1. Data Exploration

(1). Dataset Sumary

Numpy library was utilized to calculate the amount of the training data, validation and test set. The size of traffic sign images is 32x32x3, and there are totally 43 kinds of signs in the data set. The results were printed out in the following cell.

(2). Exploratory Visualization

A image in the training data set was selected randomly and printed out (Fig 1.). The three bar charts below in Fig 2 illustrates the amount of 43 kinds of signs in the training, validation and test data set respectively.

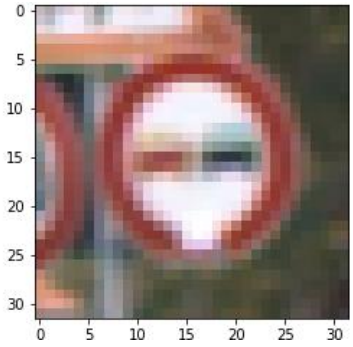

Fig 1. A training image

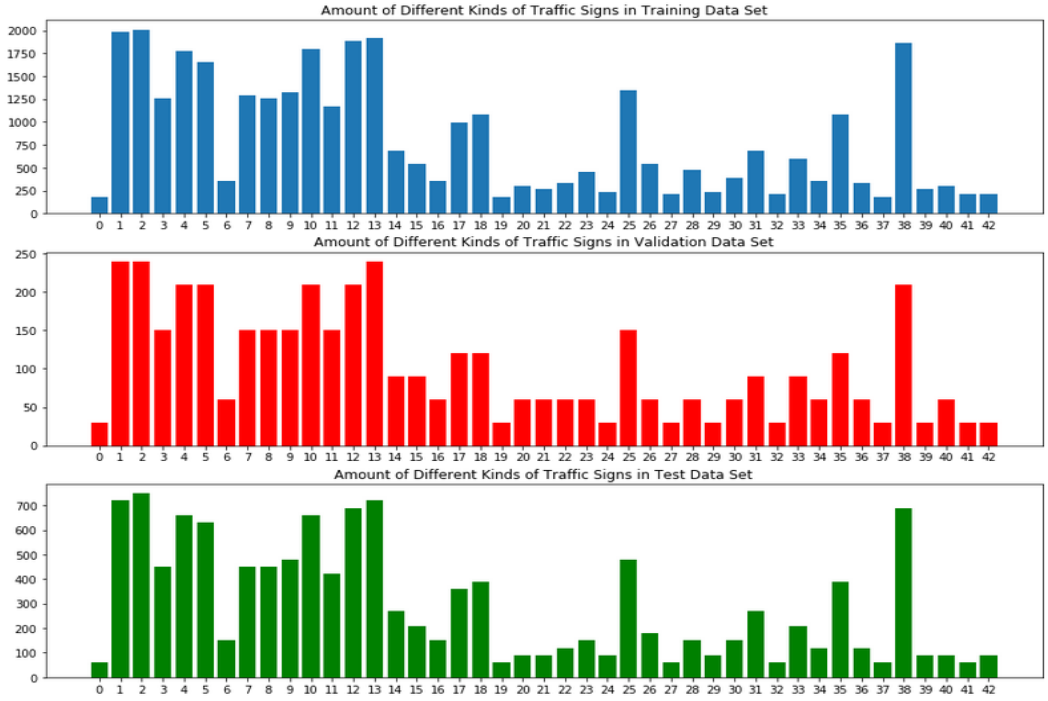

Fig 2. Amount of 43 classes in training, validation and test data set

2. Design and Test a Model Architecture

(1). Prepocessing

All the images were converted to grayscale and then normalized. The equation (pixel - 128)/128 was utilized to normalize the gray image data because it can produce zero mean and small equal variance. Here is an example showing before and after grayscaling:

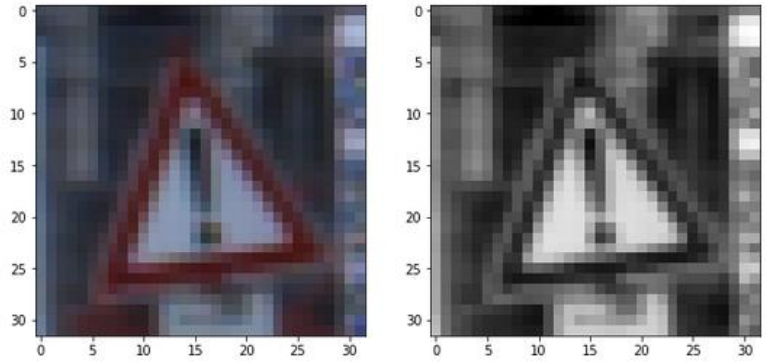

Fig 3. Initial vs grayscale image

(2). Model Architecture

LeNet was chosen to build the model. My final model architecture consisted of the following layer:

	Layer	Description	Output Size
Input	Grayscale image	32x32x1 Grayscale Image	32x32x1
Layer 1	Convolution Layer 1	Size 5x5; 1x1 Strides; Valid Padding	28x28x50
	RELU	Activation Function	28x28x50
	Dropout	Keep Probability = 0.6	28x28x50
	Max Pooling	Ksize = 2x2; 2x2 Strides	14x14x50
Layer 2	Convolution Layer2	Size 5x5; 1x1 Strides; Valid Padding	10x10x100
	RELU	Activation Function	10x10x100
	Max Pooling	Ksize = 2x2; 2x2 Strides	5x5x100
Layer 3	Fully Connected Layer	Input: 2500	700
	RELU	Activation Function	700
Layer 4	Fully Connected Layer	Input: 700	250
	RELU	Activation Function	250
Layer 5	Fully Connected Layer	Input: 250	43
	Logits	Output of the Model	43

There are totally 5 layers, with two convolutional layers and three fully connected layers. There is a dropout layer with keep probability 0.6 connected to the first convolution layer to prevent overfitting.

(3). Model Training

To train this model, I used AdamOptimizer as training optimizer because it reached higher accuracy with faster speed compared with the GradientDescenOptimizer. The batch size was selected to be 50 because it can reach higher accuracy at starting point compared to other batch size, which makes it faster to reach high accuracy. Epochs was set to be 30 because model did not realize higher accuracy after the 30^{th} epoch. The validation accuracy fluctuated when reached 94%, and once the validation accuracy was higher than 95%, the loop was terminated then the model was saved. The learning rate was finally set to be 0.0005, because if it was set higher, the accuracy would fluctuate a lot, which makes it harder to be convergent.

(4). Solution Approach

The initial and final parameters of the model are respectively shown below:

Epochs:	40	30

Batches	128	50
Learning Rate:	0.001	0.0005
Convolution Layer1:	5x5x6	5x5x50
Convolution Layer2:	5x5x16	5x5x100
Fully Connected Layer 1	Input:400 output:120	Input:2500 output:700
Fully Connected Layer 2	Input:120 output:84	Input:700 output:250
Keep Probability	No dropout	0.6

With the initial setting, the final validation accuracy was approximately 92%. By reducing the learning rate and batches size, the accuracy rose to about 93%. The final learning rate is 0.0005 because the training would be very slow if the learning rate is lower. Since the accuracy can be higher, I believed the model is too simple to process the images, which means a little under-fitting. To solve this, the depth of the convolutional layer and fully connected layers were raised up, which brings more parameters to be trained in the model. However, complex model may cause over-fitting problem, to prevent this, a drop-out layer was introduced and the keep probability was set to be 0.6. The final performance of the model is shown below:

Training data accuracy:	100%
Validation data accuracy:	96.1%
Test data accuracy:	93.8%

3. Test a Model on New Images

(1). Acquiring New Images

Here are ten German traffic sign found on the web:

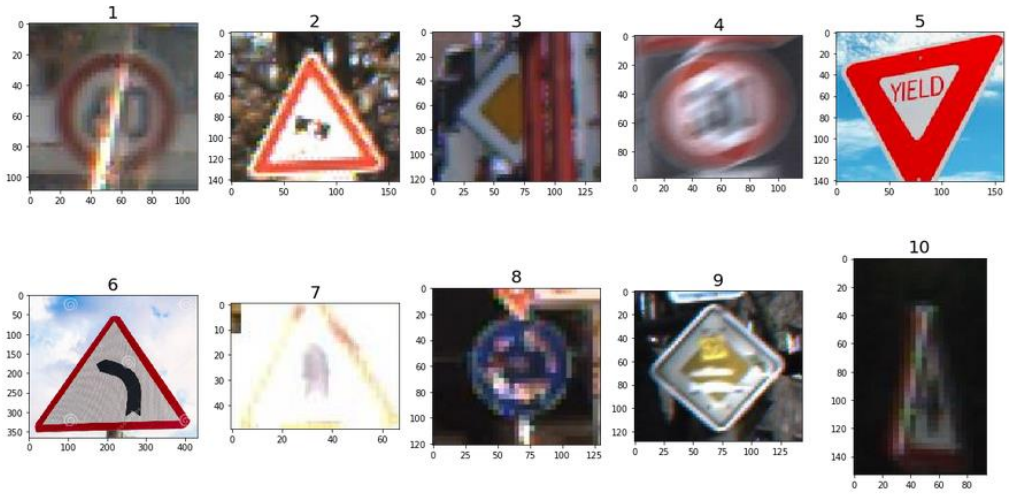

Fig 4 New images found on the web

The 1st, 7th and 9th images may be difficult to classify because of the sun glare. The 2nd and the 8th seems to be damaged. There is something holding part of 3rd traffic sign back and makes it incomplete. The 4th one is fuzzy because of motion blur. The 5th and the 10th image is taken from the right side rather than in front. Only the 6th image is relatively clear.

(2). Performance on New Images

Here are the prediction of the model. The accuracy of the prediction on new images is 90%.

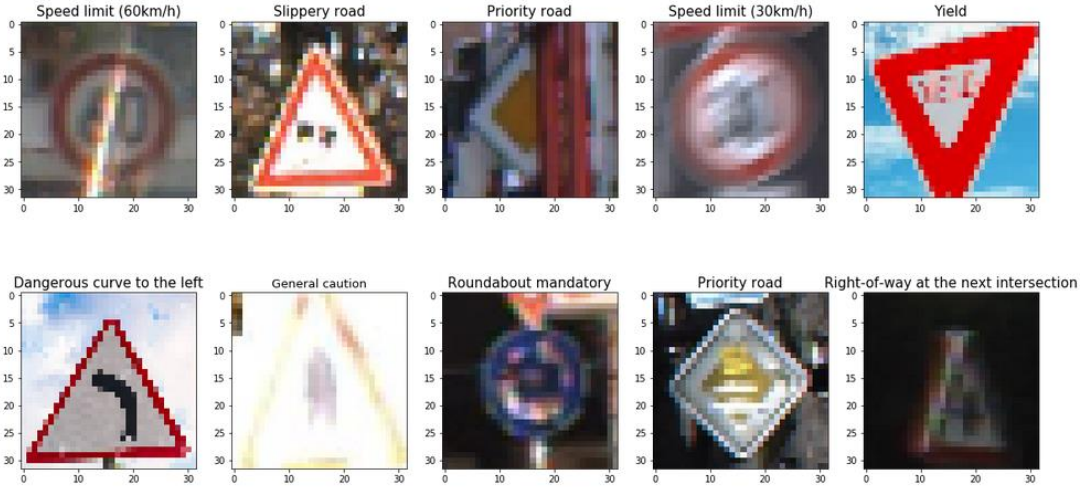

Fig 5 Predictions on the new images

Image	Prediction
Speed Limit (60km/h)	Speed Limit (60km/h)
Slippery Road	Slippery Road
Priority Road	Priority Road
Speed Limit (30km/h)	Speed Limit (30km/h)
Yield	Yield
Dangerous Curve to the Left	Dangerous Curve to the Left
General Caution	Right-of-way at the next intersection
Roundabout mandatory	Roundabout mandatory
Priority Road	Priority Road
Right-of-way at the next intersection	Right-of-way at the next intersection

(3). Model Certainty – Softmax Probabilities

The top 5 probabilities of all the ten images were printed out in the 'Output Top 5 Softmax Probabilities For Each Image Found on the Web' code cell. It was found that for the 1st 2nd 3rd 4th 5th 6th 8th and 9th images, the highest probability is over to 94%, which means the model is certain about the content. The model is confused about 10th image as the probabilities shown in Fig 6. It can be seen that the model is confused among 'Right-of-way at the next intersection' (38.00%), 'Dangerous curve to the right' (16.90%) and 'Dangerous curve to the left' (14.60%) signs. That may impute to the similar triangular shape and large side shooting angle. The model prediction is wrong on the 7th image (Right-of-way at the next intersection), with 37.99% probability to be seen as General Caution because the image quality is bad due to the strong sun glare. Moreover, these two signs have the same triangular shape and similar pattern.

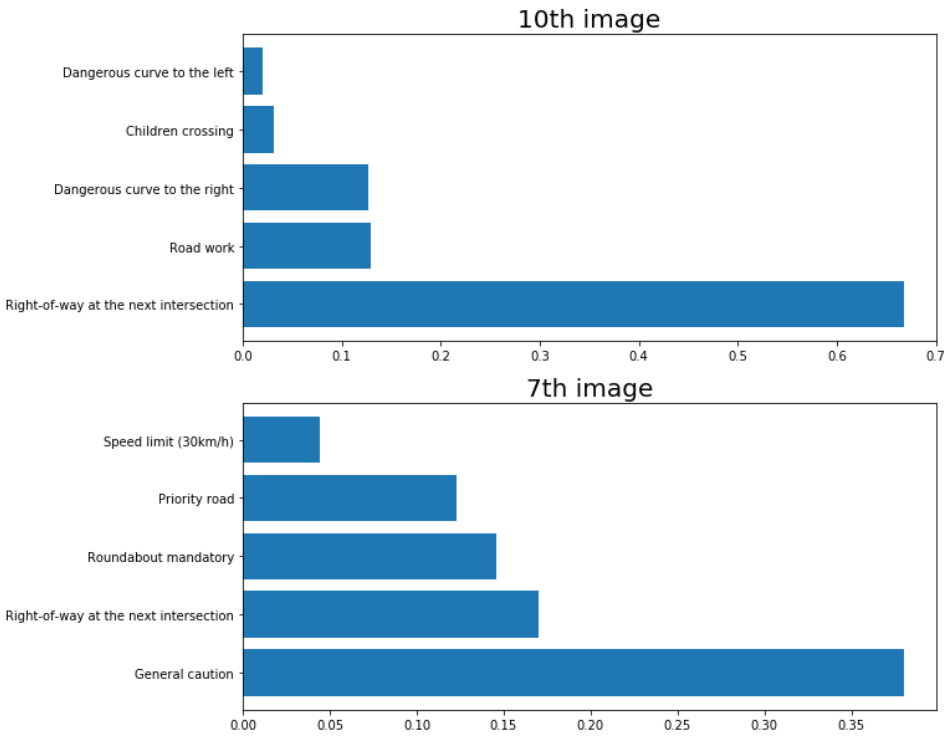

Fig 6 Top 5 probabilities predicted on the 10th ,7th and 2nd images

4. Visualize the Neural Network's State with a Test Image

The output of the first layer (max_pooling layer) in size 14x14x50 is visualized in the final code cell using the outputFeature() function provided. The height of this layer is 50, which means the model are looking for 50 features in this layer. The output is shown below. It can be seen that almost all the feature contains the triangular outline shape, and some are finding the shape of the inside pattern.