Project 2: Building a Traffic Sign Recognition Classifier

Build a Traffic Sign Classifier Project

The goals / steps of this project are the following:

- Load the 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 softma probabilities of the new images
- Summarize the results with a written report

1. Data Set Summary & Exploration

Data Set Summary:

####1. Following is the basic summary of the data.

Used Picked, and pickled data is a dictionary with 4 key/value pairs:

'features' is a 4D array containing raw pixel data of the traffic sign images, (num examples, width, height, channels).

'labels' is a 1D array containing the label/class id of the traffic sign. The file signnames.csv contains id -> name mappings for each id.

'sizes' is a list containing tuples, (width, height) representing the original width and height the image.

'coords' is a list containing tuples, (x1, y1, x2, y2) representing coordinates of a bounding box around the sign in the image.

Used the provided German traffic sign data:

training_file = 'trafficSignData/train.p'

testing_file = 'trafficSignData/test.p'

The following are the details about the data read using Pickle:

```
Number of training examples = 39209
Number of testing examples = 12630
Image data shape = (39209, 32, 32, 3)
Number of classes = 43
```

Exploratory Visualization

####2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set.

Here is a bar chart, which shows the traffic sign types and count of the various signs.

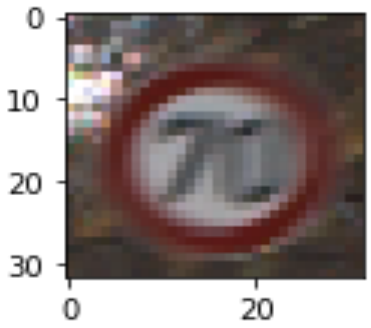
Design and Test Model Architecture

Preprocessing:

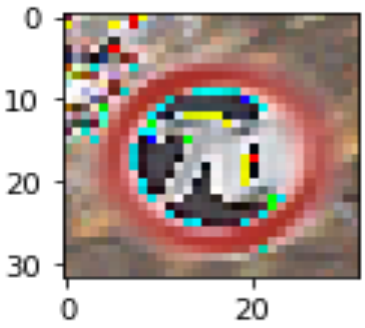
- Shuffle the input data
- Used the existing color images instead of grey scaled images, as the color signs are most relevant in traffic sign classification
- Applied Normalization to be in the range -1 to +1, which provides faster processing of data
    - Reference: http://stackoverflow.com/questions/4674623/why-do-we-have-to-normalize-the-input-for-an-artificial-neural-network
- Tried splitting the training and validating set to 80%, 20% respectively

Below are the Original Image and Normalized image

Original Image

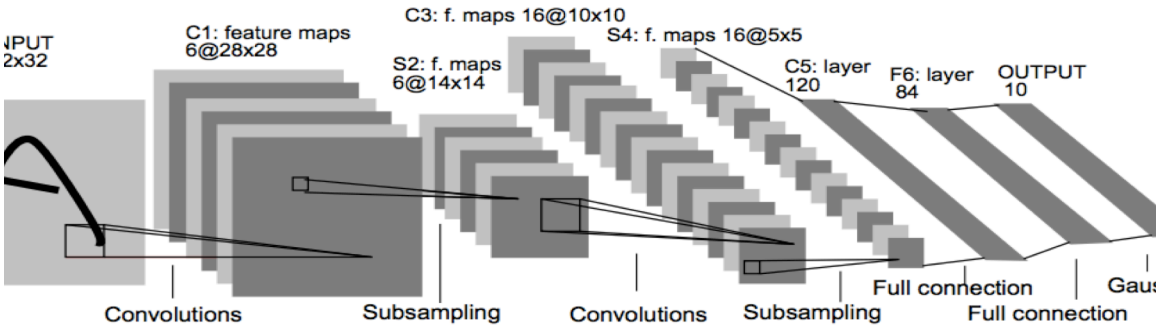


Normalized Image



Model Architecture:

LeNet-5 is a pioneering 7-level convolutional network by Yann LeCun that classifies digits. Several banks used to recognize handwritten numbers on checks digitized in 32x32 pixel images. The same model is used in this traffic sign classification.



Various layers of the model

Input Layer	1
Output Layer	1
Internal Hidden layers	6

Detailed summary of the layer, type, image size, filter size and stride. Padding is VALID

Layer Type	**Image Size**	**Filter Size / Stride**
Input	32 x 32 x 3	-
Convolutional	28 x 28 x 6	5 x 5 / 1
Max Pooling	14 x 14 x 6	2 x 2 / 2
Convolutional	10 x 10 x 16	5 x 5 / 1

Max Pooling	5 x 5 x 6	2 x 2 / 2
Full Connected	120	-
Full Connected	84	-
Output	43	-

Model Training:

The Lenet model was trained using the Adam Optimizer for this traffic sign classifier.

The hyper parameters are listed below.

Parameter Type	**Value**
Beta	0.01
Learning rate	0.0001
Weight/Mu	0
Weight/Sigma	0.1
EPOCHS	20
BATCH_SIZE	128

Solution Approach:

To start with understanding the Tensorflow and CNN took sometime, as this is the first time looking in to neural networks and various tools used including numpy.

The Lenet example helped in getting basic understanding and to see how to use the convolutions neural networks and tensor flow.

Tried to do the incremental approach in determining the correct filter size, filter stride and modeling the CNN.

Initially had the beta .01 and tried the Adam optimizer 0.01 and the accuracy not upto the expected level. Then tried reducing the the learning rate 0.001 and slowly to 0.0001 improved. Finally able to get 0.985 accuracy with 0.0001

Snippet of the actual run…

```
EPOCH 17 ...
Validation Accuracy = 0.983
EPOCH 18 ...
Validation Accuracy = 0.982
EPOCH 19 ...
Validation Accuracy = 0.983
EPOCH 20 ...
Validation Accuracy = 0.985

Model saved
```

Test a Model on New images

Acquiring New Image:

Here are five German traffic signs that I found on the web, resized it to 32x32 and created a NewImages.p format.

Performance on New images:

####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).

Images #1, #2

The first 2 signs road closed and crossing signs not in the provided CSV files, and the Lenet model also did not detect those images as expected.

Image #3

This is the pedestrian sign, in the CSV file its #27 and it was detected

Image #4

This is the Men and work sign, in the CSV file its #25 and it was detected

Image # 5

This is the Left turn sign, in the CSV file its #34 and it was detected

```
5  Accuracy on the new images = 0.600
```

The model was able to correctly guess 3 of the 5 traffic signs, which gives an accuracy of 60%. Rest of the 2 images are not detectable as its not part of the original training set.

Got the accuracy of the 5 new images = 0.600 , the first 2 images are not in the provided CSV. The other 3 images are detected that comes to correct result of 60%.

Here are the results of the prediction:

Image	Sign Type	CSV #	Predicted
1	Road Closed Sign	Not available	Not predicted
2	Train Crossing Sign	Not available	Not predicted
3	Pedestrian Crossing sign	27	Predicted
4	Mean at Work Sign	25	Predicted
5	Left turn Sign	34	Predicted

Model Certainty – Softmax Probabilities:

Here is the softmax probabilities for the 5 new images got from the web:

Top_k probabilities stored in 'top_k_probs' variable.

Displaying all the 5 entries...

[[9.99927640e-01 6.08194168e-05 9.69805933e-06 1.65395113e-06
1.41554892e-07]

[9.47582543e-01 5.20880707e-02 1.37308598e-04 1.23060090e-04
5.49632859e-05]

[9.98363912e-01 1.60616252e-03 2.28319805e-05 7.10072754e-06
6.26630969e-10]

[9.98239636e-01 1.75990677e-03 4.54889232e-07 7.57175378e-10
2.39843284e-11]

 [9.99703348e-01 2.80096283e-04 1.65568290e-05 1.19498147e-10
1.04988324e-10]]

 [[13 23 40 9 20]

 [37 40 23 10 21]

 [27 18 24 26 25]

 [25 30 31 21 23]

  [34 11 18 12 38]]

Predicted as Pedestrian : 99.8%

Predicted as Road Work : 99.8%

Predicted as Turn left ahead :99.7%

IMAGE NOT IN CSV.

IMAGE NOT IN CSV.

Future Work:

- This project gave me a good learning of basic of LeNET/CNN and Tensor flow usage,
- Work on exploring different training model
- Familiar more on Tensor flow, Numpy and plotting
- Try to use my image library to use it for facial recognition, using CNN model