*3
- The number of unique classes/labels in the data set is 43

Include an exploratory visualization of the dataset.



Design and Test a Model Architecture

- Preprocess data
  The data is converted GRAY image to avoid color bias.

- Balance data

The input training data is biased to certain traffic sign. Balance data step will make all class have the same number of examples by duplicating the existing example.

We are using 5 layer Convolutional Neural network.I reused the lenet pipeline with more filter and addtional dropout pipeline to avoid overfitting. This pipeline also used AdamOptimizer with adaptive learning rate to with better SGD optimization.

1. Convolutional.
   - Filter layer :Input = 32x32x1. Output = 28x28x6.
   - Relu activation layer
   - Pooling layer Input = 28x28x6. Output = 14x14x6.
2. Convolutional
   - Filter layer Input = 14x14x6. Output = 10x10x32.
   - Relu activation layer
   - Pooling layer Input = 10x10x32. Output = 5x5x32
   - Fatten layer Input = 5x5x32. Output = 800.
3. Fully connected Input = 120. Output = 84.
   - Relu activation layer
   - Dropout layer 0.5 keep rate
4. Fully connected Input = 120. Output = 84.
   - Relu activation layer
   - Dropout layer 0.5 keep rate
5. Fully connected Input = 84. Output = 10.

Describe how you trained your model.

This pipeline use small 128 batch size for max 60 epoch to train the network with previous defined network. The learning rate is 0.001. This pipeline also uses early termination, if the valication acuracy can not improve 0.01 for last 5 epoch, it will stop.
With this we are able to archive 95% -96% validation accuracy.

Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93.

Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.
My final model results were:

- training set accuracy of 0.991
- validation set accuracy of 0.951
- test set accuracy of 0.822011

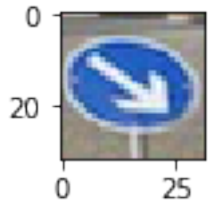
This is a common problem for image recognition, so CNN+ fully connect neural network model will work for this.

1. Use Lenet pipeline as is to check run the pipeline.
2. Train with more epoch to see if can improve accuracy
3. It can not improve the accuracy with more epoch, I tried to balance the input data by duplicating the existing data since the input data is not balanced. Certain class is more frequent than the others.
4. Convert the image to gray scale since the color does not matter too much for traffic sign and it will be affected by light
5. The training accuracy is high(0.999) and test accuracy still can not reach 0.93, we we add additional dropout layer to help overfitting issue
6. The accuracy reached 0.93, test with more filter for network and the accuracy improved slightly.

###Test a Model on New Images
####

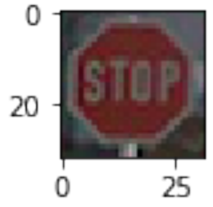
1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.
Here are five German traffic signs that I found on the web:

Case 0
correct
predict = Keep right
expect = Keep right

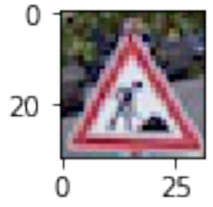


```
prediction rank=0 weight=0.999999 name =Keep right
prediction rank=1 weight=0.000001 name =No entry
prediction rank=2 weight=0.000000 name =Dangerous curve to the right
prediction rank=3 weight=0.000000 name =Turn left ahead
prediction rank=4 weight=0.000000 name =Go straight or right
```

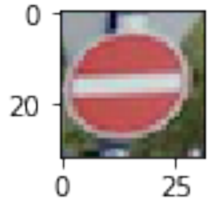
Case 1
correct
predict = Stop
expect = Stop



```
prediction rank=0 weight=0.850305 name =Stop
prediction rank=1 weight=0.048051 name =No entry
prediction rank=2 weight=0.029941 name =Turn right ahead
prediction rank=3 weight=0.009610 name =Yield
prediction rank=4 weight=0.008222 name =Keep right
Case 2
correct
predict = Road work
expect = Road work
```

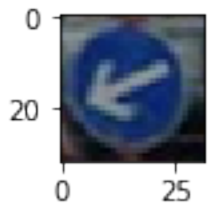


```
prediction rank=0 weight=0.745930 name =Road work
prediction rank=1 weight=0.162092 name =Dangerous curve to the right
prediction rank=2 weight=0.019920 name =Keep right
prediction rank=3 weight=0.015916 name =Bicycles crossing
prediction rank=4 weight=0.015053 name =Bumpy road
Case 3
correct
predict = No entry
expect = No entry
```



```
prediction rank=0 weight=0.999932 name =No entry
prediction rank=1 weight=0.000048 name =Stop
prediction rank=2 weight=0.000013 name =Priority road
prediction rank=3 weight=0.000004 name =Turn left ahead
prediction rank=4 weight=0.000002 name =Keep right
Case 4
correct
predict = Keep left
```

expect = Keep left



```
prediction rank=0 weight=0.990053 name =Keep left
prediction rank=1 weight=0.009778 name =Turn right ahead
prediction rank=2 weight=0.000132 name =Go straight or left
prediction rank=3 weight=0.000010 name =Speed limit (70km/h)
prediction rank=4 weight=0.000009 name =Stop
```

For the sample image find on the web, the current model is able to classify them. Case 2 have 2 have lower confidence comparing to other cases since the image is more blur. All of the examples are from Google Street view in German