Traffic Sign Recognition

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:
- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

Here is a link to my project code

Data Set Summary & Exploration

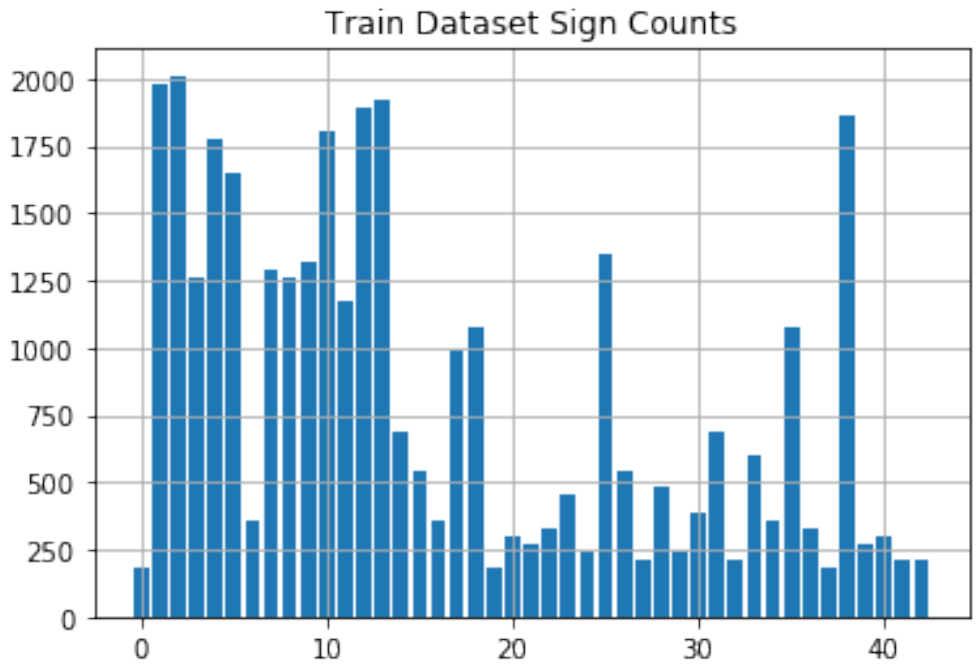
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

I used the numpy library to calculate summary statistics of the traffic
signs data set:

Number of training examples = 34799
Number of validation examples = 4410
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes/labels = 43

2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data is unbalanced.
Some classes have more data than others.

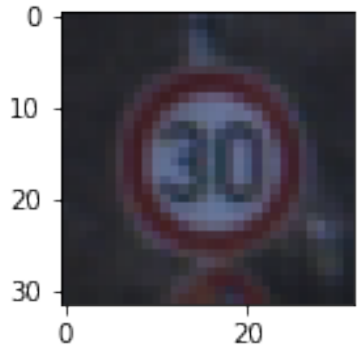


Design and Test a Model Architecture

1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

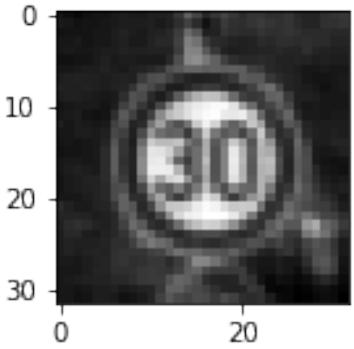
[Jupyter Cell# 6]

Original sample image:

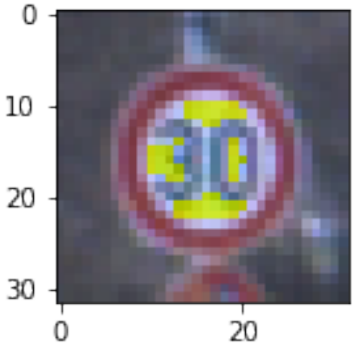


As a first step, I decided to convert the images to grayscale because it was recommended by the course, Sermanet-lecun mentioned converting image into grayscale improved their accuracy in their paper “Traffic Sign Recognition with Multi-Scale Convolutional Networks”.

Here is an example of a traffic sign image after grayscaling.

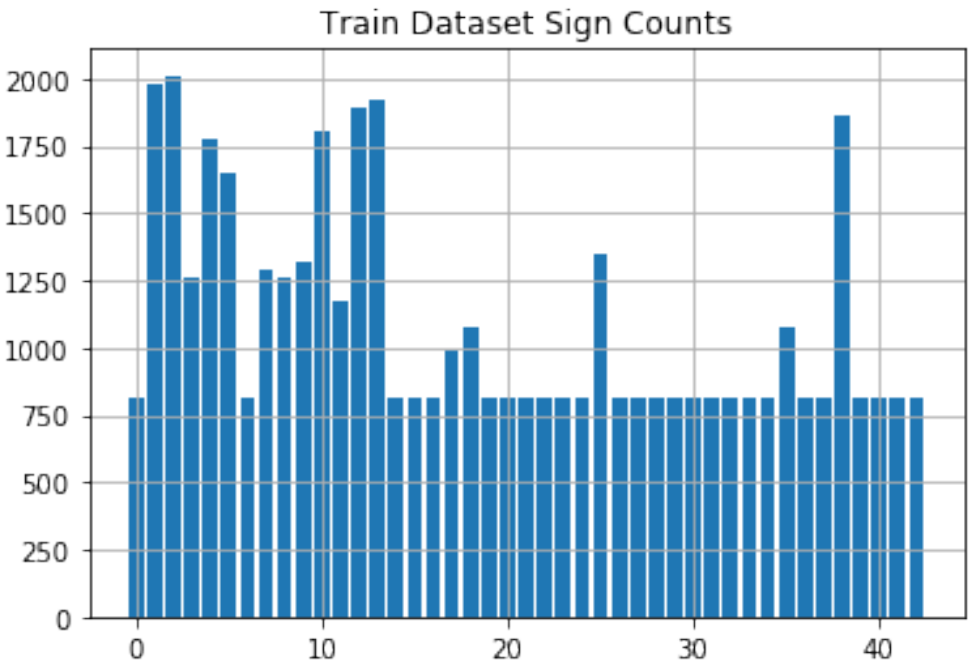


As a last step, I normalized the image data because we want our data to have mean of 0 and a variance of 1. This means we want to have a same range of values for all the inputs to have stable weights and bias from neural network. Its hard for our model to generalize if 1 image is in strong dark range and another is in strong bright range.



Since the data was not balanced I generated additional data so that model is not bias to some classes more than others.

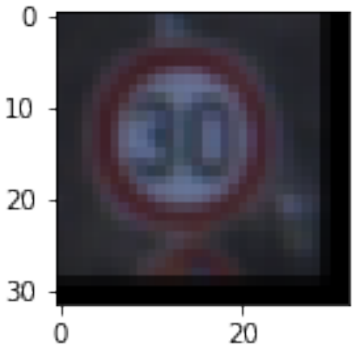
[Jupyter Cell# 14]
I calculated the average number of images per class. Any class with total number of examples less than the average was increased to the average amount.



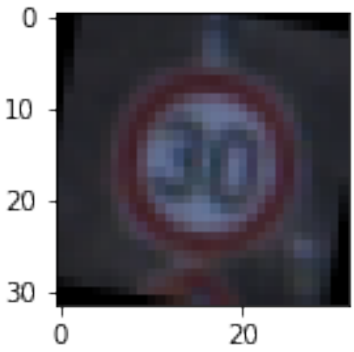
All the new images were created with data augmentation. Four steps used for data augmentation are rotate, translate and affine transformation.

Here is an example an augmented image:

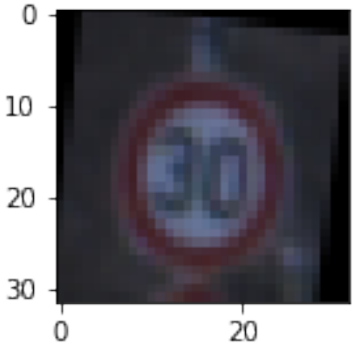
[Jupyter Cell# 6]
Translate:



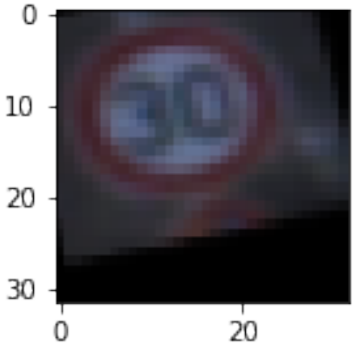
Rotate:



Affine Transform:

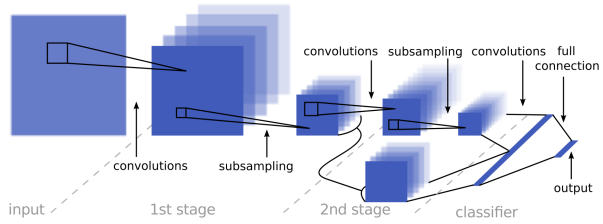


Augmented as a whole:



2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

My final model architecture was from “Traffic Sign Recognition with Multi-Scale Convolutional Networks ” by Pierre Sermanet and Yann LeCun.



Final model, LeNet_sermLecun *[Jupyter Cell# 21]*

Input	32x32x1 RGB image
Convolution 5x5x1	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max Pooling	2x2 stride, outputs 14x14x6
Flatten the result, conv1	
Convolution 5x5x6	1x1 stride, Valid padding, outputs 10x10x16
RELU	
Max Pooling	2x2 stride, outputs 5x5x16
Flatten the result, conv2	
Concatenate conv1, conv2	
Fully connected	input 1576, outputs 120
RELU	
Dropout	
Fully connected	input 120, outputs 43

3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

[Jupyter Cell# 24]

To train the model, I used
- Adam Optimizer(from Lenet)
- learning rate: 0.0009
- epoch: 80
- batch size: 120

All the example set was shuffled
Hyperparameters were set with a lot of trial and testing.

4. 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.

I worked on more or less 3 architecture, one was default 'Lenet' architecture which was provided in the course. I wasn't satisfied with the accuracy as it was around 93%. I edited this architecture with addition of third convolution layer and removing the third fully connected layer. My accuracy was around 91%. I still did not get the satisfactory accuracy. Third architecture I tried was from "Traffic Sign Recognition with Multi-Scale Convolutional Networks " by Pierre Sermanet and Yann LeCun which eventually gave me satisfactory accuracy of around 96%.

Lenet architecture from course:

Grayscale, Normalization but no augmentation, no drop out

Training accuracy: 99.3

Validation accuracy: 93.7

Since there is a overfitting, I introduced dropout from now on.

Grayscale, Normalization with augmentation but no drop out

Validation accuracy: 93.5

Grayscale, Normalization with augmentation and drop out

Validation accuracy: 92.6

Lenet_updated

Validation accuracy: 91.4

LeNet_sermLecun

Validation accuracy: 96.2

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 six German traffic signs that I found on the web which was edited and/or resized to 32x32

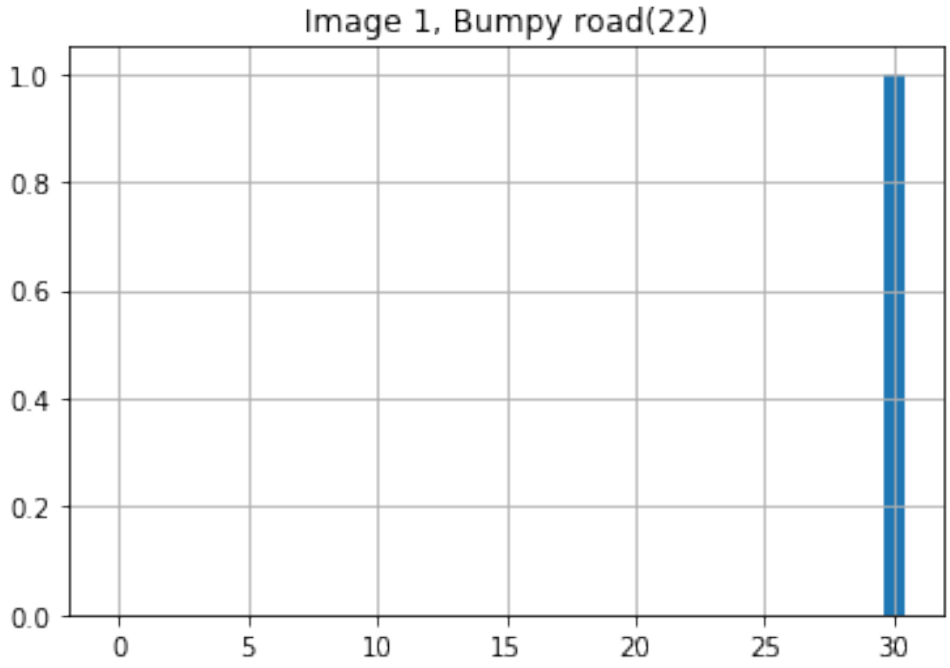
2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

Accuracy on provided Test Set = 0.812
Accuracy on my new images test set = 0.667

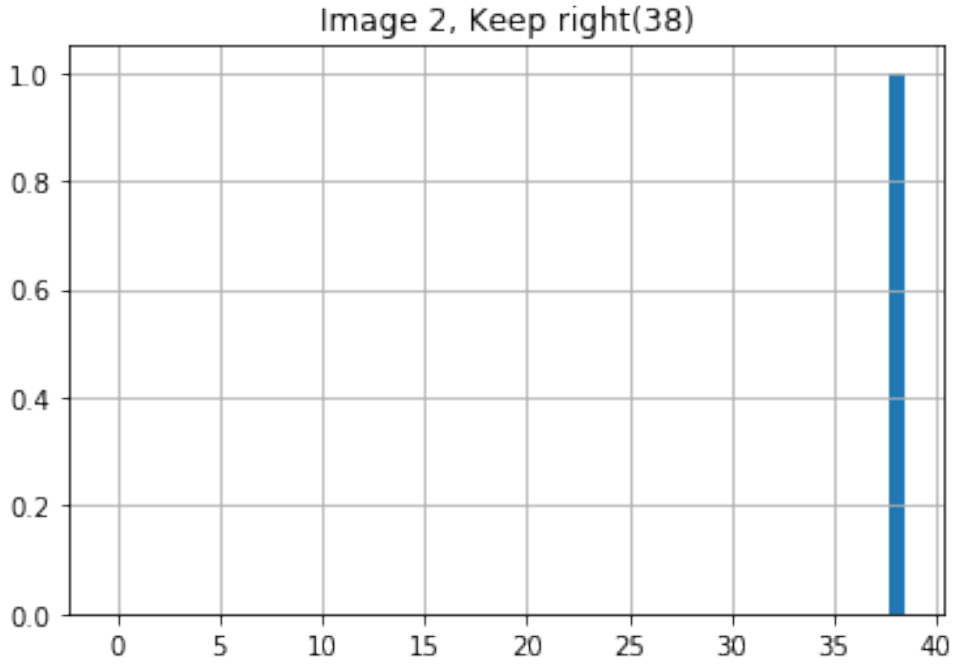
Model predicts 4 out of 6 images correct in which 3 of them have 100% prediction and 1 has 99%, still pretty high. My model has low prediction estimate on ‘bumpy road’ and ‘stop sign’. Although it predicted well on ‘stop sign’ sometimes, I think model has trouble predicting when the sign has more curves than straight lines. Also ‘bumpy road’ and ‘stop sign’ both has average number of training example.

Top 5 softmax probabilities on new images:

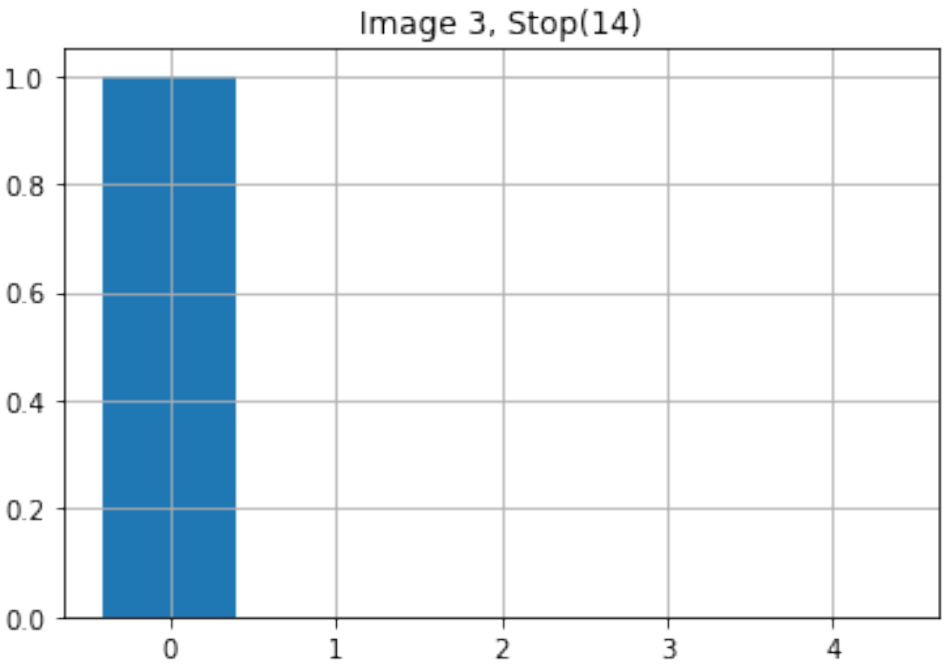


Top 5 softmax probabilities for Bumpy road(22)
Beware of ice/snow(30) : 1.0
Right-of-way at the next intersection(11) : 1.451479567777769e-26

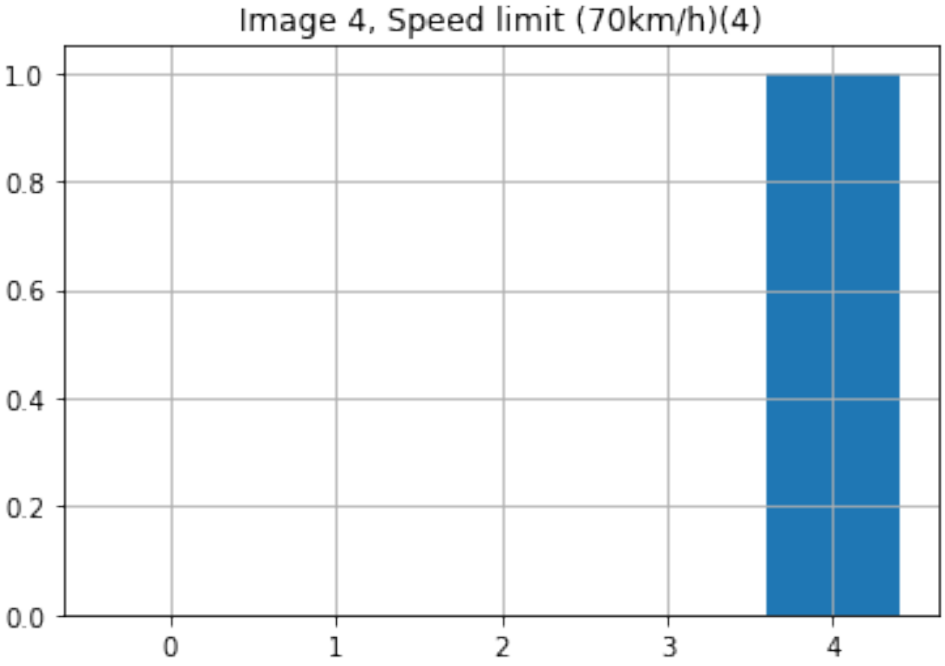
Speed limit (20km/h)(0) : 0.0
Speed limit (30km/h)(1) : 0.0
Speed limit (50km/h)(2) : 0.0



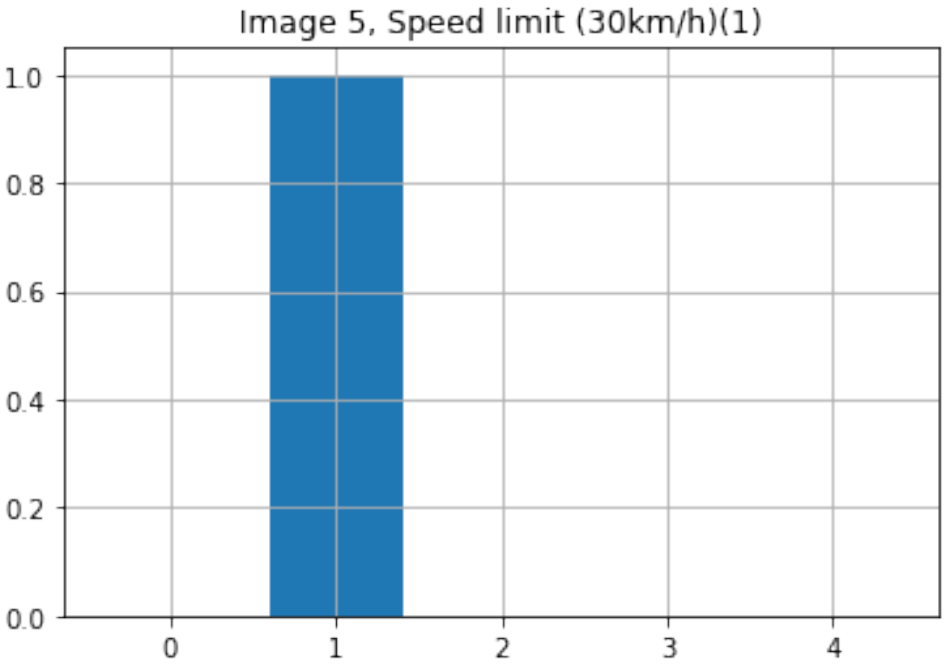
Top 5 softmax probabilities for Keep right(38)
Keep right(38) : 1.0
Speed limit (20km/h)(0) : 0.0
Speed limit (30km/h)(1) : 0.0
Speed limit (50km/h)(2) : 0.0
Speed limit (60km/h)(3) : 0.0



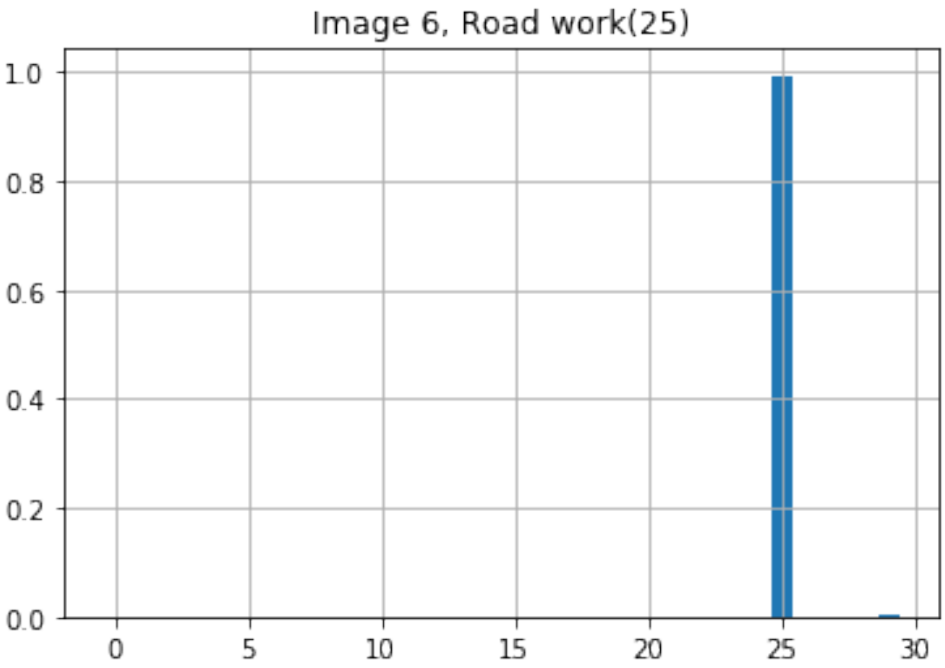
Top 5 softmax probabilities for Stop(14)
Speed limit (20km/h)(0) : 1.0
Speed limit (30km/h)(1) : 0.0
Speed limit (50km/h)(2) : 0.0
Speed limit (60km/h)(3) : 0.0
Speed limit (70km/h)(4) : 0.0



Top 5 softmax probabilities for Speed limit (70km/h)(4)
Speed limit (70km/h)(4) : 1.0
Speed limit (20km/h)(0) : 0.0
Speed limit (30km/h)(1) : 0.0
Speed limit (50km/h)(2) : 0.0
Speed limit (60km/h)(3) : 0.0



Top 5 softmax probabilities for Speed limit (30km/h)(1)
Speed limit (30km/h)(1) : 1.0
Speed limit (20km/h)(0) : 0.0
Speed limit (50km/h)(2) : 0.0
Speed limit (60km/h)(3) : 0.0
Speed limit (70km/h)(4) : 0.0



Top 5 softmax probabilities for Road work(25)
Road work(25) : 0.9940594434738159
Bicycles crossing(29) : 0.005940550938248634
Speed limit (20km/h)(0) : 0.0
Speed limit (30km/h)(1) : 0.0
Speed limit (50km/h)(2) : 0.0

Todo: Optional(Step 4) of the project, try augmenting more examples in training set to analyze the accuracy, calculate precision and recall.