Traffic Sign Recognition

Overview

For making an SDC(Self Driving Car) make decisions on traffic signs as humans while driving we need data of the camera feed, but for making decisions we first need a way to detect and classify the traffic signs, there are many traditional ways of doing this task using image processing and computer vision techniques, using a neural network to make a classification, using deep learning techniques like CNN, etc.

Traditional ways for traffic sign recognition using image processing is computationally expensive and is not suitable for applications that need an accurate prediction stats, it is hard to make them adapt to the present day camera technology that involves generating high-resolution images, a large number of frames per second(FPS).

Normal neural networks don't take into account the local relationships of each pixel in the image to the surrounding pixels because in normal NN's we are just flattening the image and feeding to a NN so each pixel is connected to a neural in the next layer independently, but we need a way to maintain this relationship among pixels. And also using normal NN for tasks that include images increases the number of parameters, so it computationally expensive to train the network. Neural networks also may not adapt well to high FPS video feed.

The convolutional neural network takes into account the local relationship of pixels in the image, aims to decrease the number of parameters by sharing the parameters among the pixels by using a concept of kernels and strides. CNN is also good at the high fps video feed.

So, by the above analysis, CNN works better for the traffic recognition task .

Goals

1. Load the data set (see below for links to the project data set).
2. Explore, summarize and visualize the data set.
3. Design, train and test a model architecture.
4. Use the model to make predictions on new images.

5. Analyze the softmax probabilities of the new images.
6. Summarize the results with a written report.

Dataset Summary and Exploration:

Dataset Exploration , I have used python's pickle library to load the dataset from the files "train.p" , "test.p" , "valid.p" , pickle is a python library that gives us the flexibility to serialize the data in the form of python dictionaries ,while loading the data from these files as if we are accessing data from a python dictionary i.e., through keys.

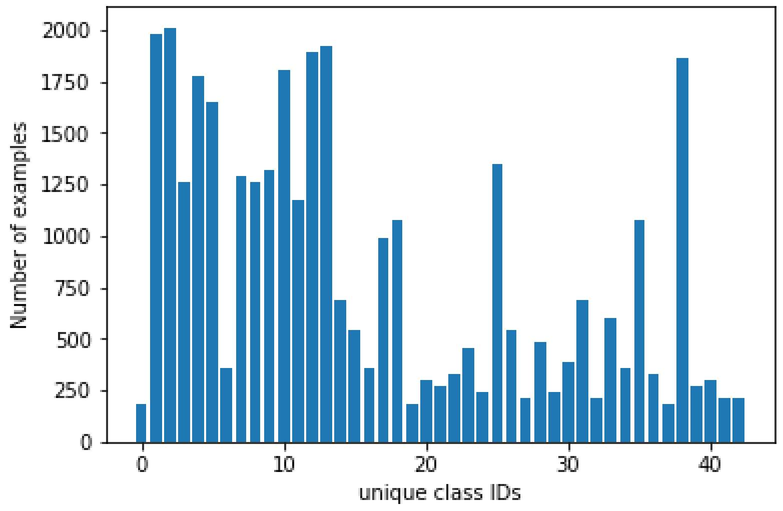
Pandas and numpy are great tools for exploring the dataset, Pandas can deal with relational data , numpy can deal with numerical data like images, etc. Pandas is used for extracting number of unique classes in the training dataset , turns out that there are 43 unique classes in the training set and also this also is the number of classes for our classification.

Following picture shows some of the images sampled randomly from the training dataset along with their respective id's. Each image is of 32X32 (height, width) size with 3 channels RGB.

There are 34799 Number of training examples , 12630 number of testing examples , 43 number of classes , 4410 number of validation data examples.

Following bar graph displays the distribution of images per class , this type of exploration gives us the evidence on what type of evaluation metrics to employ inorder to evaluate the model performance.

There are some classes with relatively large number of images concentrated , there are alos some classes with least number of images concentrated, so it is enough look at accuracy based model evaluation, but it would also be better to evaluate the model using precision , recall , f1-score because they provide how certain our model in classification per class so we can investigate into what examples are missclassified and why they are causing model to missclassify.



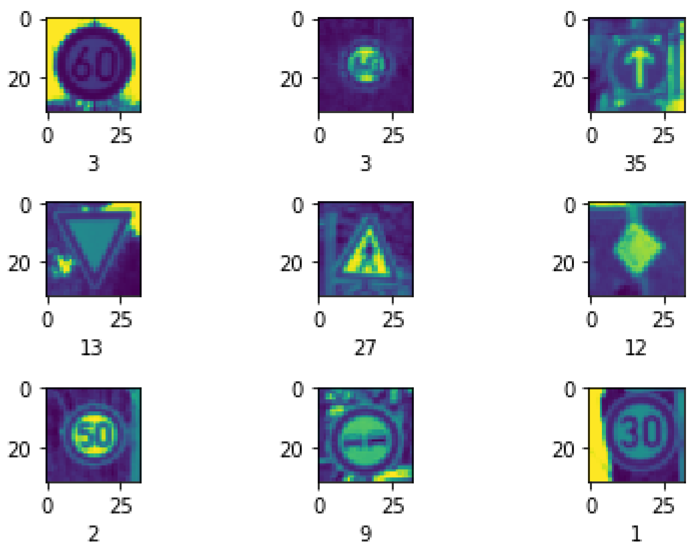
Design and Testing Model Architecture

Different color space representation of images have different regions in image that are enhanced , this may make the model to learn features of the image that are helpful to classify image into its labelled class very fast .

Inorder to make our model more robust in classifying images in different conditions of image we may augment the dataset using some of the image processing techniques that doesnot remove main features from the image but just adds some small aberrations that indirectly makes network to learn even some more in depth features that are required to classify.

A “Preprocess” class has been defined with methods that can change color space of the image to a desired color space dynamically , method to add blurring to the images, method to rotate the image to certain degree, method to shift center of the image to some point in the 32X32 grid of points.

Following image shows the gray scaled converted version of images in the training dataset. (Even though they are gray scaled Matplotlib has added some color to display)



LeNet CNN architecture is being used for Traffic Sign Recognition task. LeNet is a simple architecture with successive convolution layers with some subsampling layers in between (pooling layers).

Layer	Description
Input Image	(None, 32,32,1)
Convolutional Layer	Stride: 1(h,w) , Kernels: 5X5X1 #6, Output:28X28X6
Relu	Applies relu activation on the previous layer feature maps
Maxpooling	Stride: 2, Kernel:2X2 , Output: 14X14X6
Convolutional Layer	Stride: 1, Kernels:5X5X6 #16,Output:10X10X16,
Relu	Applies relu activation on the previous layer feature maps
Maxpooling	Stride : 2, Kernel:2X2 , Output:5X5X16
Flatten	Output: 400
Fully Connected Layer(1)	Input : 400, Output: 120
Dropout	Keep Probability:0.8(train), 1(valid nd test)

Fully Connected Layer(2)	Input:120, output: 84
Dropout	Keep Probability:0.8(train), 1(valid nd test)
Fully Connected Layer	Input: 84, output: 43
Softmax	Probability Distribution of Output Over 43 output logits.

Approaches Took To Train The Model:

Different color representations of images for training the model and performance of these models on validation accuracy is recorded.

Following table shows the different validation accuracies for different models corresponding to different hyperparameter values epochs and batchsize.

Model(on different Color Spaced Images)	Epochs	Batch_size	Validation Accuracy (in %)
Gray Scale Image Model	15	128	93.4
YUV color Space image Model	15	128	88
HLS color space image Model	15	128	90.6
--------------------------	----------------WITH	REGULARIZATION	---------------------------
Gray Scale Image Model	30	128	96
YUV color Space image Model	30	128	93
HLS color space image Model	30	128	94

Initially LeNet CNN architecture is taken as our project CNN architecture , this is with no regularization. This model is tested on different color spaced image models with batch size 128 and epochs 15 , above table shows the corresponding validation accuracies ,gray scale model is doing well on validation data but with only 93.4% . This clearly indicates that it not doing well in images that have never seen that is it not able to do generalize well on these images. So this gives the hint of employing some of the regularization techniques like l2 regularization , Dropout, EarlyStopping.

DropOut is the standard to use in neural network because it depicts the scenario of combining different neural network models performance into one model there by increasing the accuracy of the overall model .

After first and second fully connected layers a DropOut layer is employed that retains each node in this layer with a probability assigned .

This approach plus an increase in number of epochs significantly improved the validation accuracies of the three models. But again Gray Scaled Image out weighs other models with validation accuracy of about 96%.

With just a simple architecture of 2 convolutional layers and three fully connected layers we are able to get an accuracy of about 96%. Because of this accuracy and also the number of parameters in network, depth of the network i think this is a better network to use for this task , rather than a network with much depth this may overfit the data.

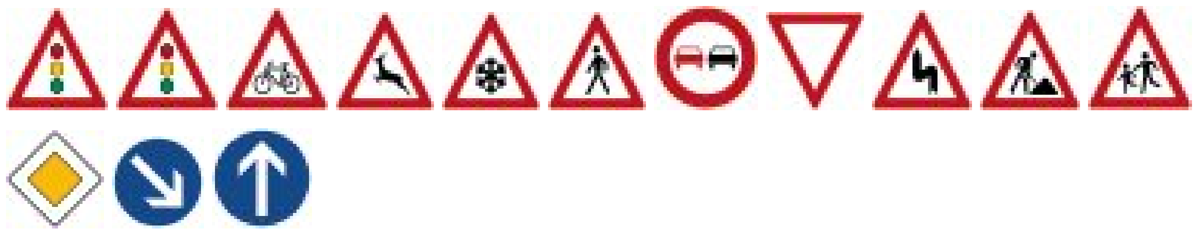
Model Performance On Test DataSet:

Gray Image Model performed really well on the test dataset with an accuracy of about 93%.

Model Performance On Test Images From Web.

(Image were taken from the website https://routetogermany.com/drivingingermany/driving-in-germany)

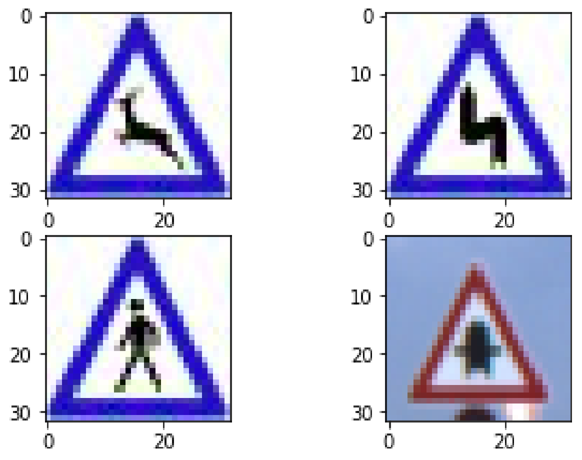
Here are the images taken from the web :



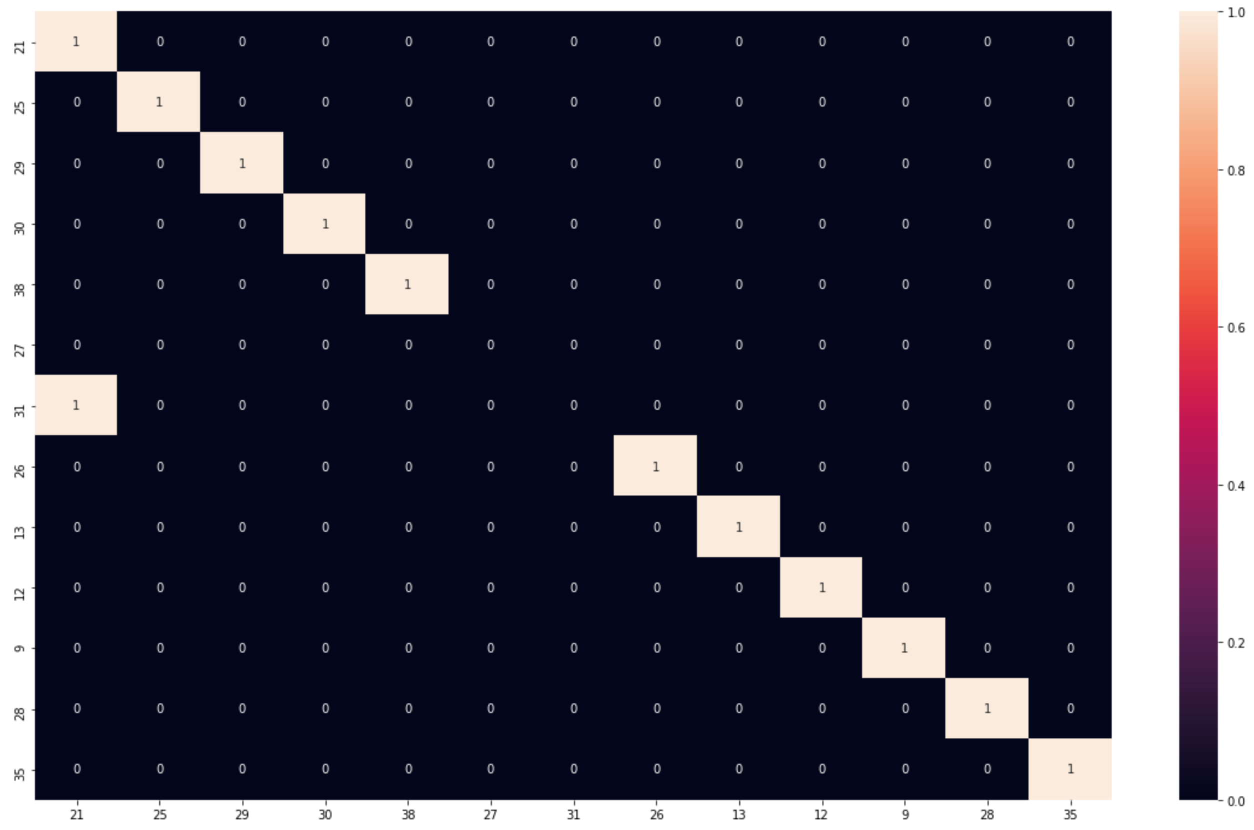
Following Table show s the actual labels and predicted labels.
Two samples misclassified , this makes the accuracy of model on web images : around 85%

Actual	Prediction
Double Curve	Double Curve
Road Work	Road Work
Bicycles Crossing	Bicycles Crossing
Beware of ice /snow	Beware of ice /snow
Keep Right	Keep Right
Pedestrians	Right-of-way at the next intersection
Wild animals crossing	Double Curve
Traffic Signals	Traffic Signals
Yield	Yield
Priority Road	Priority Road
No passing	No passing
Children crossing	Children crossing
Ahead only	Ahead only

The following image shows the comparison of actual pictures that were missclassified into pictures shown on the right.



Following is the confusion matrix of the model on the web images.



Precision and recall of the model on the web images:

	precision	recall	f1-score	support
9	1.00	1.00	1.00	1
11	0.00	0.00	0.00	0
12	1.00	1.00	1.00	1
13	1.00	1.00	1.00	1
21	0.50	1.00	0.67	1
25	1.00	1.00	1.00	1
26	1.00	1.00	1.00	1
27	0.00	0.00	0.00	1
28	1.00	1.00	1.00	1
29	1.00	1.00	1.00	1
30	1.00	1.00	1.00	1

```
         31       0.00      0.00      0.00         1

         35       1.00      1.00      1.00         1
         38       1.00      1.00      1.00         1

avg / total       0.81      0.85      0.82        13
```

Following table shows the top5 probabilities of 5 images in the data collected from the web and corresponding traffic signs.

Top 5 Probabilites per image	Traffic Sign Type
[0.02414097, 0.02323612, 0.02323471, 0.0232347 , 0.0232347]	Double curve,Beware of ice/snow,Road work,Speed limit (20km/h),,Speed limit (30km/h)
[0.0241448 , 0.02323465, 0.02323465, 0.02323465, 0.02323465]	Road work,Speed limit (20km/h) ,Speed limit (30km/h) ,Speed limit (50km/h) ,Speed limit (60km/h)
[0.0238607 , 0.02336429, 0.02324253, 0.02323836, 0.02323831]	Children crossing,Bicycles crossing,Beware of ice/snow,Bumpy road,Speed limit (20km/h)
[0.02414346, 0.02323516, 0.02323467, 0.02323467, 0.02323467]	Beware of ice/snow,Children crossing,Right-of-way at the next intersection,Slippery road,Speed limit (20km/h)
[0.0241448 , 0.02323465, 0.02323465, 0.02323465, 0.02323465]	Keep right,Speed limit (20km/h),Speed limit (30km/h) ,Speed limit (50km/h) ,Speed limit (60km/h)