Traffic Sign Recognition

Mohit Pandey

2018
January

Chapter 1

Introduction

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

- Load the data set (see below for links to the project 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 softmax probabilities of the new images
- Summarize the results with a written report

Chapter 2

Writeup/Readme

You're reading it. Project code is at *https://github.com/diamondspark/Traffic-Sign-Classifier/blob/master/Traffic_Sign_Classifier.ipynb*

Chapter 3

Data Set Summary and Exploration

3.1 Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

I used numpy to calculate summary statistics of the traffic signs data set:
1. Number of training examples= 34799
2. Number of validation examples = 4410
3. Number of validation examples = 12630
4. Shape of traffic sign image = (32,32,3)
5. Number of distinct classes = 43

3.2 Include an exploratory visualization of the dataset.

We see that the distribution of classes in the dataset is highly non uniform. This class imbalance would probably result in low precision and recall even if we got decent accuracy. To counter this class imbalance, we'll do data augmentation as a pre-processing step.

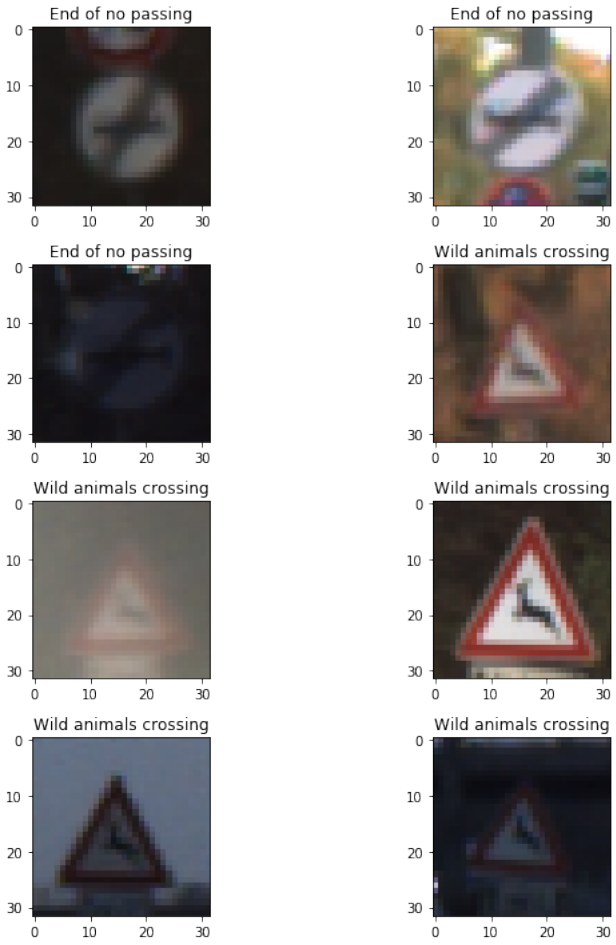

Figure 3.1: Sample images from Original Dataset

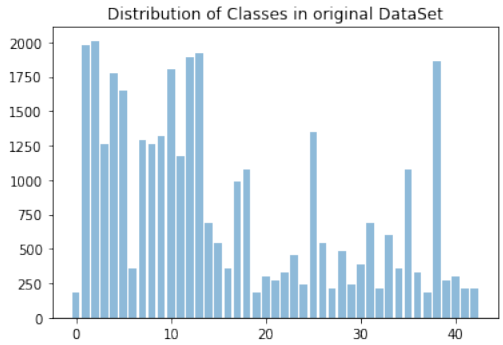

Figure 3.2: Class Distribution of Original Dataset

Chapter 4

Design and Test a Model Architecture

4.1 Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

4.1.1 Normalization

The first step, I decided to normalize the images. I used (pixel-128)/128 to approximately normalize the data. This was done because this helps to normal-

ize each dimension so that the min and max along the dimension is -1 and 1 respectively. In our case of RGB images, this ensures that the contribution of each of the 3 channel is approximately the same and the set of weights learned affect each of the 3 channels in approximately same manner.

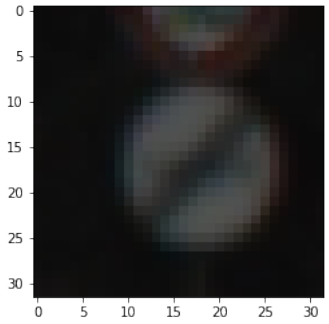

Figure 4.1: Before Normalization

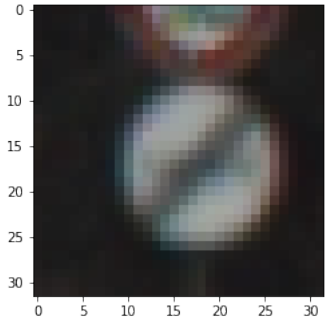

Figure 4.2: After Normalization

4.1.2 Data Augmentation

The next step was data-augmentation. This was done after I saw the distribution of classes in the dataset [3.2] There was significant difference between number of instances of each class in the dataset. I tried to balance the class distribution by making instances in each class almost equal to 5000. This was done by randomly changing the brightness and rotation. I used OpenCV library to achieve this.

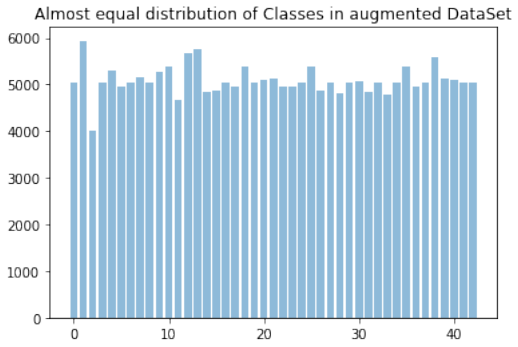

Figure 4.3: Class Distribution in Augmented dataset

4.2 Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

I used Lenet with l2 regularization to prevent overfitting.

Layers and Operations	**Description**
Input	32x32x3 RGB image
Convolution 5x5	1x1 stride, VALID padding, outputs 28x28x6
RELU	
Max Pooling	2x2 stride, VALID padding, outputs 14x14x6
Convolution 5x5	1x1 stride, VALID padding, outputs 10x10x16
RELU	
Max Pooling	2x2 stride, VALID padding, outputs 5x5x16
Flatten	Output = 400
Fully Connected # 1	Output = 120
RELU	
Fully Connected # 2	Output = 84
RELU	
Fully Connected # 2	Output = 43
Regularizer	L-2 on all weights

Table 4.1: The LeNet model architecture used

4.3 Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

I used Adam Optimizer to train Lenet architecture as described in [4.1] with Batch size of 128 and ran for 150 Epochs while saving the model at each epoch. The learning of 0.001 was chosen by doing grid search for hyperparameter tuning.

4.4 Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.

My final model results were:
1. Training accuracy: 99.35%
2. Validation accuracy: 95.64%
3. Test Set accuracy: 94.33%

These results are calculated and printed in cell #16 of the code file.
The architecture chosen was LeNet described in [4.1]. This model was chosen as a continuation of an earlier assignment dealing with classification of MNIST dataset. Since this model performed well on multi-class image classification (MNIST), it makes sense to try this model on a similar multi-class image classification (Traffic-sign classification) problem. On running the vanilla LeNet model with given dataset achieved nearly 90% validation accuracy hence giving hope to investigate this model further on given problem. However, the architecture of LeNet for MNIST took as input a single channel image. In our case, we have a 3 channel RGB Image to work with hence the input tensor needs to

be modified. Upon performing grid search for tuning the learning rate, I saw that as I varied the learning rate parameter even though the training accuracy kept going up, the validation accuracy still hovered around 89 and 90.5%. This was a clear indication that the learning model was overfitting. To fix this I used an L-2 regularization. I chose 0.001 to be the optimal one since it gave a high training as well as validation accuracy (as noted above). Not surprising that this trained model performed pretty well on the test set as well.

Chapter 5

Test a Model on New Images

5.1 Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

Here are five German traffic signs that I found on the web:

,

Figure 5.1: Label: Stop

Figure 5.2: Label: Speed limit 70 Km/hr

Figure 5.3: Label: Ahead Only

Figure 5.4: Label: No Entry

Figure 5.5: Label: Yield

Since the above figures are of different sizes and our trained model takes in images of shape (32x32x3), all the images were reshaped to this shape using numpy library.
I noted the following qualities of the images were necessary for them to be classified correctly.
1. The chosen images should obviously be present in our training set as one of the class.
2. The traffic sign itself must be in the center of the image covering about two-thirds of the entire image.
3. The image must not be cluttered with a lot of noise like the road, cars, pedestrian etc.
4. [5.3] and other images of this kind, irrespective of following all the above criteria were still hard to classify correctly. One possible explanation for this

maybe that there was no image in the training set which was simply the traffic sign on a white background.

5.2 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).

Ground Truth	**Prediction**
Stop	Stop
Speed Limit 70	Speed Limit 70
Ahead Only	Ahead Only
No Entry	No Entry
Yield	Yield

Table 5.1: Prediction of Online Images

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the test set of 94.33% [4.4]

Probability	Prediction
1.0	Stop
0.99	Speed Limit 70
1.0	Ahead Only
0.99	No Entry
1.0	Yield

Table 5.2: Prediction of Online Images

5.3 Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

The results look pretty impressive. The correct label is detected with high probability. Please refer to code cell #13 for top 5 softmax probabilities. For each of the 5 test images, the model was pretty sure of the correct label and returned very high probability of correct class.

- Image#1

Probablity	Prediction
∼1.0	Stop
9.1e-12	Speed Limit 80
1.0e-12	Speed Limit 20
1.8e-13	No Vehicles
9.7e-14	Speed Limit 100

Table 5.3: Softmax probablities Image 1

- Image#2
- Image#3
- Image#4
- Image#5

Probablity	Prediction
0.99	Speed Limit 70
1.0e-4	Generak Caution
6.8e-5	Roundabout Mandatory
2.9e-5	Speed Limit 20
1.3e-5	Pedestrians

Table 5.4: Softmax probablities Image 2

Probablity	Prediction
∼1.0	Ahead Only
2.4e-9	Turn Right Ahead
6.8e-13	Go straight or right
6.9e-14	Turn left ahead
6.5e-15	Speed Limit 60

Table 5.5: Softmax probablities Image 3

Probablity	Prediction
0.99	No Entry
1.3e-4	No passing
8.7e-6	Stop
1.8e-6	Priority road
8.1e-9	Yield

Table 5.6: Softmax probablities Image 4

Probablity	Prediction
∼1.0	Yield
2.1e-11	Speed limit 30
1.3e-11	Keep left
7.6e-12	No vehicles
5.4e-12	No passing

Table 5.7: Softmax probablities Image 5