Traffic sign Recognition

Goal of this exercise is to create Convoluted neural network that will help in identifying Traffic signs. Model will learn(supervised) with sample dataset collected from German Traffic signs.

Relevant files

Code implementation: Traffic_Sign_Classifier.ipynb

Model Saved File: lenet.meta, lenet.index, lenet.data-00000-of-00001, checkpoint

New Images directory: ./ German_5_images

Writeup: Current file

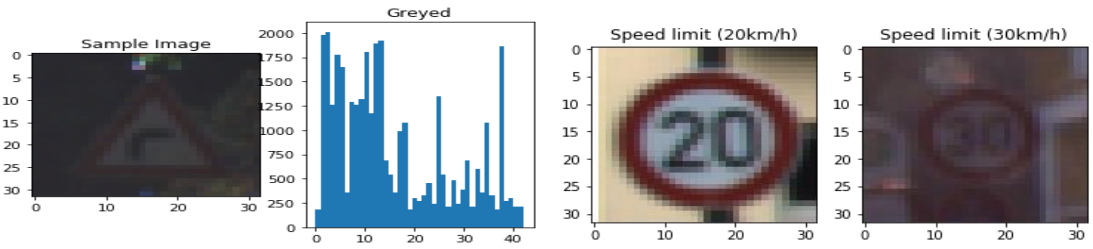
Dataset Summary & Exploration:

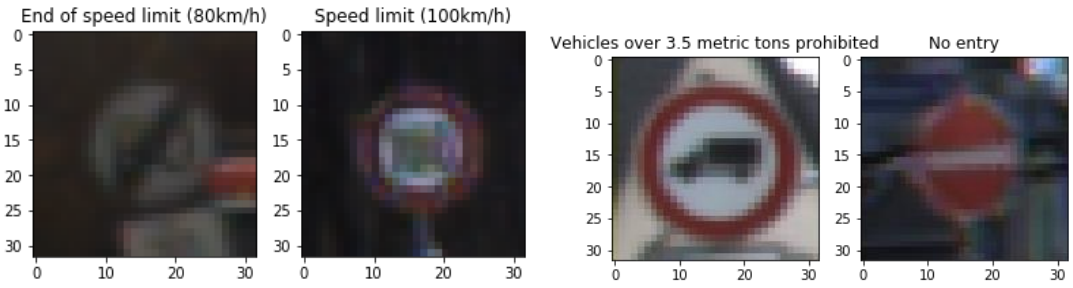
There were two sample dataset collection used for training and testing. Training dataset will be split into 80/20 ratio for training and validation steps. Each dataset(train & test) consists of 34,799 sample images along with their classification codes.

Based on profiling, below are few observations

1. There are 43 unique images available in both datasets. List of Unique 43 classification codes are also available. As 43 images were used in 34,799 sample set and frequency distribution of all images are not near equal. Therefore, data augmentation might help model.
2. Upon Random sampling of training set, it was also found images' contrast are not consistent.
3. All image collected were 32x32x3 in size.

Below are histogram of training dataset distribution and sample images from training dataset





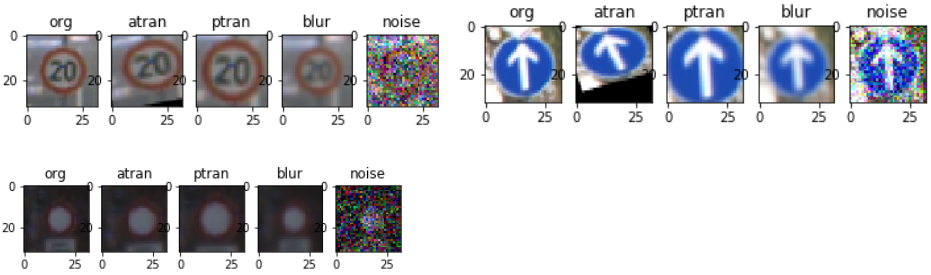
Preprocessing

Profiling result led to following preprocessing steps

1. ***Data augmentation*** done to improve the data versatility in images. Augmented images are based on following techniques
    a. Geometric image transformation techniques like Affinity transform, Perspective Transform and Image rotation were utilized. OpenCV python package was used for processing these image transformation
    b. Blurring image and Noise into image techniques were implemented for improving model.

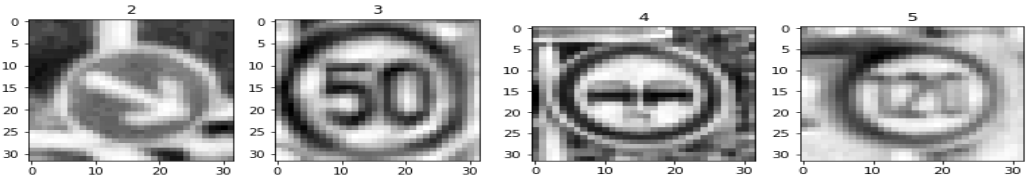
    In above techniques, it was found that *Perspective transform* helped to achieve high accuracy in validation. Therefore, functions for other technique mentioned above are implemented in code, however, not utilized.

    After augmentation, in total 69,598 training dataset available for model. Below are images after applying 5 techniques mentioned in augmentation step



2. ***Preprocessing***: Few preprocessing steps applied before running it through model. Techniques like
    a. *Gray scaling* - to ignore color as decisive factor Convolution neural net
    b. *Normalization* - Min Max Normalize for numerical stability
    c. *Adaptive Histogram equalizer* - to adjust the contrast locally with filter size of 5x5.

Below are image snapshots after pre-processing step.



Model Architecture

Lenet Architecture was designed for recognizing Traffic signs. There are total 5 layers adapted consisting of

- 2 layers of Convolution, Activation and Pooling,
- 2 layers fully connected and activation
- 1 layer fully connected layer with logit generation.

Layer	Component	Input	Output	Description
Layer1	Convolution	32x32x3	28x28x6	Weights are random number generated with normal distribution mu = 0 sigma = 0.1. Bias will be array Zeros(6). strides=[1, 1, 1, 1], padding='VALID'
	Relu			Activation
	Max Pooling	28x28x6		ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID'
Layer2	Convolution	14x14x6	10x10x16	Weights are random number generated with normal distribution mu = 0 sigma = 0.1. Bias will be array Zeros(16). strides=[1, 1, 1, 1], padding='VALID'
	Relu			Activation
	Max Pooling	10x10x16	5x5x16	ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID'
	Flatten	5x5x16	400	
Layer3	Fully Connected	400	120	Weights are random number generated with normal distribution mu = 0 sigma = 0.1. Bias will be array Zeros(120)
	Relu			Activation
Layer4	Fully Connected	120	84	Weights are random number generated with normal distribution mu = 0 sigma = 0.1. Bias will be array Zeros(84)
	Relu			Activation
Layer5	Fully Connected	84	43	Weights are random number generated with normal distribution mu = 0

				sigma = 0.1. Bias will be array Zeros(43)

Model Training

Model was trained based on Mini batching in view of constraints on memory available to train this data set. Epochs are used for forward and backward passes. After training model, validation dataset is used for measuring accuracy of the training.

Epochs = 10

Learning Rate = 0.01

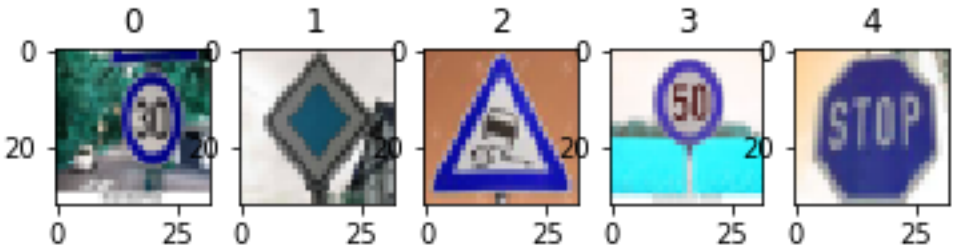
Batch size = 128

Training Steps:

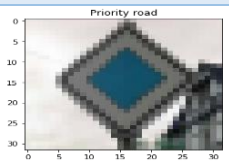
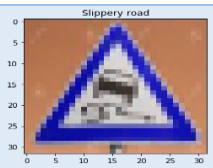
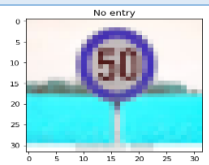
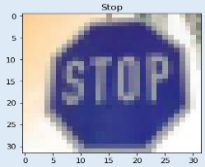
1. Logits are calculated using Lenet architecture
2. One hot encoding
3. Softmax function used to calculate probabilities from logits and loss values identify using cross entropy.
4. AdamOptmizer use for minimizing the loss. AdamOptimizer keeps track of moving averages of parameters(momentum) and therefore converges easily.
5. Tried using Dropout to avoid overfitting, however, Dropout was reducing accuracy level to very low like 70%. Therefore, further analysis would be needed to analyze reason for drop in accuracy due to introduction of Dropout in Lenet architecture.
6. With this setup, Training and *Validation accuracy* was around ***98.1%*** and also tried tuning hyperparameters like increasing learning rate to 0.02 and0.03 followed by changing batch size to 150. These parameters changing did not provide much improvement.
7. Test accuracy with provided test data was ***91.7%***

Test Model with New Images

Five new German traffic signs were downloaded from random sites and test was conducated with these images. Images are



After test results were with 60% accuracy.

New Test Images & Model Classification	Top 5 - Softmax Probabilities
	Road work ==> 0.996076 Double curve ==> 0.00243679 Speed limit (50km/h) ==> 0.000886609 Keep right ==> 0.00051367 Speed limit (80km/h) ==> 5.98775e-05
	Priority road ==> 0.99629 Keep right ==> 0.00188798 Yield ==> 0.000681015 Roundabout mandatory ==> 0.000506358 Speed limit (50km/h) ==> 0.000395788
	Slippery road ==> 0.999961 Road narrows on the right ==> 2.03658e-05 Wild animals crossing ==> 1.12616e-05 Beware of ice/snow ==> 5.05149e-06 Dangerous curve to the left ==> 9.20911e-07
	No entry ==> 0.415968 Slippery road ==> 0.20641 Priority road ==> 0.142504 Keep right ==> 0.113933 No passing ==> 0.113336
	Stop ==> 0.532501 Speed limit (80km/h) ==> 0.325142 Speed limit (60km/h) ==> 0.124059 Turn right ahead ==> 0.0116697 Speed limit (20km/h) ==> 0.00364296

Further work

1. At current, entire model was testing only Lenet. Regularization features like Dropout can be implemented.
2. New architectures like GoogLeNet and Microsoft ResNet can be designed for Traffic sign Recognition.
3. Implement DeConvNet to get better understanding and improve accuracy on new images' classification from current accuracy rate of 60%.
4. Need more study on Filters used by Lenet, GoogLeNet and ResNet. This might help identify new filter or get more understanding on learning details of architecture and math behind it.

References

1. Udacity Self-Driving car Nanodegree materials
2. Lenet Implementation: http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf
3. Reference of different architectures: https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html
4. Preprocessing references: http://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html