CarND Term1 Proj2

German Traffic Signs Classification

Joe Zhou – ibalpowr@gmail.com

In CarND Term1 Proj2, we explore a relatively small dataset provided by Udacity and apply convolutional neural network on the task of traffic signs classification. My results meet the project requirements. The framework I use is Tensorflow / CPU / Ubuntu. A good reference paper is [Sermanet/LeCun 2011].

1. Dataset Exploration and Augmentation

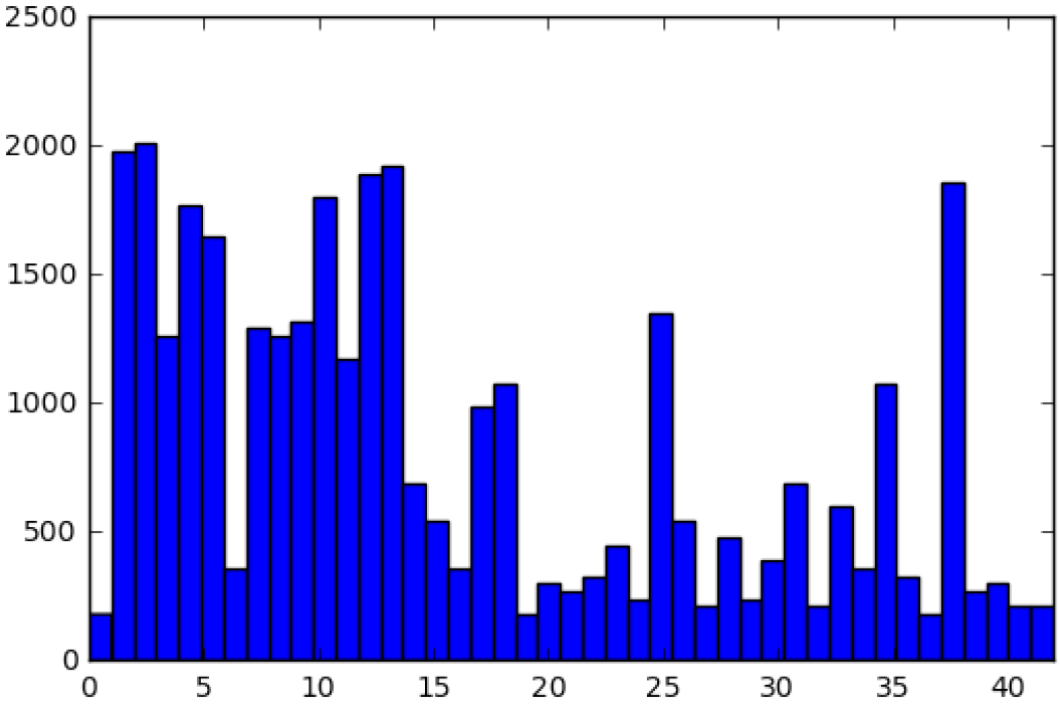
The goal of dataset exploration is to develop a level of domain knowledge on the dataset and then to gain a good sense of what can be accomplished and what pitfalls to avoid.

A. Dataset Summary

In my understanding, dataset summary means to quantify the Udacity dataset, which is derived from the GTSRB dataset. The training set has 34,799 images, which is relatively small. A good size should be in the range of 100k or even millions, such as ImageNet. There are 43 distinct classes of traffic signs and each image has the same shape of 32x32x3. Some images have poor exposure. So there is a need to augment and preprocess the original training set before feeding them to the convolutional neural network.

B. Exploratory Visualization

The first aspect of the original training set is the extremely uneven distributions among its 43 classes. Maybe this unevenness reflects the reality on the street. For example, Speed Limit 30km/h signs appear much more frequently than Speed Limit 20km/h signs on German streets. The largest class has more than ten times of number images than the smallest class has. So in the augmentation stage, we should make the distribution more even. Here is the histogram of the original training set.

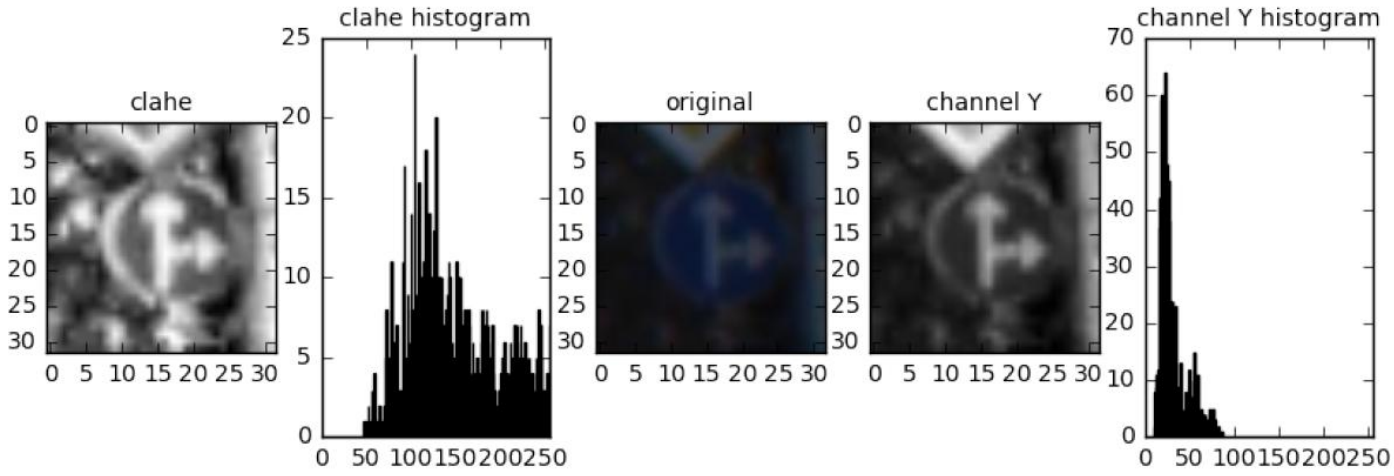


Histogram of Original Training Set

The images are captured and cropped from video files to make the traffic signs centered in the images and occupying the majority real estate of the images. Below are ten examples of Class 36 "Go straight or right":

Ten Sample Images of Class 36 and Their CLAHE's

The first row composes of five images from video frame sequences (as indicated by their images numbers) and five random images. The image sequences produce a large degree of redundancy, which should be addressed during the training. The second row is their respective CLAHE's (Contrast Limited Adaptive Histogram Equalization). Histogram Equalization (HE) is a popular image processing technique [Stanford EE368 Autumn 2015 Lect3 Histogram]. Here is an example:

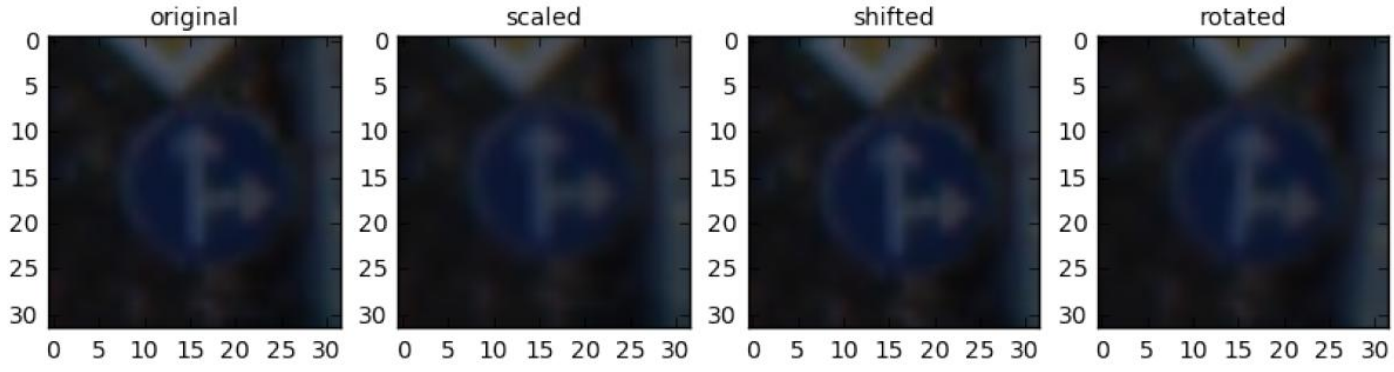


Histograms of X_train[1000] Before and After CLAHE

The original image is in the middle. Channel Y from YCrCb is basically the gray version (intensity) of the image. As shown in above, the histogram of CLAHE is more spread than the channel Y's histogram. The wider spread histogram in CLAHE produces a clearer image, for example, as shown in images 1004 and 1110 of Class 36.

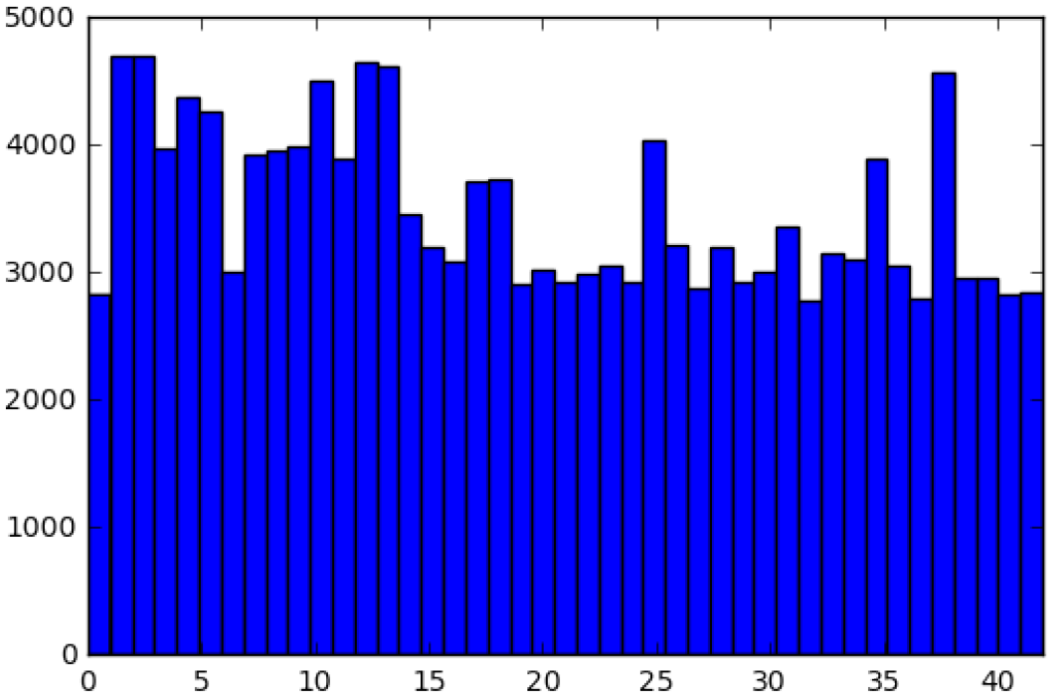
C. Data Augmentation

Constructing a sufficient and effective dataset is half of the battle in deep learning training since algorithms and architectures are quite open source. I augment the training set up to 150K images by three geometric transformations: scaling by [0.9, 1.1], shifting by [-2, 2] pixels, and rotating by [-15 °, 15 °], as suggested in reference paper [Sermanet/LeCun 2011]. Below is an example of X_train[1000]:



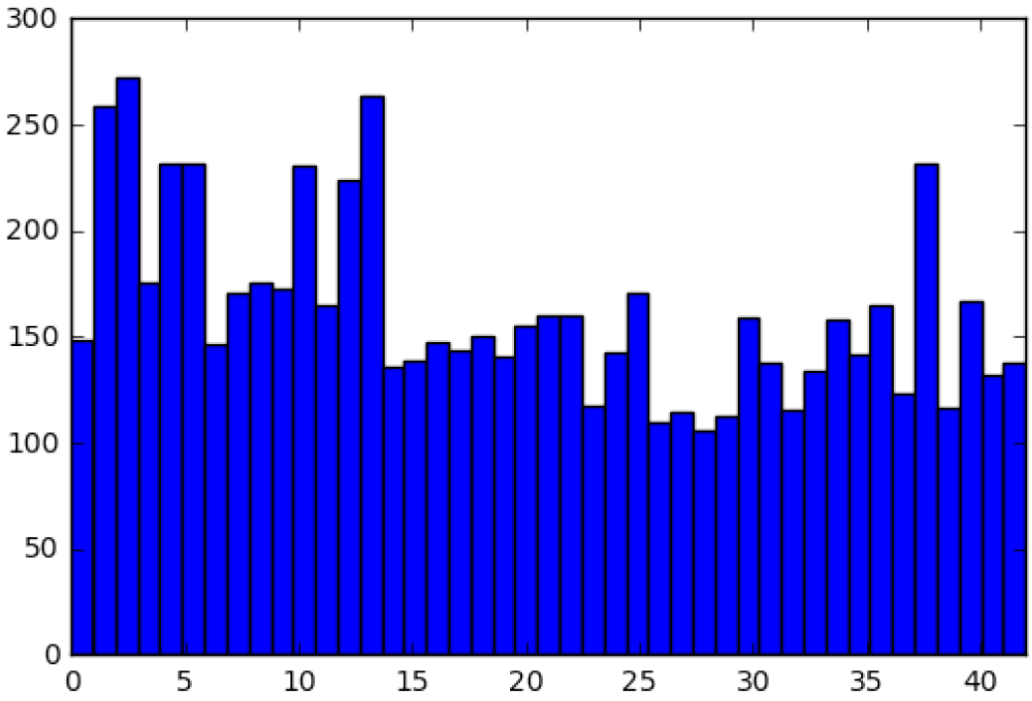
Examples of 3 Geometric Transformations on X_train[1000]

After applying augmentation to make the distribution more even among different classes, I have the final histogram of the training set as:



Histogram of Augmented Training Set

Similar transformations are applied to validation set too. Below is the final histogram of the validation set:



Histogram of Augmented Validation Set

Please note that after augmentation, training and validation sets should have similar shape.

2. Design and Test a Model Architecture

A. Preprocessing

I decide to use only Y channel for two reasons. First the traffic signs should be independent from the choice of color because there are color-blind people. Second the addition of two more channels will almost double the training time. After getting the Y channels, I preprocess them with CLAHE as discussed in previous section.

B. Model Architecture

LeNet5 architecture works well in MNIST dataset classification, which has training set size of 60K, ten classes, and image shape of 28x28x1. Original Udacity traffic sign training set has same order of magnitudes as MNIST's (35K vs. 60K, 43 classes vs. 10 classes, 32x32x3 vs. 28x28x1). Furthermore, using LeNet5 is also a show of my respect to its significance in the recent advancement of deep learning. So LeNet5 is my choice for classification. LeNet5 has five layers: 3 convolution layers and 2 fully connected layers. For details, below is my design sheet for LeNet5.

design parameters			Term1-Proj2						weights	bias	pixels
number of pixels					b	0	1	c	64,242	269	1,040,256
			input volume		128	32	32	1			131,072
pad	0	conv1	layer 1	conv-6_5x5	6	5	5	1	150	6	
stride	1										
			activation volume 1		128	28	28	6			602,112
		relu1	ReLU function								
pool_size	2	pool1	MaxPool function		128	14	14	6			150,528
pool_stride	2										
pad	0	conv2	layer 2	conv-16_5x5	16	5	5	6	2,400	16	
stride	1										
			activation volume 2		128	10	10	16			204,800
		relu2	ReLU function								
pool_size	2	pool2	MaxPool function		128	5	5	16			51,200
pool_stride	2										
			flatten function		128			400			
		fc1	layer 3	fc-120	120			400	48,000	120	
					128			120			15,360
		relu3	ReLU function								
		dropout1	dropout function		0.7						
		fc2	layer4	fc-84	84			120	10,080	84	
					128			84			10,752
		relu4	ReLU function								
		dropout2	dropout function		0.7						
		fc3	layer5	fc-43	43			84	3,612	43	
					128			43			5,504
			logit function								

((image - kernel + pad *2)/stride) + 1

((activation - pool)/stride) + 1

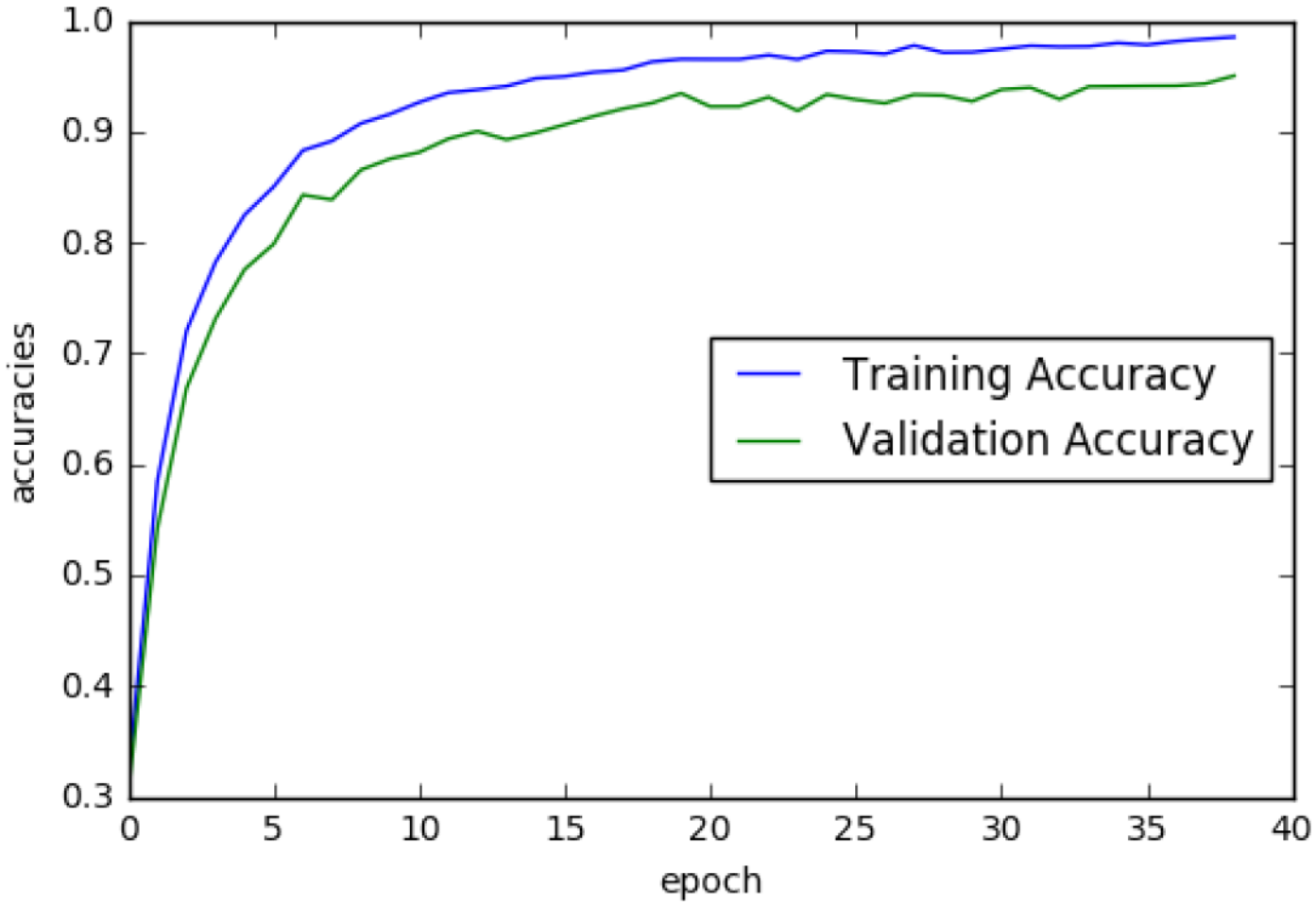
Design Sheet for LeNet5

Since deep learning training is a work of both art and science, faster iteration is desired. In the optimization step, the above design sheet helps me trying out different set of parameters much faster. The cells with double underlines are design parameters which can be tuned. The cells in cyan color are the memory usages which can

be used to find out the maximum batch size. The cells in black numbers are for sizes of weights/bias files. The correct sizes of pads, strides, and shapes are automatically calculated and propagated among all layers.

C. Model Training and Solution Approach

Model training is an iterative process. There are three major parameters to be optimized: learning rate, dropout percentage, and number of epochs. Since the training set size is relatively small, in 100K's, the choices of optimizer, weight initialization, and batch size are less relevant. I just use Adam (vs. SGD) optimizer, normal weight (vs. Xavier) initialization and batch size of 128 (vs. 256). According to Stanford CS231N Winter 2016 Lecture6 Slide 38, I start with "high learning rate" of 1e-3 then change it to "good learning rate" of 1e-4 when the accuracy starts to be plateau. Since there is large degree of redundancy in the original training set, in order to break the symmetry and reduce the amount of overfit, I settle with the dropout at keep_prob as 0.7 after trying out others. And the number of epochs I settle is about 40 after consistently reaching training accuracy of 98% and validation/test accuracies of 94%, which meets the project requirement of minimum 93%. Below is a plot of training accuracy and validation accuracy on 40 epochs:



Plot of Training/Validation Accuracies

(Validation accuracy can be further improved if I double the training set to 300K by applying more augmentation and train longer by increasing the epochs. But this improvement will take in hours in training rather than my current training time of minutes. So there is a tradeoff.)

3. Test the Model on New Test Images

A. Acquiring New Test Images

I download a German street view video from Youtube. Then I capture and crop out five traffic signs from video frames. Here are the five new test images:

Five New Test Images from a Youtube Clip

The 1st test image is normal. The 2nd image is not centered. The 3rd image 120km/h closely looks like 20km/h or 30km/h. The 4th image is partially occluded on the left bottom. The 5th image is rotated to the left.

B. Performance on New Test Images

The score is 4 out of 5, i.e. 80% correct, which is worse than the 94% score in original testing set maybe due to the fact that the small size (only size of five) of the new test images is not a good representation of test set.

Image	Top Prediction (red = wrong)
Turn right ahead	Turn right ahead
Priority road	Priority road
Speed limit (120km/h)	Speed limit (120km/h)
General caution	General caution
Speed limit (30km/h)	Keep right

C. Top 5 Softmax Probabilities on New Test Images

Probabilities are very interesting.

The failure case is on the "Speed limit (30km/h)" sign. Below are its top five softmax probabilities:

```
1st prediction w/ probability of 0.7848: Keep right
2nd prediction w/ probability of 0.1439: Speed limit (120km/h)
3rd prediction w/ probability of 0.0264: Speed limit (20km/h)
4th prediction w/ probability of 0.0139: Speed limit (50km/h)
5th prediction w/ probability of 0.0129: Turn left ahead
```

The top probability prediction is “Keep right”, which is totally off from the correct answer. I can improve the prediction by augmenting the “Speed limit (30km/h)” sign class with more rotations since the test image is rotated to the left.

Two signs of “Turn right ahead” and “Priority road” almost get 100% correct probabilities. So I can keep their training sets as they are.

```
1st prediction w/ probability of 0.9991: Turn right ahead
2nd prediction w/ probability of 0.0007: Roundabout mandatory
3rd prediction w/ probability of 0.0001: Keep left
4th prediction w/ probability of 0.0000: No entry
5th prediction w/ probability of 0.0000: Ahead only

1st prediction w/ probability of 1.0000: Priority road
2nd prediction w/ probability of 0.0000: Roundabout mandatory
3rd prediction w/ probability of 0.0000: No vechiles
4th prediction w/ probability of 0.0000: No passing
5th prediction w/ probability of 0.0000: Yield
```

And the last two signs of “Speed limit (120km/h)” and “General caution” have the correct probabilities below 90%. But they are in the right direction. Maybe I need to train them longer.

```
1st prediction w/ probability of 0.8520: Speed limit (120km/h)
2nd prediction w/ probability of 0.1081: Speed limit (100km/h)
3rd prediction w/ probability of 0.0216: Speed limit (70km/h)
4th prediction w/ probability of 0.0148: Speed limit (20km/h)
5th prediction w/ probability of 0.0030: Speed limit (30km/h)

1st prediction w/ probability of 0.7114: General caution
2nd prediction w/ probability of 0.2848: Traffic signals
3rd prediction w/ probability of 0.0025: Go straight or right
4th prediction w/ probability of 0.0003: Pedestrians
5th prediction w/ probability of 0.0003: Ahead only
```