Traffic Sign Recognition
Project 3

Sarbjeet Singh
Feb 18, 2019

Rubric Points

Dataset Exploration

The provided template for determining the size of dataset and variables within was completed using python methods and no hardcoding. To better understand the quantity of each label type present in the image set, histograms for each data set were plotted. Additionally a random set of 8 images was pulled from the training set and shown with its label.

Design and Test a Model Architecture

Images were preprocessed to normalize the image and converted to gray scale. The LeNet architecture was modified, adding dropout to better train the model, but otherwise was left untouched, as this was effective to pass the validation threshold of 93%. I played around with the hyper parameters and settled with an epoch of 35, a batch size of 128 and training rate of 0.001, resulting in a validation accuracy of 94% and a test accuracy of 92%.

Test a Model on New Images

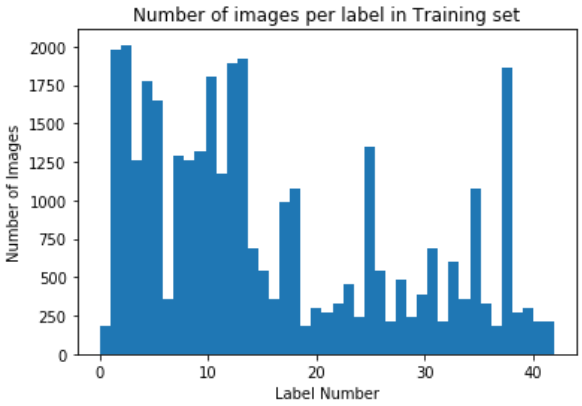
Five images from the Internet were selected and than converted from different sizes to be 32px X 32 px. This resulted in the images becoming distorted. The test accuracy on the images was only 20%, and may be a result of the distortion making it difficult for the model to identify it.

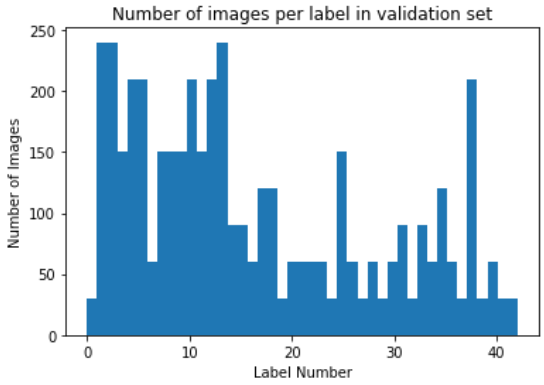
Data Set Summary & Exploration

To determine the size of different dataset parameters, I used python methods,

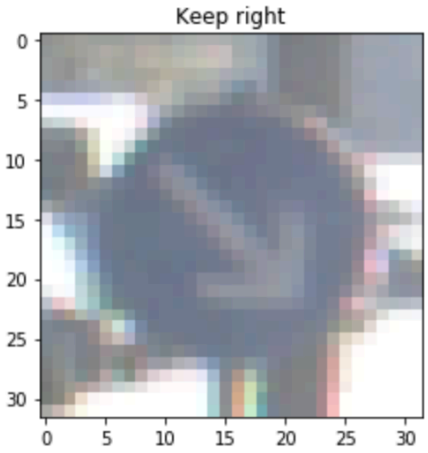
```
n_train = len(X_train)

# TODO: Number of validation examples
n_validation = len(X_validation)

# TODO: Number of testing examples.
n_test = len(X_test)

# TODO: What's the shape of an traffic sign image?
image_shape = X_train[0].shape

# TODO: How many unique classes/labels there are in the dataset.
#Looks at the largest number in the y_train set. Since it starts at 0, 1 is added to the value.
n_classes = max(y_train)+1
```

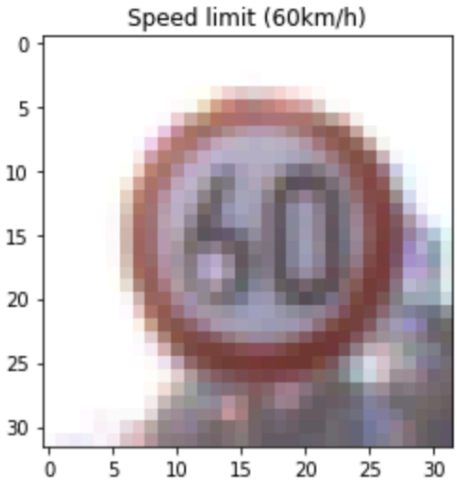
Below is an exploratory visualization of the images in the training, validation and test sets. The histograms depict the quantity of images for each label type in the set.

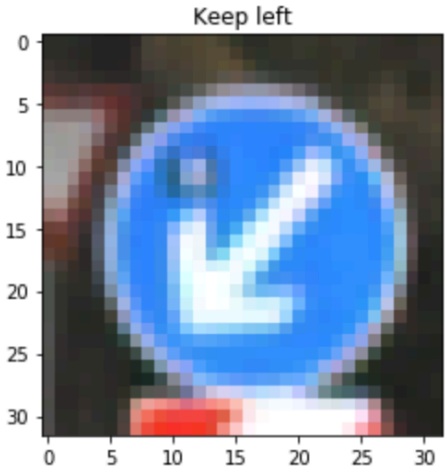


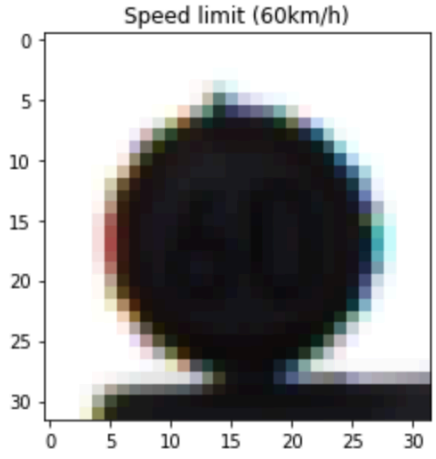


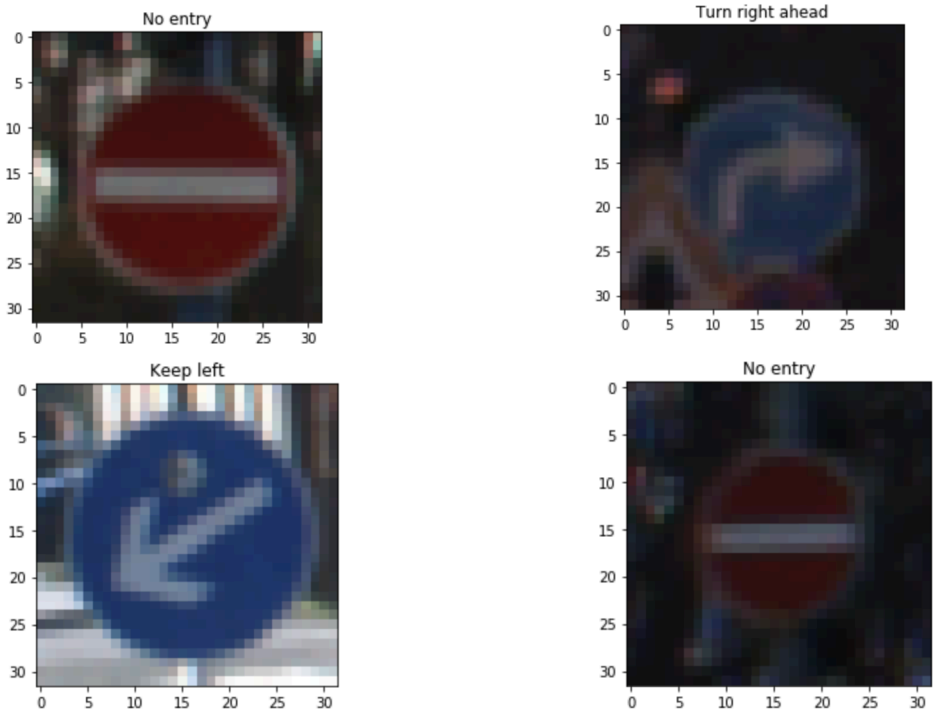
Additionally a random selection of 8 images from the training set were selected and shown with their label to give a better representation of what the data actually looks like.





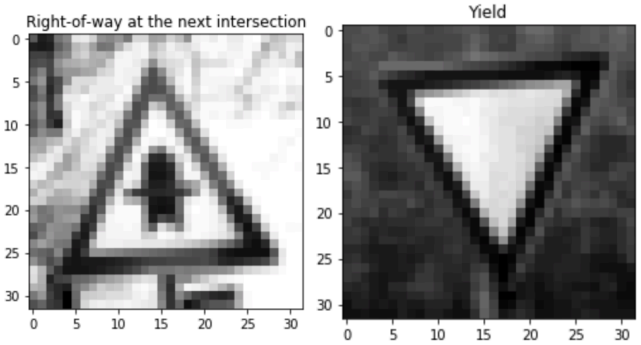






Design and Test a Model Architecture

1. My first pass through the training, I did not preprocess the data and utilized the RGB images. I found this to give accuracy around 89%. I than normalized the data so that the model could better compare the images, but found only a slight bump on the validation accuracy of ~0.5%. This led me to convert the images to gray scale. At first I attempted to utilize the CV2 library methods to accomplish this, but found that the dimensions of the matrix would not comply with the LeNets set-up, so I utilized numpy to achieve the conversion. The result of this pushed validation to ~90%. I read through Yann LeCuns LeNet paper and found that complex preprocessing could be done to achieve better results but manipulating the data to add shifts, different color spaces, etc… but since the accuracy was above ~90% I chose to move forward with modifying LeNet. Below are exmaples of preprocessed images



2. My final model is shown in the table below. It is basically the LeNet model for a gray scale image (32x32x1) with the addition of two drop out layers. The dropouts are performed after each activation on Layers 3 and Layers 4. I utilized this to make sure over fitting of the data did not occur, and utilized it twice to double the effect. A probability of 50% is used during training only.

Layer	Description
Input	32x32x1 Gray Scale image
Convolutional 5x5	Valid padding, Output 28x28x6
Activation	RELU
Pooling	2x2 stride, output 14x14x6
Convolutional 5x5	Valid padding, 10x10x16
Activation	RELU
Pooling	2x2 stride, 5x5x16
Flatten	5x5x16 to 400
Fully Connected	Output 120
Activation + Dropout	RELU
Fully Connected	Output 84
Activation + Dropout	RELU
Fully Connected	Output 43

3. To train this model, I utilized the Adam Optimizer from the lab and experimented with the hyper parameter values to tune the model. I found a batch size of 128, 35 EPOCH and a learning rate of 0.001 to be effective for passing the minimum validation accuracy threshold. A larger EPOCH did not change this but less than 15 was not efficient enough. A smaller batch size resulted in a decrease in accuracy.

4. As mentioned above, an iterative approach to tuning the model was used. This resulted in the following accuracies:

Test Accuracy: 99.8%
Validation Accuracy: 95.4%
Test Accuracy: 93.7%

The goal of my iterative approach was to start with LeNets, and see if I could optimize the hyper parameters to a point where no more accuracy increase were possible, at which point I would make a slight modification to the LeNet architecture, and than repeat this step.
The issue with the base LeNets was that I could not get it to have greater than 90% accuracy once I had optimized the hyper parameters and implemented some image pre-processing. Additionally, the test accuracy was much higher than the validation accuracy, which indicated possible over fitting of the data. Thus I chose to add

dropout to the activation layers. I experimented with one dropout step and again optimized but could not break past the 93% validation accuracy. I added another dropout step in final activation layer and found this to be effective in achieving accuracy beyond 93%. As the test accuracy and validation accuracy indicate, the model is not over fitting.

Test a Model on New Images

1. I found the following images on the internet

The images are clear but unlike the pictures in the current dataset, these images have some signs at angles or contain extra details. For example the priority road and stop sign images are from a distorted perspective. The model may fail to identify them as it wasn't trained on very many distorted images. The speed sign and caution sign both contain additional words under them, which could trick the model.

2. The model achieved 20% accuracy, thus only correctly predicting one image. The results are as follows:

Stop Sign	Stop Sign
Pedestrian	20 km/h Speed Sign
General Caution	No Entry
30 km/h Speed Sign	Keep Right
Priority Road	Turn Right Ahead

As we can see, the model failed at classifying 4 of the 5 images. This is far below the test set accuracy from before, and could be explained by the difference in the images discussed above. As the table below shows, for the speed and priority signs, the softmax values are low, indicating that the model is not very sure about its classifications and that more training is needed on the model. There is a definite issue with the pedestrian sign, where the model is 100% confident but the result is 100% wrong.

Stop Sign	99.9
Pedestrian	100
General Caution	92
30 km/h Speed Sign	38

Priority Road	33.5