Traffic Sign Classification

Traffic Sign Classification is a multiclass image classification project. I have built a deep learning model to classify Traffic Sign images in to 43 categories. I have used Keras Deep Learning Library to Build Architecture, Train model and for Data Augmentation.

For Training, Validation and Testing, I have used 'German Traffic Sign Dataset'. Additional Five Images for Testing is available in the 'five_test_images' folder.

All the coding for this project is available in 'Traffic_Sign_Classifier.ipynb' file.

Dataset Exploration

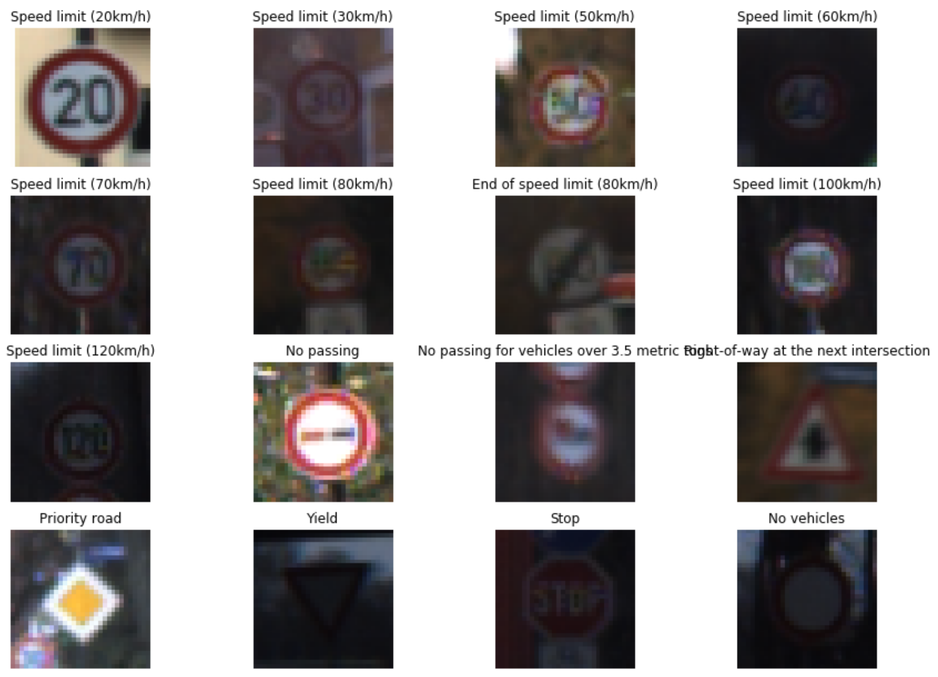
Dataset Summary

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

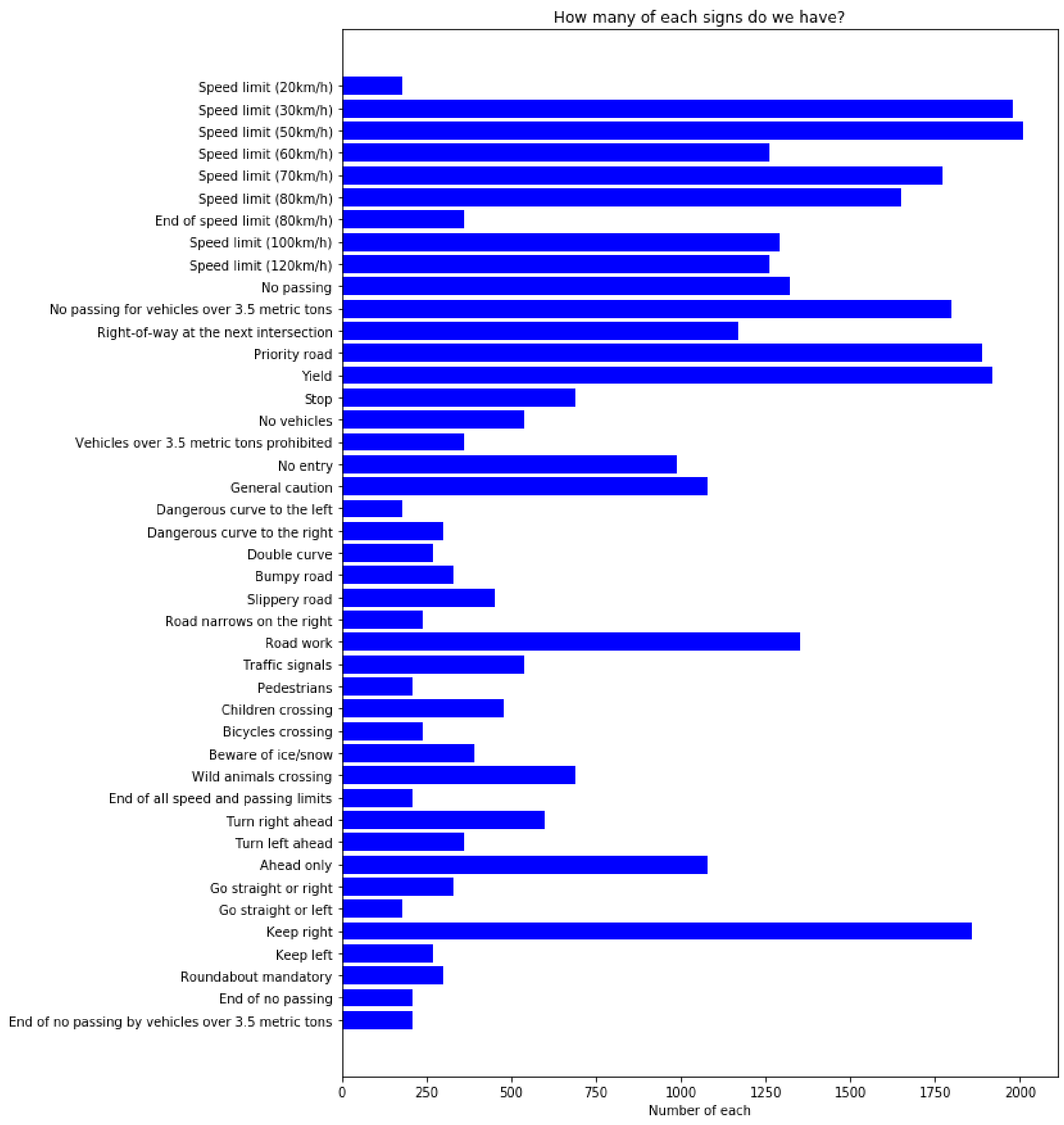
Exploratory Visualization

I have shown 43 images of each Traffic sign from Training set in the notebook file. Few of them are shown below.

Here I have shown graph of few images with labels,



I have also plotted Graph which contains count of images of each category, this help us to understand the distribution of categories inside Training set,



Design and Test a Model Architecture

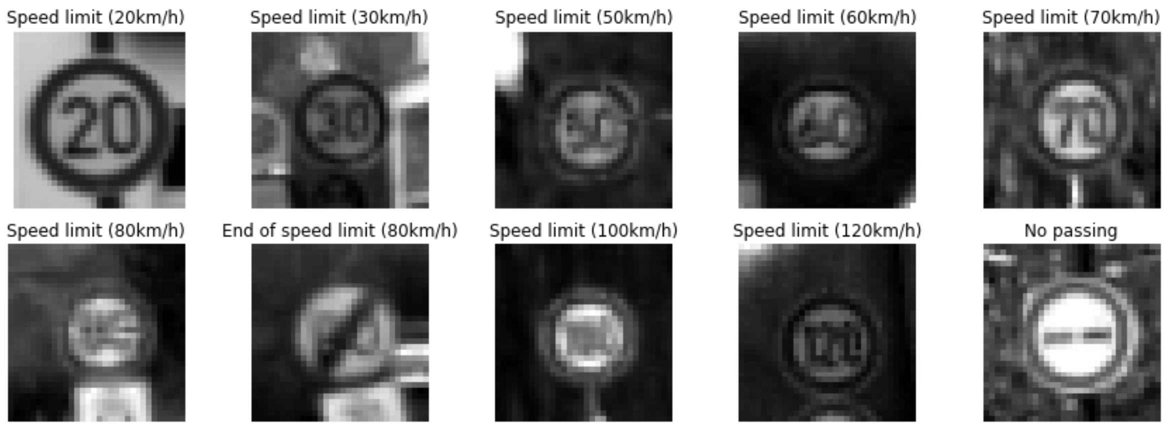
Preprocessing

1. My Keras model don't understand Direct Integer labels so I have one hot encoded labels of all datasets:

2. We don't need colour image to identify traffic signs so we convert images in the all dataset to grayscale.

```
RGB shape: (34799, 32, 32, 3)
Grayscale shape: (34799, 32, 32, 1)
```

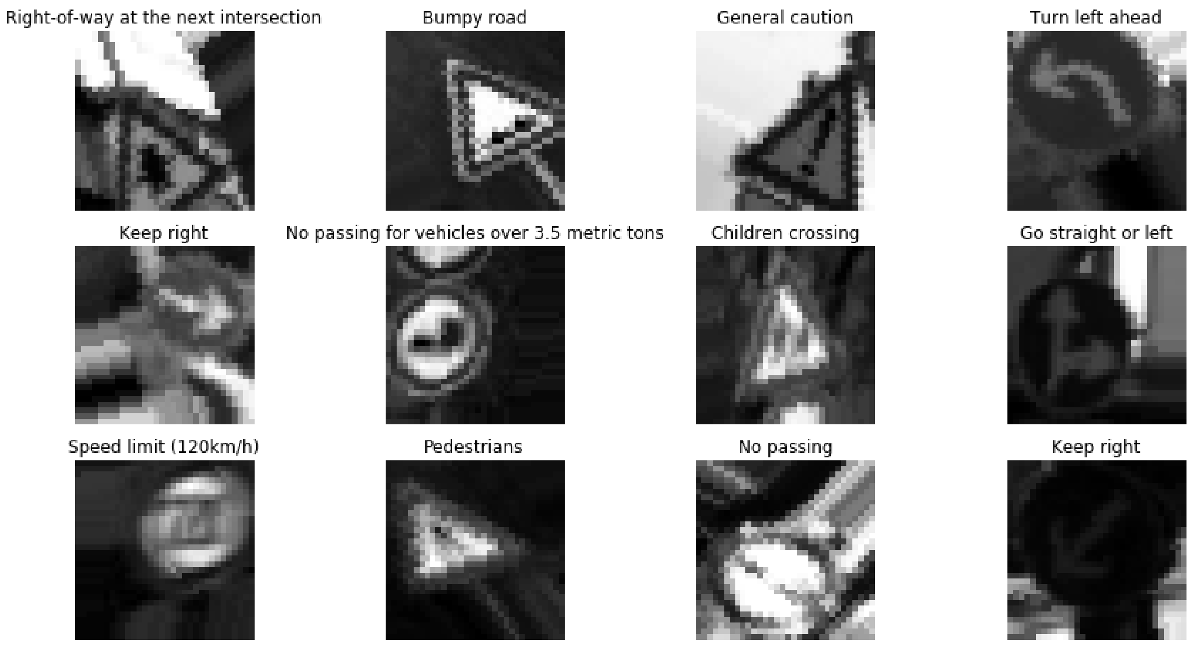
Images of Training Set After Gray scale conversion:



3. As mentioned in the notebook file, datasets had mean value around 80, so Normalization was necessary.

4. Augmenting the training set might help improve model performance. Common data augmentation techniques include rotation, translation, zoom, flips, and/or colour perturbation. This Image Augmentation can be done using Keras library, where ImageDataGenerator class is used to perform different image Augmentation Operations

Images of Training set after Data Augmentation:



Model Architecture

This Model Architecture is build using Keras library. It is based on VGG16 pre-trained model of Image Classification.

It has less number of layers compare to VGG16 because we need to keep less numbers of layer for small images like (32, 32) px images that the reason, VGG16 must have input image more than (48, 48) px.

Here is the Architecture of the Model,

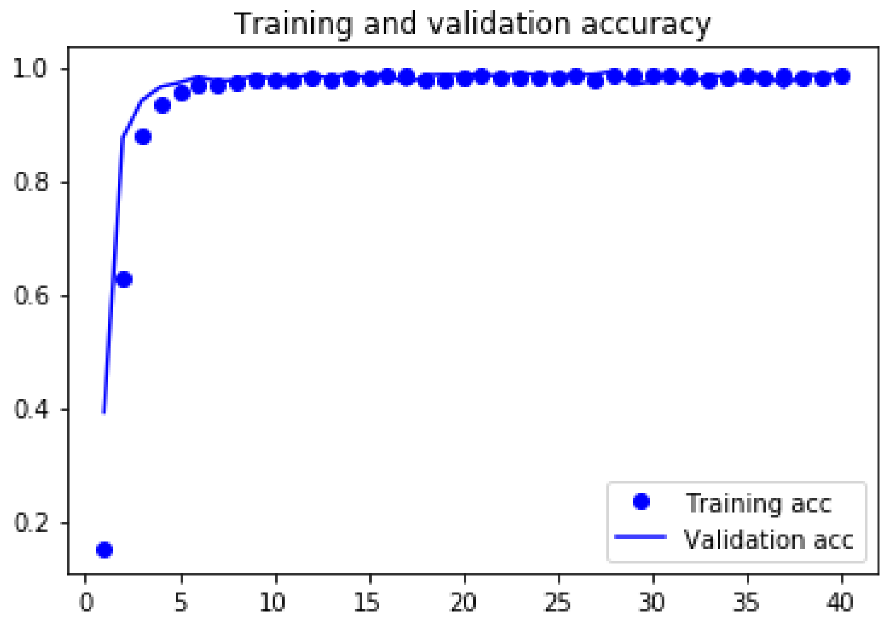
As You can see we have 8 convolution layers, 4 max pooling layer, 1 Dropout, 1 Flatten and 3 Dense Layers, All the Dense and Convolution layer has Relu Activation Function except last layer, last one has Softmax Activation function.

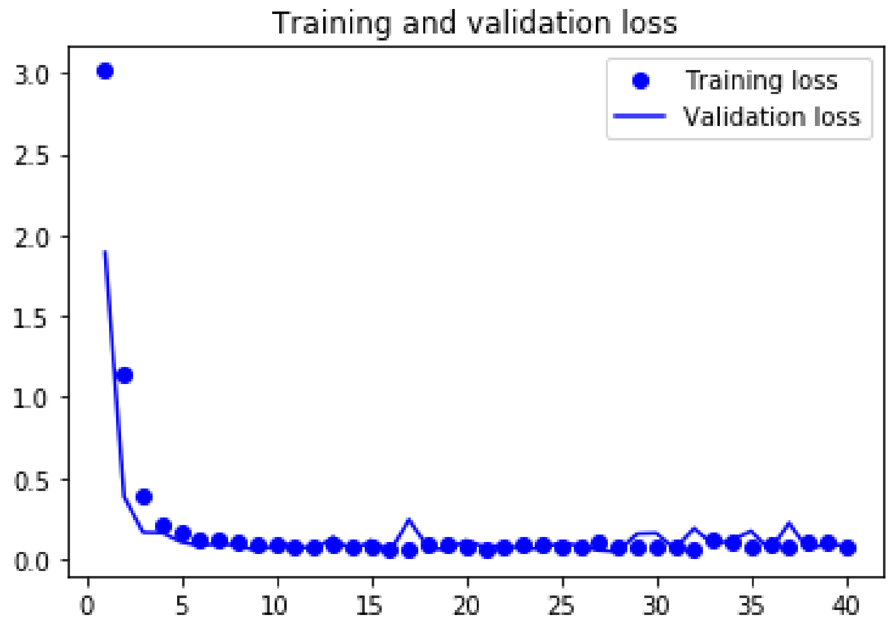
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 16)        160
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        9280
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 256)       147712
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 256)       590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 256)         590080
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 256)         590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 256)         590080
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 4, 4, 512)         1180160
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 512)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2, 2, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               524544
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 43)                5547
=================================================================
Total params: 4,260,619
Trainable params: 4,260,619
Non-trainable params: 0
```

Model Training

I have used

1. Adam optimizer with learning rate 0.001
2. This is multiclass classification so loss function is 'categorical_crossentropy'
3. Our Objective is to increase accuracy of prediction so I have used ‘acc’ as matric parameter
4. Epochs: 40
5. Batch Size: 128





Solution Approach

Earlier I was using LeNet but got accuracy around 85% with all the Preprocessing steps, Model was under fitting, So I Increased total number of layers with increased dimensions in each layers.

Earlier I was using RGB Images, it was working fine but why should we use 3 dimension, when one 1 dimension is enough for us so, I used gray scaled images.

I was earlier using Horizontal Flip from Image Augmentation, I have removed Horizontal Flip from Image Augmentation part because I think our model was misguided when signs like right or left turn came as training data. Model Validation Accuracy jumped from 94% to 98%

Test a Model on New Images

Acquiring New Images

I have used with two different approach to test Images:

1. Using Test-Dataset
2. Using Five Completely New Images from Web

Performance on New Images

On the Test Dataset I got test accuracy around 97%

On the Five Images, the model predicted 5 out of 5 signs correctly, it's 100.0% accurate on these new images

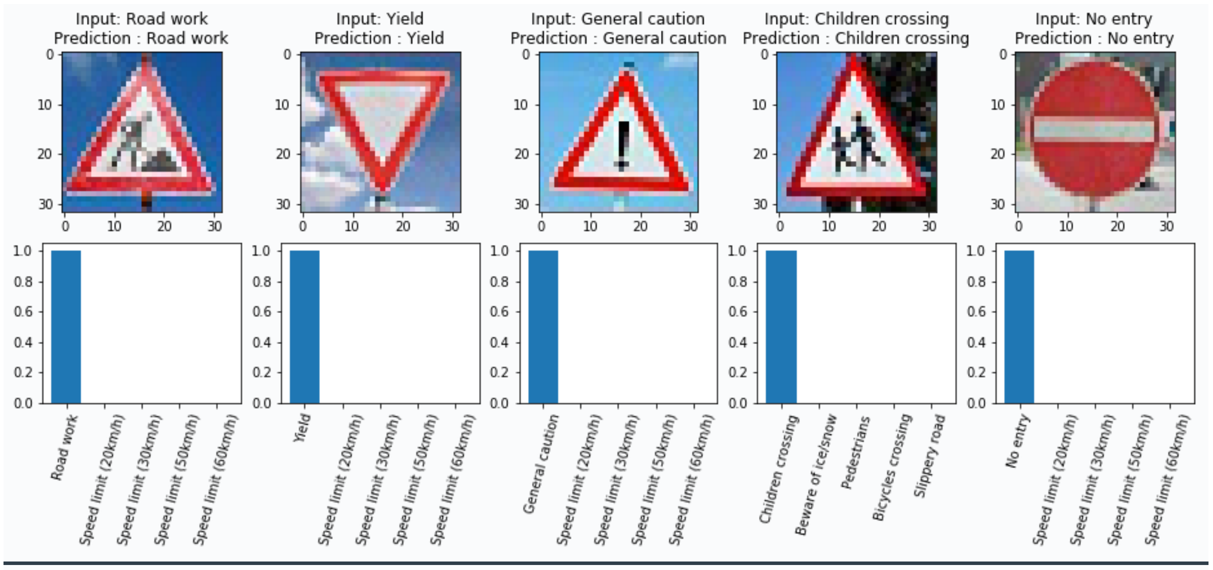
Inputs:

Predictions:

Model Certainty - Softmax Probabilities

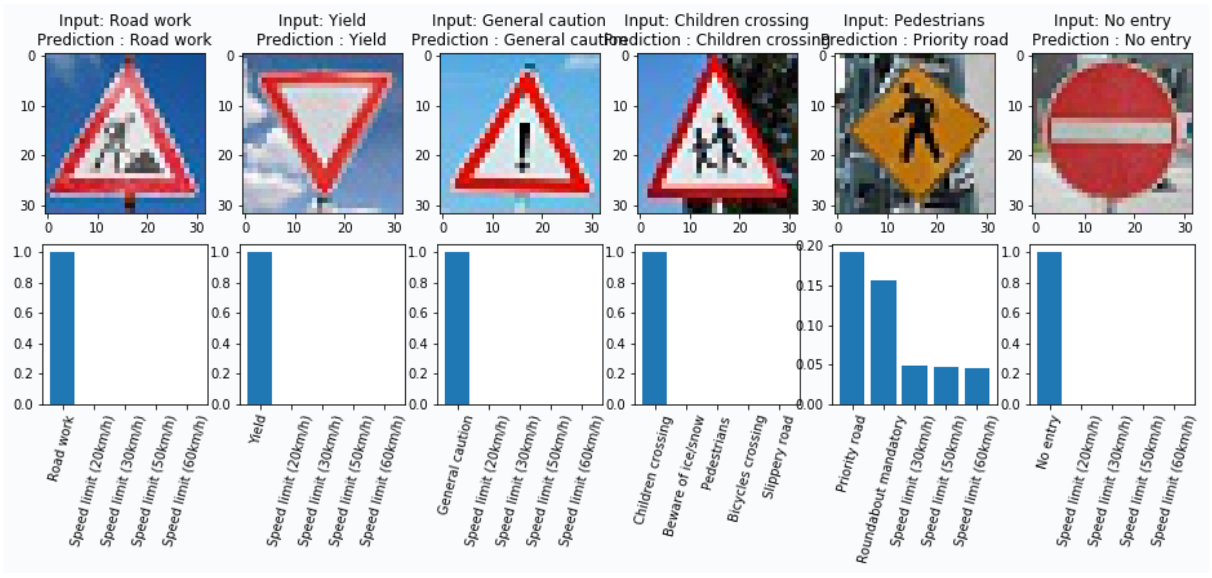
For the five images I have found top five Softmax probabilities, ‘model. predict(image)’ was taking image as input and returning array of Probabilities of for all the 43 classes. I have filtered top five classes and Represented it on bar chart with their respective probabilities

This graph shows, input prediction and Bar- chart of top five probabilities,



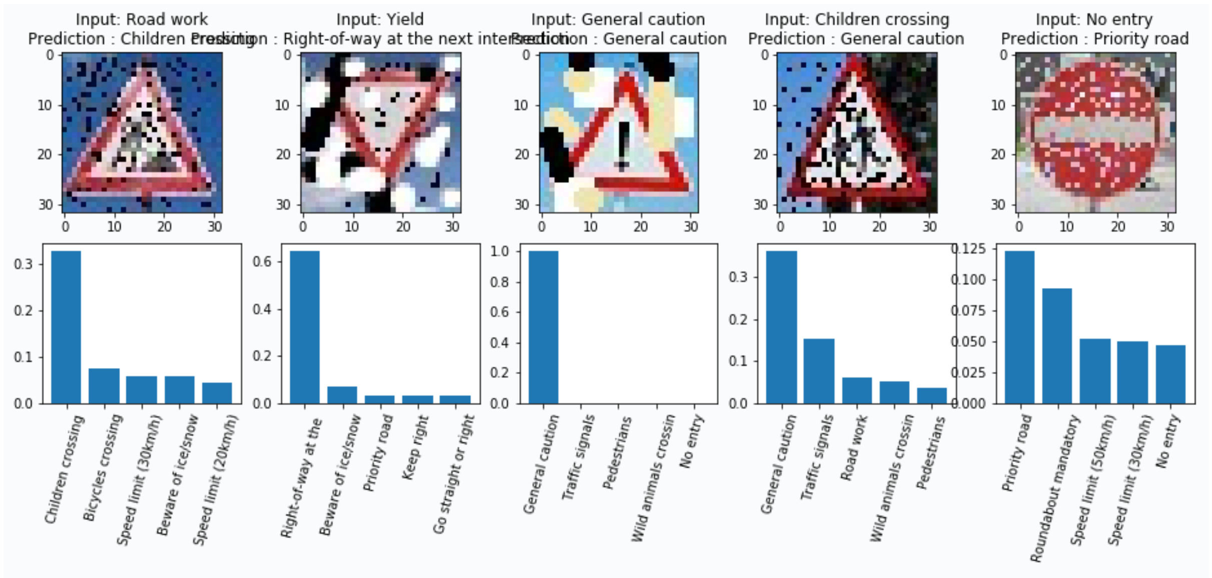
I have notice that for Pedestrian Traffic Sign, our model doesn't show good prediction,

Here is the graph,



When we apply many Very high noise content as shown below model gets bit confused which is obvious because even us can't identify Such sign if such sever noise is in image. Even though such worse images, model is able to show one right prediction.

Here I have shown bar chart as for very high noisy data just to check top five prediction.



Challenges:

1. I was facing problem while finding probabilities, I was aware that **'model. predict(image)'** Is returning prediction but I didn't know that it returns array of predictions. Fortunately, when I run it, I got all the probabilities.
2. I don't want to define functions for Image Augmentation, Keras ImageDataGenerator class made it easy for me.
3. To increase validation and prediction accuracy I was trying different things like, '**I Increased total number of layers with increased dimensions in each layer', 'I was earlier using Horizontal Flip from Image Augmentation, I have removed Horizontal Flip from Image Augmentation'**

References:

I have used Five images which is downloaded from google photos, I manually cropped this images using paint up to (32, 32) px.