Build a Traffic Sign Recognition Classifier¶

Writeup

Here is a link to my project code (Traffic_Sign_Classifier.ipynb)

Data Set Summary & Exploration

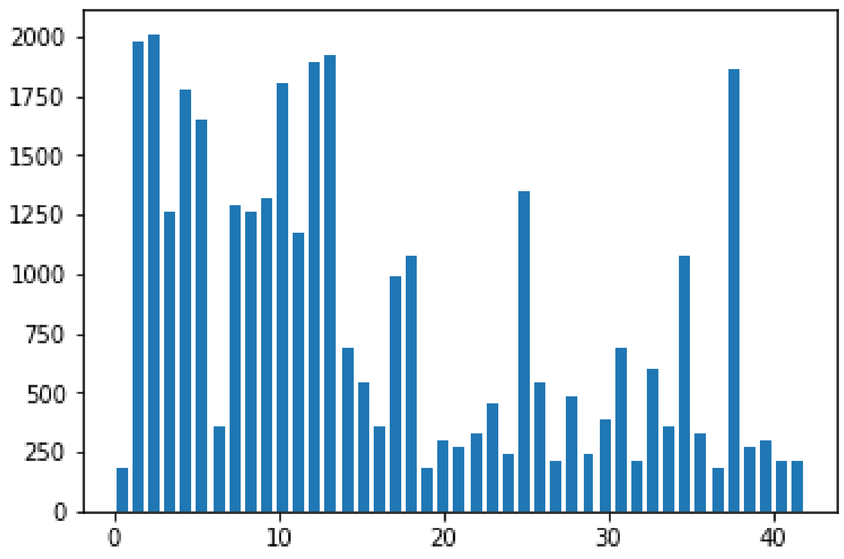
1. Summary of the data set

I used the numpy library to calculate summary statistics of the traffic signs data set:

- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is (32,32,3)
- The number of unique classes/labels in the data set is 43

2. Exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data is spread into 43 classes

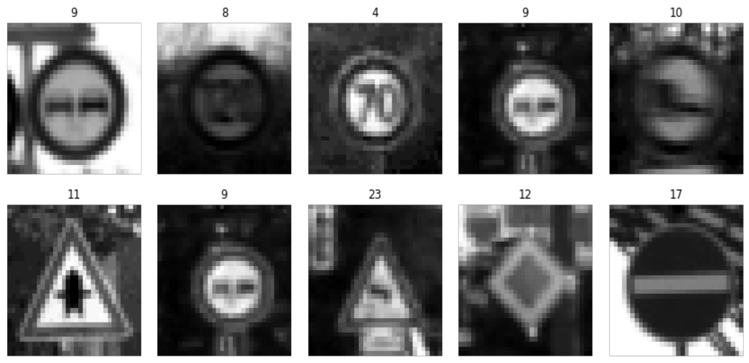


Design and Test a Model Architecture

1. Preprocess of image data

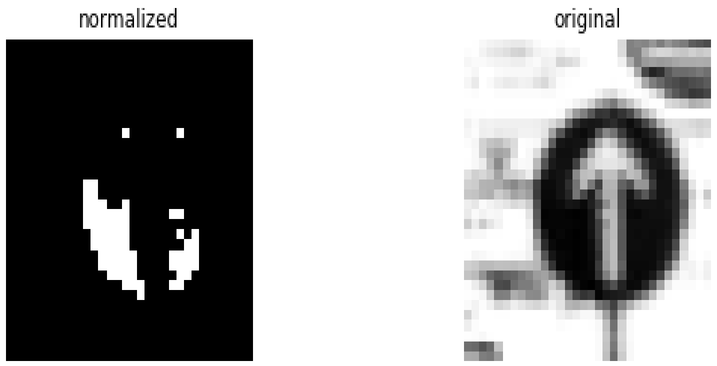
As a first step, I decided to convert the images to grayscale because the visual symbolic data is preserved in a grayscale image and a grayscale image can be processesed faster than a color image

Here is an example of a traffic sign image after grayscaling.



As a last step, I tried normalizing the image data to put it in -1 to +1 range for easy and faster processing

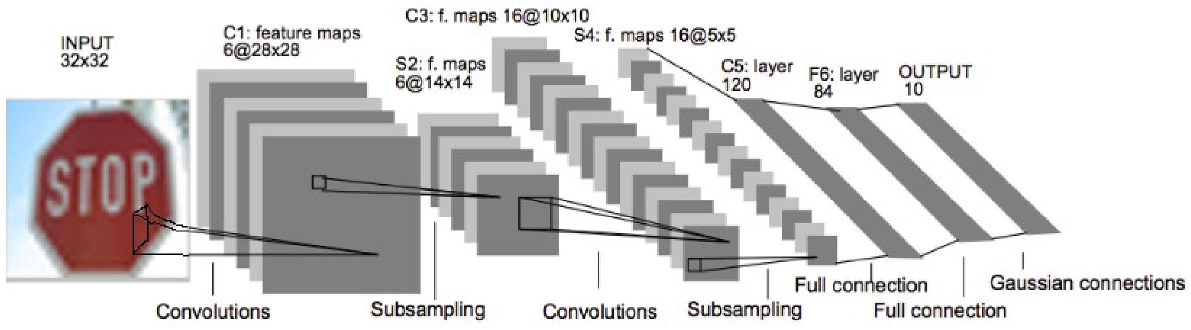
Here is an example of an augmented image and original image:



I decided not to use normalizing as i felt the displayed output had loss of data, so i decided to proceed with grayscale image.

Update: the issue was because of my jupyter running python 2. I then fixed it. Later while training the model, I realised the data took more time to train (the model was stuck @ 5%), so i decided to use the grascale image without normalization.

2. Model architecture



I've used LeNet architecture used in the lesson video, as i felt comfortable using it given it was a proven method to start with the project here we have to classify the image for shapes which is applicable to both the data set. Also, with limited knowledge on neural networks and Tensorflow i decided to use the LeNet archotecture and would love to explore other options later.

3. training of model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

To train the model, I used batch size of 100 and training rate of 0.005, 64 EPOCHS, mu of 0 and sigma as 0.1

For hyperparameters, i haven't made any change as i left it with mean of 0 and standard deviation of 0.1 which i felt comfortable with.

I chose a learning rate of 0.005 as i tried with 0.0001 which required more epochs. I also tried with 0.25 which came close to 85% accurac but later the accuracy was varying @ 88% ±2%. So i decided to use 0.005.

For epochs, i initialised to 2500 and set a conddition to finalize model @ 93% accuracy as i tried to run it for 2500, i got an accuracy of 95% @ 1500 epochs and coulddn’t save the model due to kernel issues.

For batch size i started with 128 but found it took more iteration to attain 80% accuracy. I then inceased to 5000 and 10000 data per batch and found that the data converged slow but at steady pace, but took more time. I then used 64 but the time taken to execute each epoch was more(about 20 sec whereas higher batch size took approx 12 seconds in Intel i7 with 8GB RAM). So i decided to stick with 100 as batch size.

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:

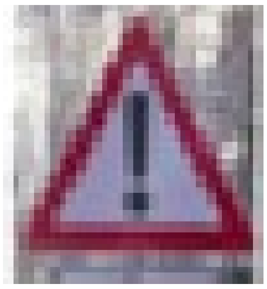
- training set accuracy of 99.94%
- validation set accuracy of 93.24%
- test set accuracy of 91.9%

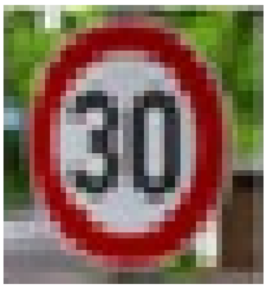
iterative approach was chosen:

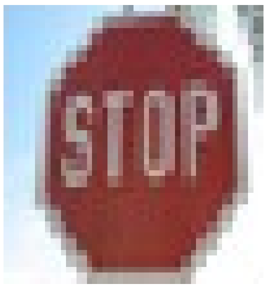
- The architecture was same but i was using different parameters
- The network did not yield accuracy more than 88%
- The batch size was reduced and the epochs were increased

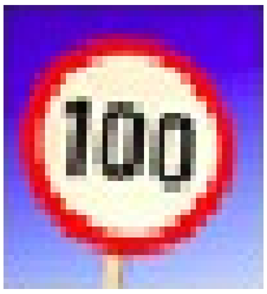
Test a Model on New Images

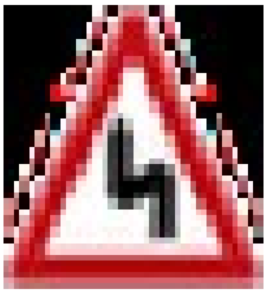
1. Here are German traffic signs that I found on the web:

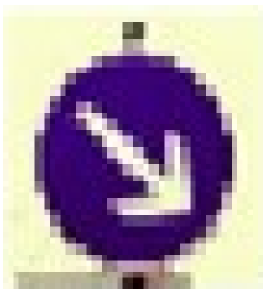












The fourth image might be difficult to classify because the text is smudged and is classified as 60 by the classifier

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).

Here are the results of the prediction:

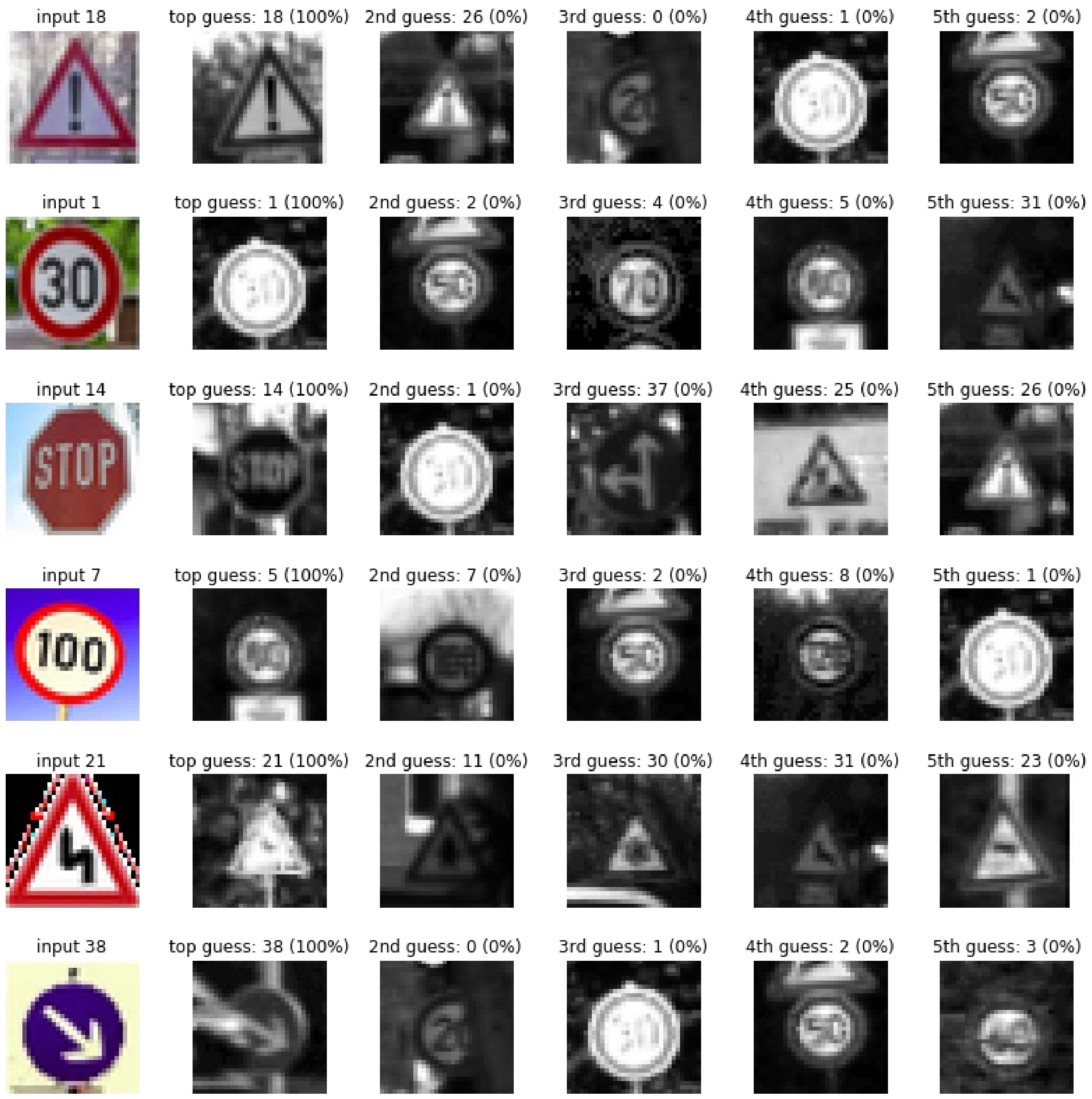
Image	Prediction
General Caution	General Caution
Speed Limit 30km/h	Speed Limit 30km/h
Stop	Stop
100 km/h	60 km/h

Image	Prediction
Double curve	Double curve
Keep right	Keep right

The model was able to correctly guess 5 of the 6 traffic signs, which gives an accuracy of 83.3%.

3. The code for making predictions on my final model is located in the 76th cell of the Ipython notebook.

For the first image, the model is relatively sure that this is a stop sign (probability of 0.6), and the image does contain a stop sign. The top five soft max probabilities were



Img1: General Caution

Probability	Prediction
100	General Caution
0	Traffic signals
0	Speed limit (20km/h)
0	Speed Limit (30 km/h)
0	Speed Limit (50 km/h)

Img 2: Speed Limit 30 km/h

Probability	Prediction
100	Speed Limit 30 km/h
0	Speed Limit 50km/h
0	Speed Limit 70km/h
0	Speed Limit 80km/h
0	Wild animals crossing

Img 3: Stop

Probability	Prediction
100	Stop
0	Speed Limit 30km/h
0	Go straight or left
0	Road work
0	Traffic signals

Img 4: Speed limit 100 km/h

Probability	Prediction
100	Speed limit 100 km/h
0	Speed Limit 80km/h
0	Speed limit (50km/h)
0	End of speed limit (80km/h)
0	Speed limit (30km/h)

Img 5: Double curve

Probability	Prediction
100	Double curve

Probabi lity	Prediction
0	No Passing
0	Beware of ice/snow
0	Wild animals crossing
0	Slippery road

Img 6 : keep right

Probabi lity	Prediction
100	Keep right
0	Speed Limit 20km/h
0	Speed Limit 30km/h
0	Speed Limit 50km/h
0	Speed Limit 60km/h