Traffic Signs Recognition system using Deep Learning Solution

In this project, i`m purposing a pipeline for traffic signs recognition using Deep learning methods. thes model get use of official german traffic signs dataset for training and testing purposes.

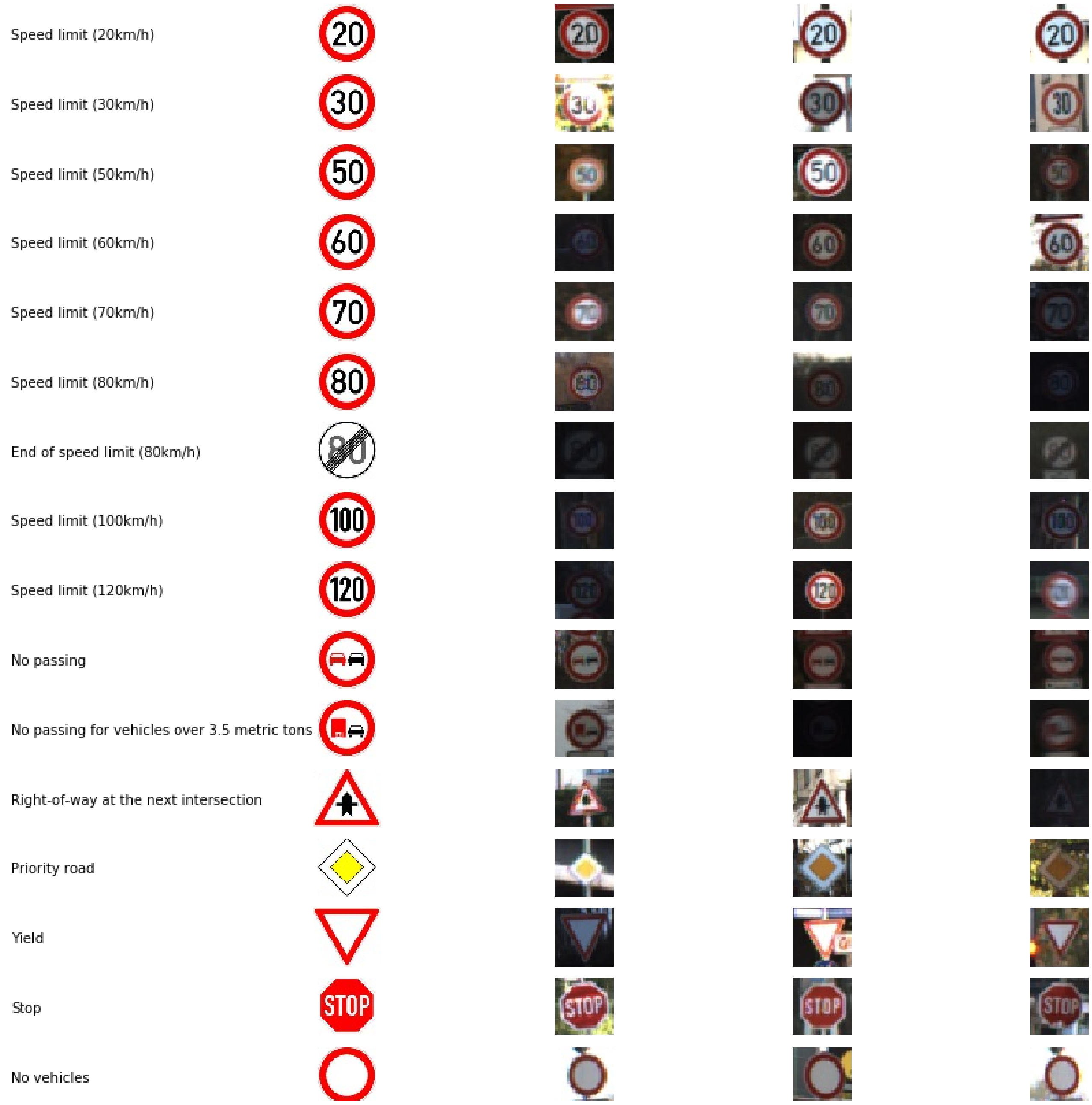
The following sections descripe dataset analysis, model architecture, evaluations and future improvements.

Data Set Summary & Exploration

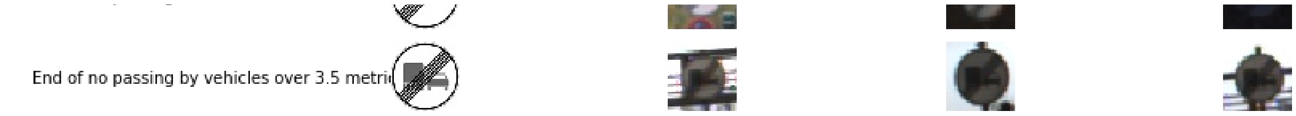
I used the Numpy library to calculate summary statistics of the traffic signs data set:

```
The size of training set is 34799
the size of Validation set is 4410
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
```

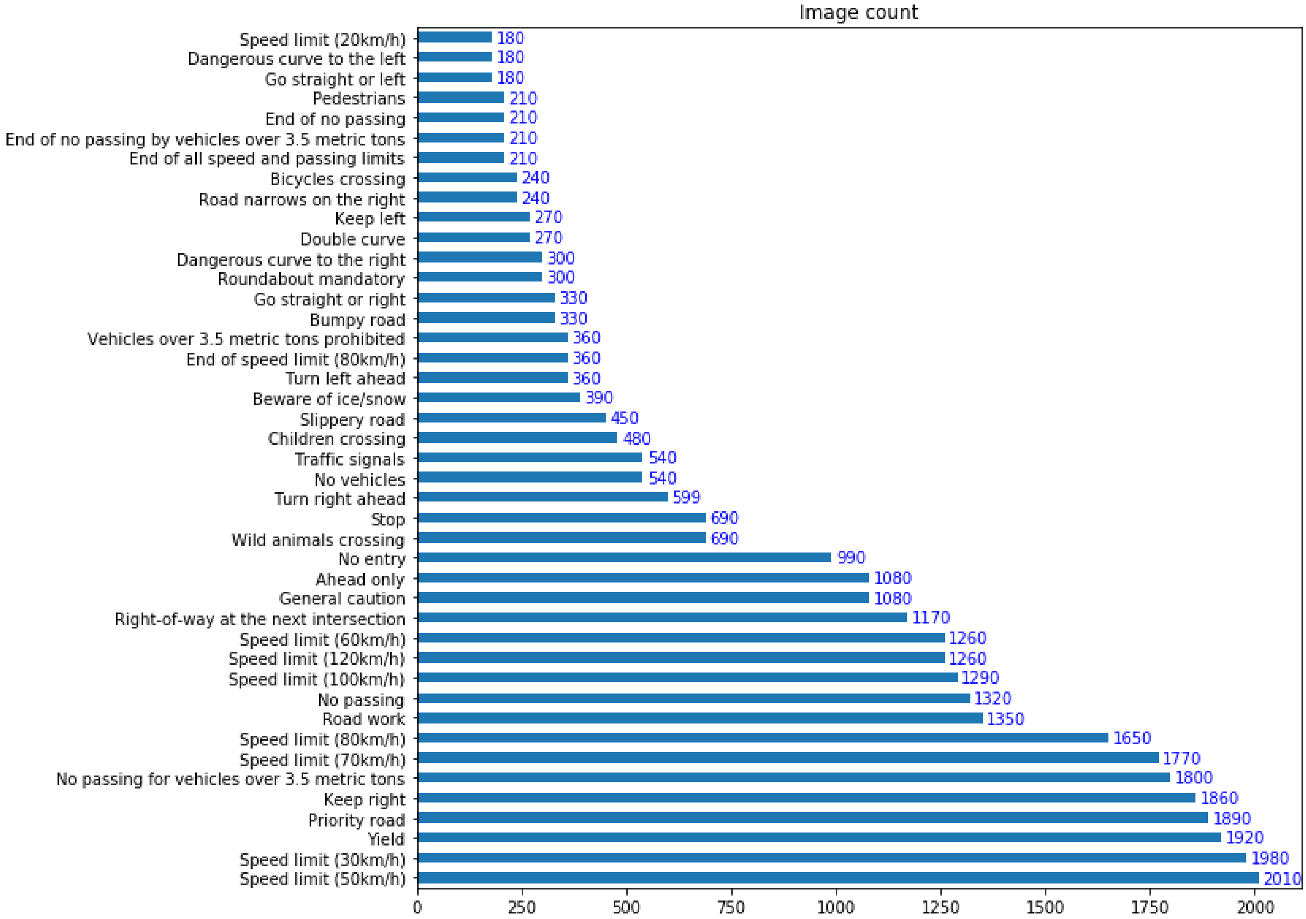
the following figure shows samples of training dataset




Vehicles over 3.5 metric tons prohibited
No entry
General caution
Dangerous curve to the left
Dangerous curve to the right
Double curve
Bumpy road
Slippery road
Road narrows on the right
Road work
Traffic signals
Pedestrians
Children crossing
Bicycles crossing
Beware of ice/snow
Wild animals crossing
End of all speed and passing limits
Turn right ahead
Turn left ahead
Ahead only
Go straight or right
Go straight or left
Keep right
Keep left
Roundabout mandatory
End of no passing



Using Pandas Liberary, Histogram of Training data is plotted to give intuation on overall distrpution

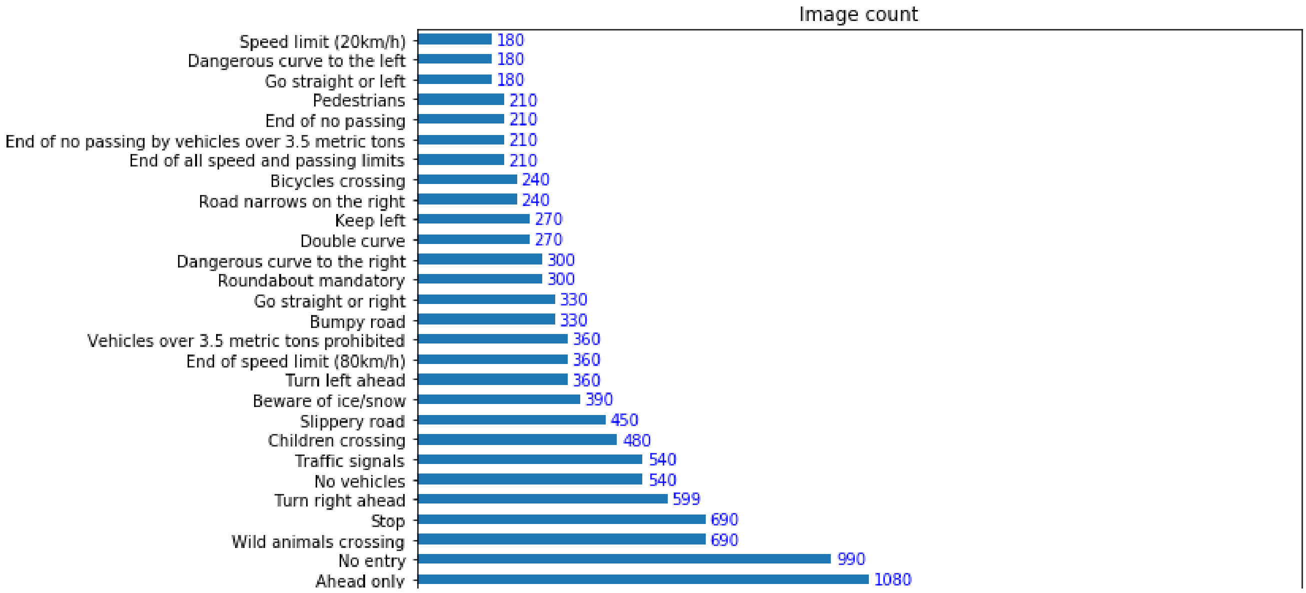


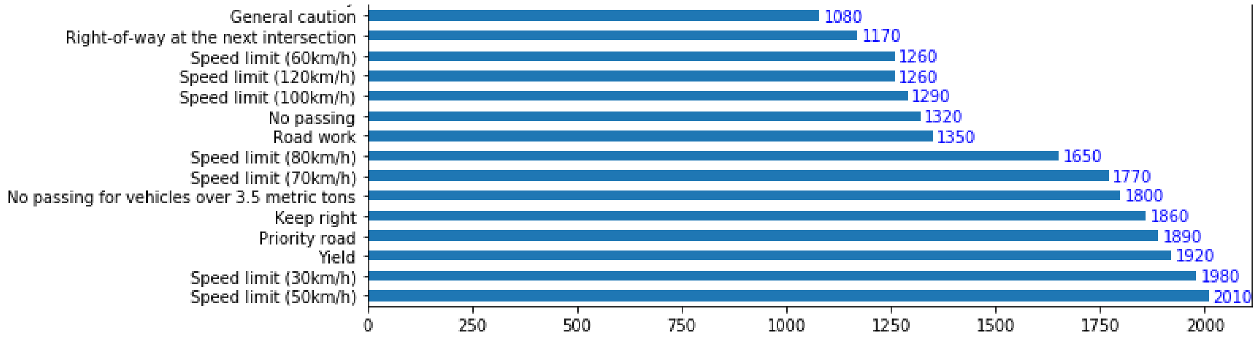
The code for this step is contained in the sixth code cells of the IPython notebook.

Then i regenerated training dattaset,I almost doubled the total samples from original samples to total 102807 samples. You can see some rotated images are showing. It will take care some cases like camera shaking or place at not perfect angle. For future work, maybe can add tilt, warp and shift to the image. The dataset can be easily grow 5-10 times bigger.

I am using 80:20 split on total dataset to get testing set samples.Then split the training set 75:25 again to get training set 61683 samples and validation set 20562 samples. Even in validation set, the lowest image count per class (such as stop sign) is over 200, greater than "30 rule", I am ok to proceed with these setting.

Figure shwing Histogram of regeenrated dataset





Design and Test a Model Architecture

Dataset Preprocessing

shuffled the training dataset

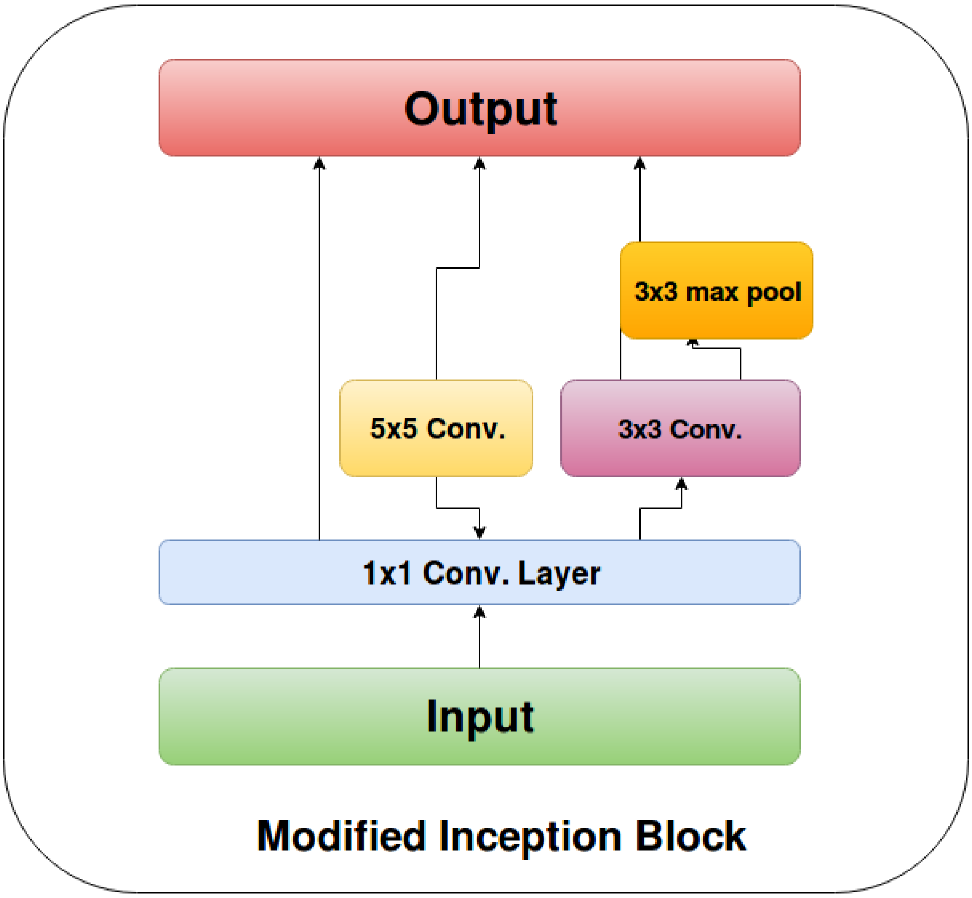
Model Architecture

my solution is inspired by [1], mainly using spatial transformater networks[2] to nullify any disturtion in traffic sign image due to translation, rotation or even contrast variation.

after that using inception model used in GoogleNet[3]. for feature extraction and classification. the diffrence between purpoced solution here and in [1] is that this purposal tries to minimize model arch. by using fewer layers to optain high accuracy.

Localization Network I used LeNet network as my loclization network to learn affine transformation parameters. also add dropout layer to prevent overfitting in training phase.

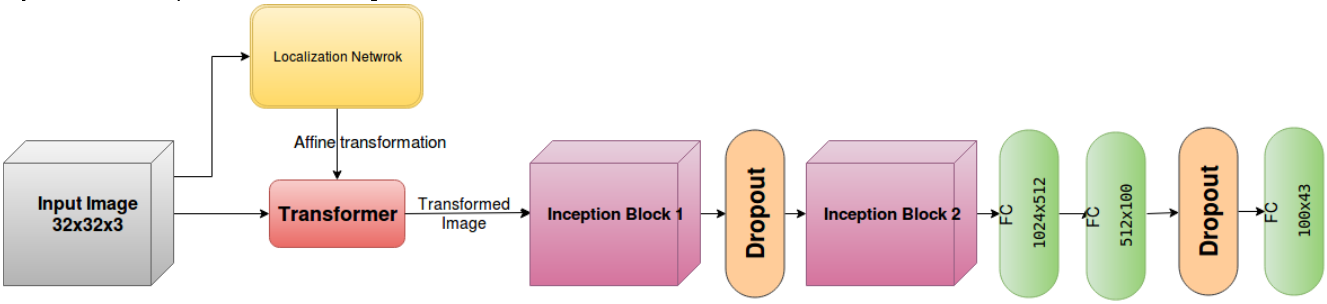
Inception Block the next figure visualize optimized inception block used as key element of my network



Layer	Description
(1) Input	32x32x3 RGB image
(2) Convolution 1x1	3x3 stride, same padding, outputs 11x11x24
(3) RELU	
(4) Convolution 3x3	1x1 stride, input(1) outputs 11x11x16
(5) RELU	

Layer	Description
(6) Convolution 5x5	1x1 stride, input(1) outputs 11x11x8
(7) RELU	
(8) Max pooling 3x3	1x1 stride, input(4) outputs 11x11x16
(9) concatnate(2,4,6,8)	outputs 11x11x64 feature maps

My final model is presented in next figure:



Layer	Description
(1)Input	32x32x3 RGB image
(2) Spatial Transformer	LeNet Network, outputs 32x32x3 transformed image
(3) Inception 3a	input 32x32x3 RGB image, output 11x11x64 feature maps
(4) Dropout	Keep Probabilit = 0.5
(5) Inception 4a	input 11x11x64 RGB image, output 4x4x64 feature maps
(6) flatten	output 1024 feature array
(7) Fully connected	output 512
(8) RELU	
(9) Fully connected	output 100
(10) RELU	
(11) Dropout	Keep Probabilit = 0.5
(12) Fully connected	output 43 class score
(13) Softmax	output 43 class probabilites

The code for training the model is located in cells [27-30] of the ipython notebook.

After tunning, the below parameters were found to yield the best results:

- Learning rate : 0.0001
- Batch size : 128
- Epoch count : 50
- Keep probability for Loclization network : 0.4
- Keep probability for feature maps : 0.5
- Keep probability for fully connected layer : 0.5

My final model results were:

- training set accuracy of 99.7 %
- validation set accuracy of 99.5%
- test set accuracy of 99.4667%

Test a Model on New Images

Here are ten German traffic signs that I found on the web:

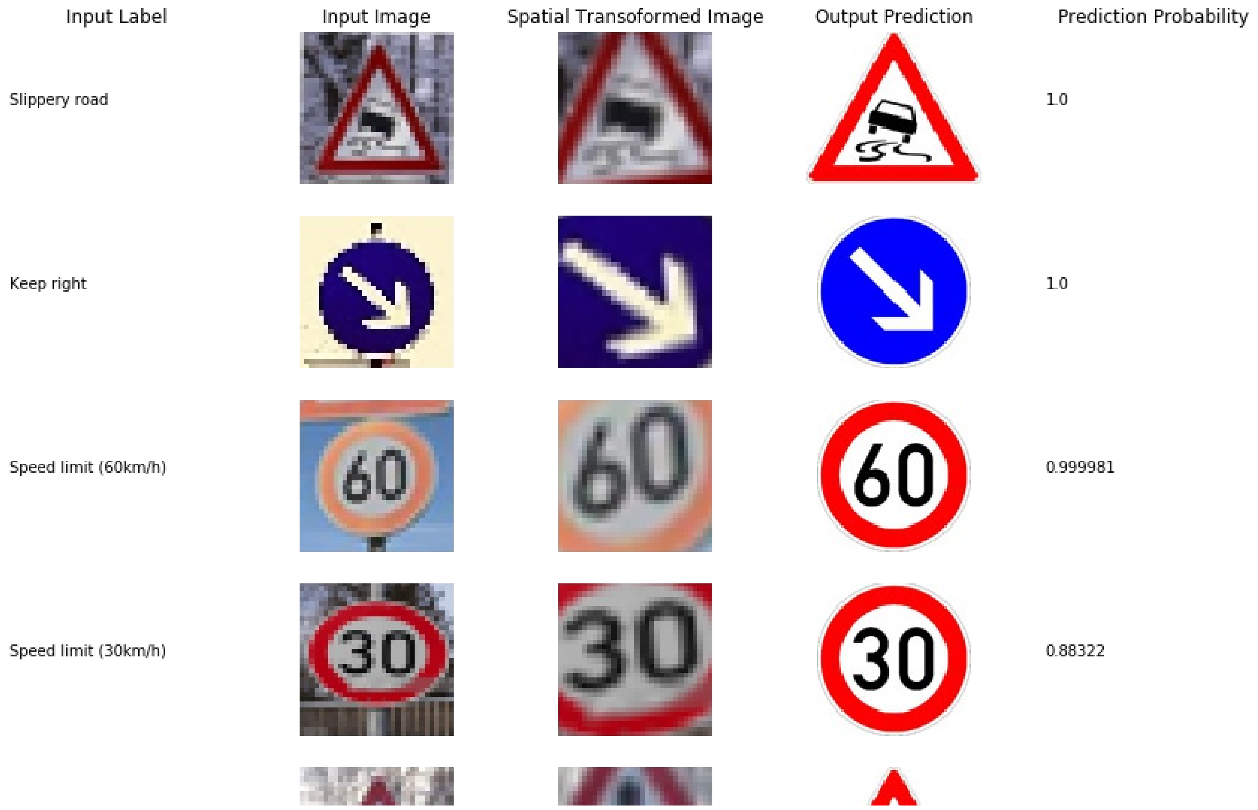
Slippery road

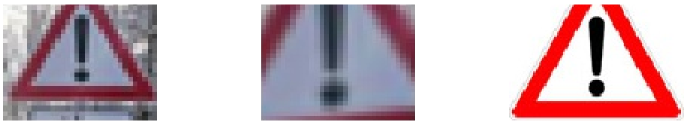
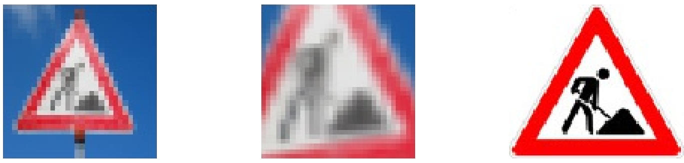
General caution

Notes about test images:

- Turn left ahead sign is very blury and sky background of sign almost has same color of sign itself
- Keep Right sign has a very high contrast
- Slippery road sign is very distorted
- gaussian noise is manually add to Speed limit 70 km/h

The code for making predictions on my final model is located in the 38th cell of the Ipython notebook. Here are the results of the prediction:



General caution		1.0
Right-of-way at the next intersection		1.0
Road work		1.0
Priority road		1.0

The model was able to correctly guess mostly all test traffic signs, This compares favorably to the accuracy on the test set of 99.466 %

Comments on Preformance of small test set

- Spatial Transformer Network was successfully focus on the spacific reagion of interset and asly denoising most of images.

this visualization can help understand basic features learnt by network to prediect its outputs, aslo help detecting of overfitting if happen

Further Improvement

1. Incease complixty of inception block by design a separet 1x1 conv. layer be each path of inception block instead of only single 1x1 conv for all
2. Making Network model deeper by using more inception blocks

Refrences

1. Mrinal Haloi 2015 "[Traffic Sign Classification Using Deep Inception Based Convolutional Networks]". arXiv:1511.02992
2. Max Jaderberg and Karen Simonyan and Andrew Zisserman and Koray Kavukcuoglu 2015 "[Spatial Transformer Networks]". arXiv:1506.02025
3. Christian Szegedy and Vincent Vanhoucke and Sergey Ioffe and Jonathon Shlens and Zbigniew Wojna 2015 "[Rethinking the Inception Architecture for Computer Vision]". arXiv:1512.00567
4. https://github.com/daviddao/spatial-transformer-tensorflow


```
In [ ]:
```