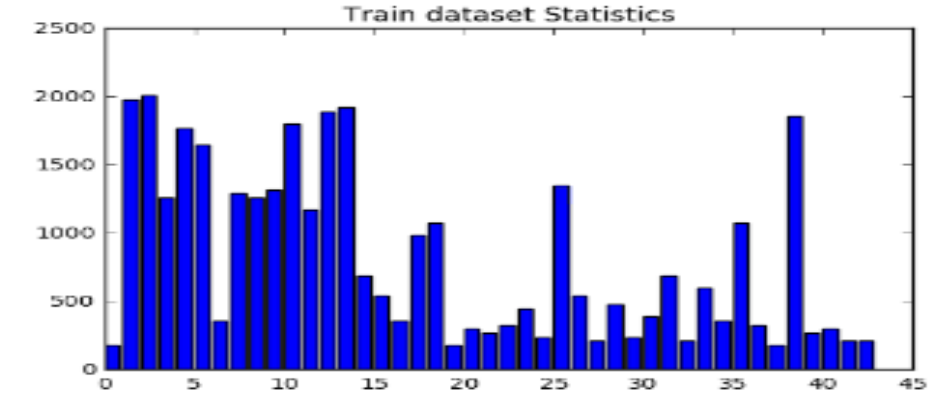
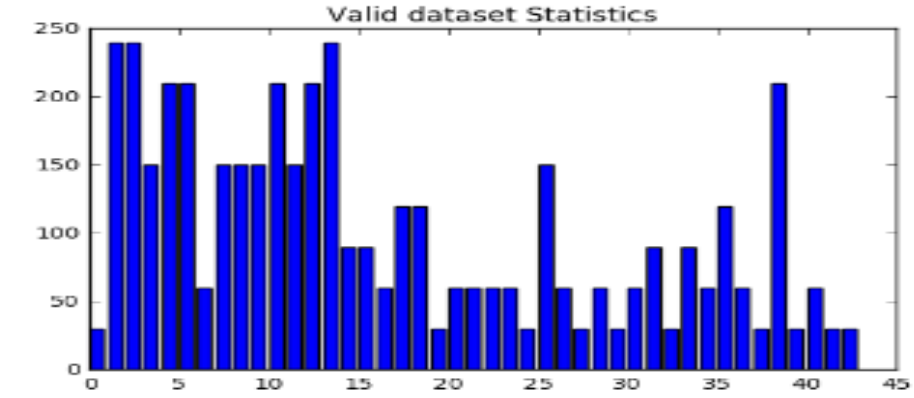
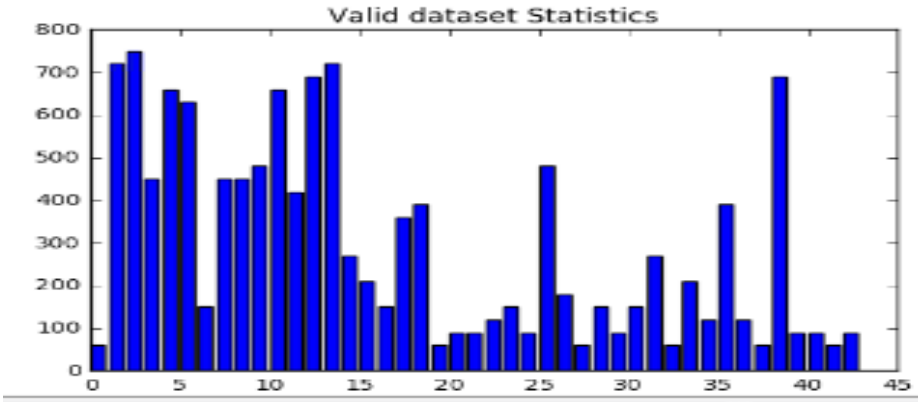
Project2 Traffic Sign Classifier

The goal of this project is to classify the German traffic sign images with convolutional neural network (CNN)

1. **Main Pipeline:**
    a. **Explore the dataset:**
    The goal is to check the statistics of the training dataset. There are about 34k images in the train dataset. We have a total of 43 different sign class, and it's clear some class has very little examples comparing to others.







b. Preprocess the dataset:

There are 3 steps in the preprocessing:

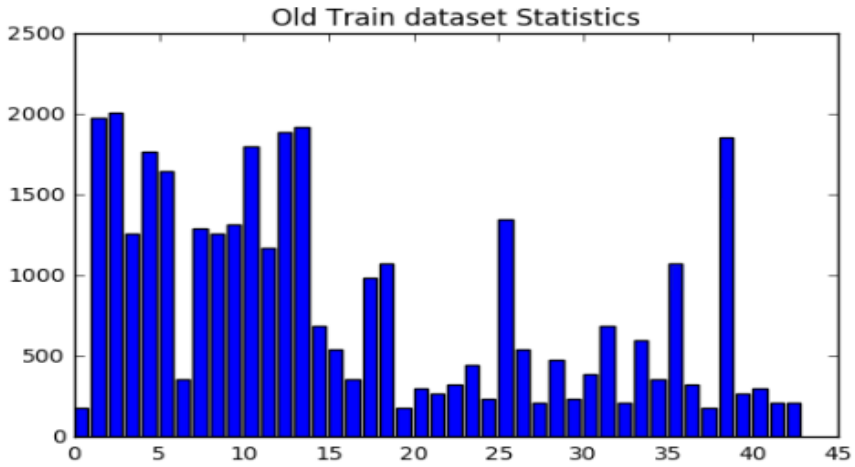
i). using the existing training dataset to generate more test case so that each class has at least ~1000 examples. For the image generation, I used:

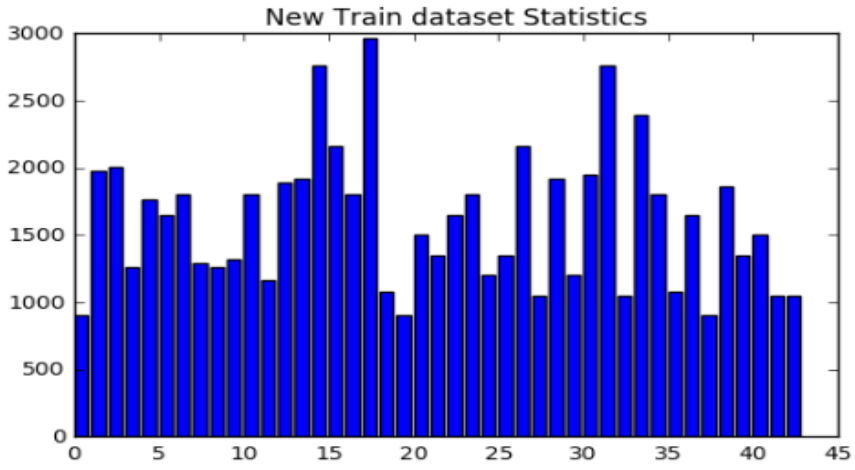
Flip(both horizontal and vertical)

Color offset

Rotation

Here is a comparison before and after the dataset expansion:





ii). Grayscale the image. For the sign classification, it's more about the shape of the sign rather than the color, so grayscaling the image is the right step

iii). Normalize the data, which put all the data in the same scale and with a zero mean distribution. I checked the dataset's mean value after normalization, it seems i not at zero, so I offset it. Here is the mean before and after:

Before:

```
train_mean: -0.301305995297
valid_mean: -0.347215411128
test_mean: -0.358215153428
```

After:

```
train_mean: 2.2978311554e-17
valid_mean: 1.93344962112e-17
test_mean: -5.68209155279e-17
```

c. CNN Architecture:

Here is my model architecture, modified from the LeNet. Mainly adding a 3rd CNN layer and dropout in the last 3 layers.

Layer	Description
Input	32x32x1 Grayscale image
Convolution 5x5 RELU	1x1 stride, Valid padding, outputs 32x32x16
Max pooling	2x2 stride, Valid padding, outputs 14x14x16
Convolution 5x5 RELU	1x1 stride, Valid padding, outputs 32x32x30
Max pooling	2x2 stride, Valid padding, outputs 10x10x30
Convolution 5x5 RELU	1x1 stride, Valid padding, outputs 32x32x400
Max pooling Dropout	2x2 stride, Valid padding, outputs 5x5x400
Flatten	outputs 400
Fully connected Dropout	inputs 400, outputs 130
Fully connected Dropout	input 130, outputs 86
Fully connected Dropout	input 86, outputs 43
Output	outputs 43

2. Model Architecture and Training & Validation accuracy

The first architecture I tried was the original LeNet5, which trained on the original dataset only gets about 90% accuracy. Based on my previous image classification experience, more filters brings more capabilities to the network, so I typically starts with a convolutional layer with 16 or 32 filters then double to the next then eventually gets to a few hundreds before a fully connected layer. The modified LeNet gave me about 5% improvement over the original LeNet5. Then the database expansion added another 2% to get the final 97~98%.

# of Filters in Each Convolution layer	LeNet5	Modified LeNet
1st	6	16
2nd	16	30
3rd		400

3. New images from the internet:

I was able to get some picture from google search. It turn out the model preform much less accurate on these images. I was only get ~30% accuracy, after some comparison between the original and new images, I realized most of the images in the training dataset are very much zoomed in on the sign itself. And in a lot of the image I downloaded, the background is relatively large (I have to resize the original image as they are much bigger than 32x32), so I hand selected 6 images that has a sign proportion similar to the original data set. Shown below

Here is a comparison of accuracy over each dataset, the model does seems over-fitted.

Training	Validation	Test	Internet –selected	Internet-raw
99.7%	97%	94%	83.3%	~30%

A couple of such examples of the internet image that produce poor test accuracy on the CNN:

And here are the 6 images I selected:





4. Final Thoughts

Overall, I'm a little disappointed with the performance on the new internet images. I think part of the reason is that the internet images was a little rawer (I only resize them into 32x32) than our training image, which was cropped to focus on the exact sign itself. However, I think if we can have larger training dataset (not the expanded with modified images) which includes more examples of rawer image would improve the accuracy.

Hyper Parameters:

```
5. epochs = 15
6. batch_size = 128
7. learning_rate = 0.001
8. keep_probability = 0.5
```