RUBRIC POINTS – WRITE UP

Data Set Summary & Exploration

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.

   Solution: the code for this is in the cell 2. I used Numpy to explore and work with the image data.

   * The size of training set is?
   Solution: 34799 rows
   * The size of the validation set is?
   Solution: 4410 rows
   * The size of test set is?
   Solution: 12630 rows
   * The shape of a traffic sign image is?
   Solution: Image shape = (32, 32, 3)
   * The number of unique classes/labels in the data set is?
   Solution: Number of classes = 43

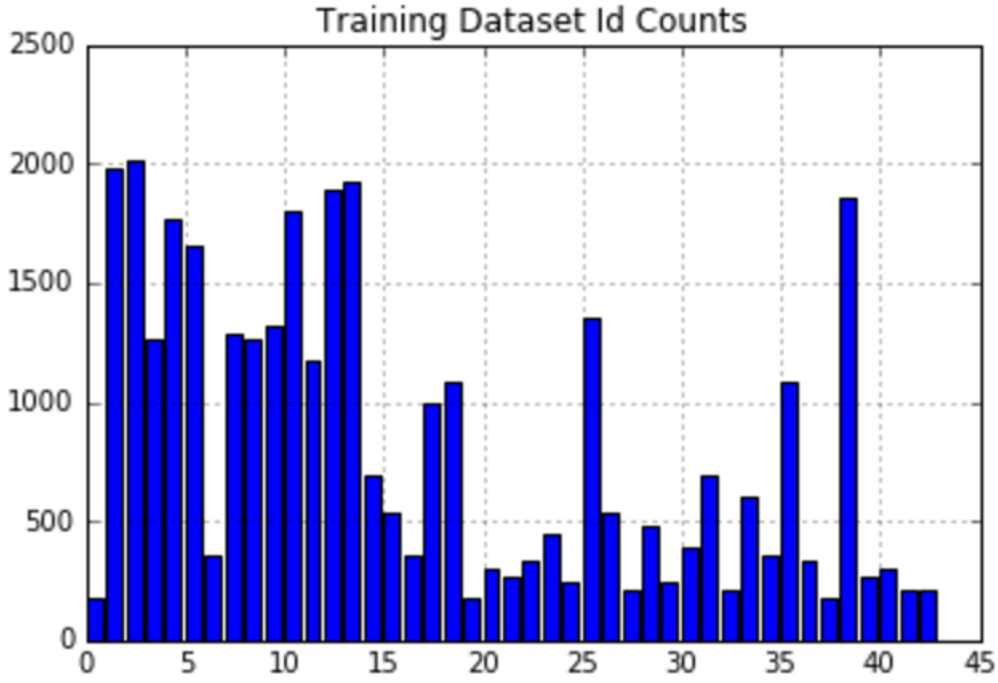
2. Include an exploratory visualization of the dataset

   Solution: Here is an exploratory visualization of the data set. It's a table of the images in the dataset with their respective labels:

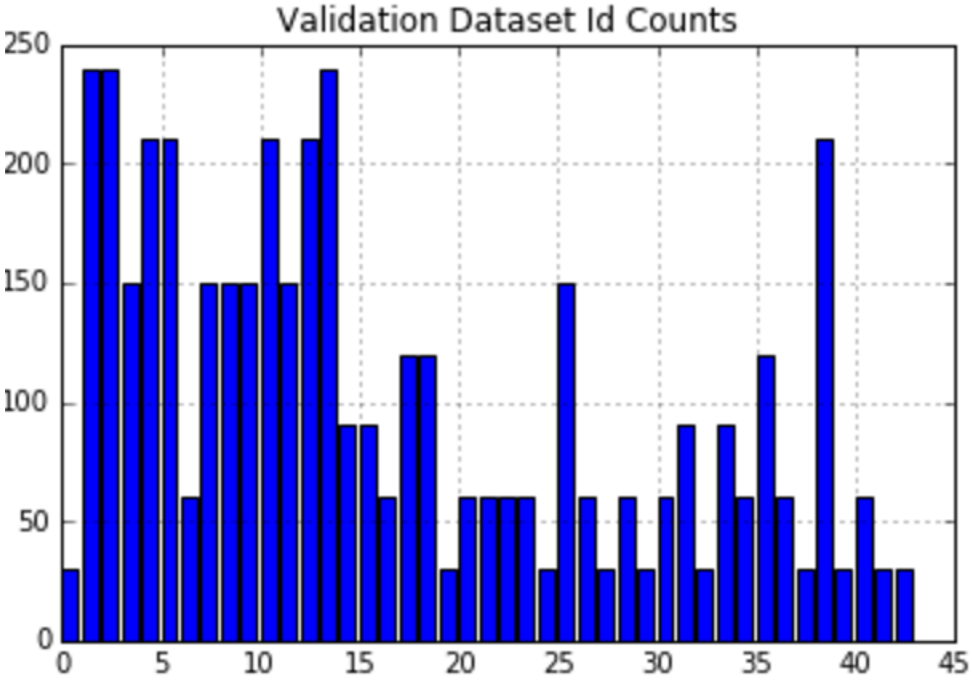
   The previous images show the dataset is made of color images.

To check the distribution of frecuency in the classes I printed a histogram for each data set:

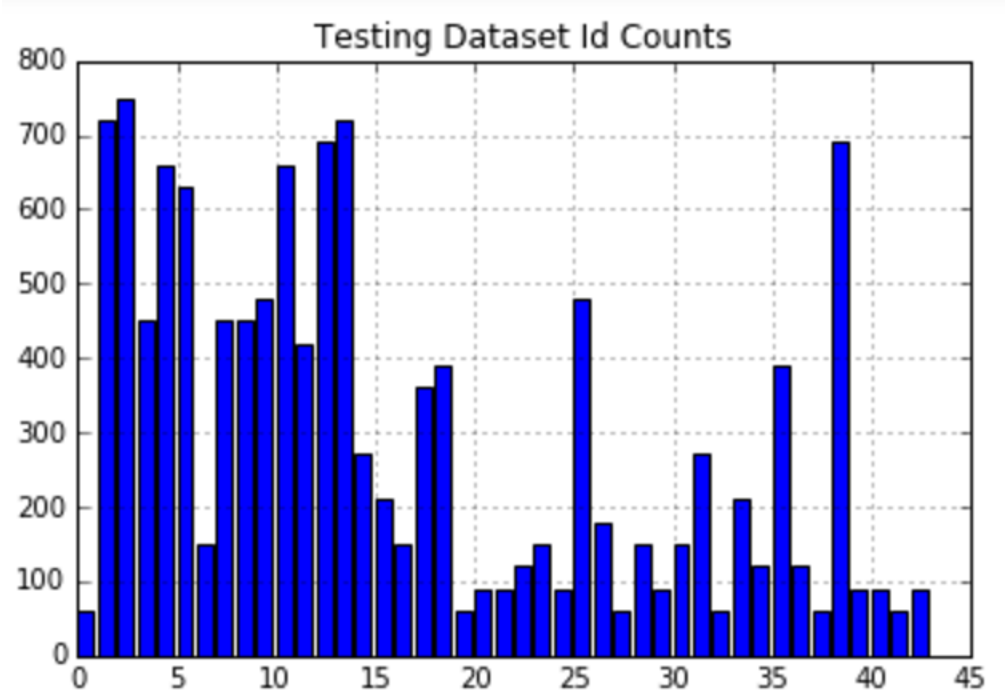
Train Data



Validation Data



Test Data



From the histograms, we can see that the data has a similar distribution.

Design and Test a Model Architecture

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.)

   Solution: As a first step, I decided to convert the images to grayscale because it was suggested in Yann LeCun's paper. I had also done a similar thing for the lane detection problem at the beginning of the Nano-degree. Grayscale preserves the enough data to identify a sign but allows for faster processing because there is less data to go through in the convolutions.

   In the cell number 7 I printed a comparison of the color images next to their grayscale equivalent:

I did this for the validation and the test dataset. All seem to be good enough for the neural network.

As a last step, I normalized the image data because it avoids having to calculate large values and it's also easier for a human to understand:

The normalized image still represents the original sign in the same way but will be faster to compute for gradient descent as well as leveling the influence of each feature on the model. I printed samples for the train, valid and test datasets and all showed the same behavior.

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.

   Solution: The bases I used is the LeNet architecture but I added dropout functions after the first two fully connected layers to reduce overfitting as I learned on the course videos.

Layer	Description
Input	32x32x1 Grayscale Image
Convolution (with Relu)	Output 28x28x6
Pooling	Output 14x14x6
Convolution (with Relu)	Output 10x10x16
Pooling	Output 5x5x16
Flatten	Output 400
Fully Connected (with Relu and Dropout)	Output 120
Fully Connected (with Relu and Dropout)	Output 84
Fully Connected	Output 43

   The code for the model can be found in the cell 15.

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.

   Solution: To train the model, I used the same approach as LeNet with an AdamOptimizer from Tensorflow and tried to use a 0.001 learning rate to start. Then I tested a 0.0009 rate and the accuracy improved. I used 100 Epochs but I saw that about 25 was enough, the model wasn't improving after that. I tried 100 and 150 for the batch size but didn't see any difference so I left it at 100.

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.

   Solution: The accuracy was 93 or higher from the first try. I ran it several times and would mostly get 93+ and a few times I got a 94+, the final run I got 95+ for the validation set and 93+ for the test set.

   My final model results were:
   * training set accuracy of?
   Solution: 0.996

* validation set accuracy of?
Solution: 0.95
* test set accuracy of?
Solution: 0.937

If an iterative approach was chosen:
* What was the first architecture that was tried and why was it chosen?
Solution: LeNet because it was suggested in the lessong
* What were some problems with the initial architecture?
Solution: I found no problems with it but I thought of adding dropouts since I just learned them from the course and it did improve the accuracy for about a percentage point.
* How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.
Solution: Dropout was added to help reduce overfitting
* Which parameters were tuned? How were they adjusted and why?
Solution: The learning rate was changed from the LeNet suggestion of 0.001 to 0.0009 to test smaller steps and it helped to get to the global minima faster.
* What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?
Solution: Convolutions are good for detecting a pattern regardless of where it's located in an image. This is particularly necessary for traffic sign images in the real world. Dropout reduces overfitting so the model becomes more compact which is also useful if this classifier needs to run on a car.

Test a Model on New Images

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.

   Solution: The images I used to test the model are in the folder "images" in the GitHub repository of this project. There are 6 in total.

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).

Solution: All 6 images were identified correctly. This leaves the model with a 100% accuracy on the extra images used

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)

   Solution: The code for making predictions on my final model is in the 29th cell of the Jupyter notebook.

   For the first image, the model is sure that this is a 'Speed limit (30km/h)' sign (probability of 100%) and the result is accurate. The top five soft max probabilities were:

   Guess (1, 'Speed limit (30km/h)') : (1.000)
   Guess (32, 'End of all speed and passing limits') : (0.000)
   Guess (11, 'Right-of-way at the next intersection') : (0.000)
   Guess (40, 'Roundabout mandatory') : (0.000)
   Guess (2, 'Speed limit (50km/h)') : (0.000)

   For the second image, the model is highly assertive that this is a ‘Bumpy road’ sign (probability of 85.9%) and the result is accurate. The top five soft max probabilities were:

   Guess (22, 'Bumpy road') : (0.859)
   Guess (26, 'Traffic signals') : (0.137)
   Guess (38, 'Keep right') : (0.002)
   Guess (25, 'Road work') : (0.001)
   Guess (20, 'Dangerous curve to the right') : (0.000)

   For the third image, the model is sure that this is a 'Ahead only' sign (probability of 100%) and the result is accurate. The top five soft max probabilities were:

   Guess (35, 'Ahead only') : (1.000)
   Guess (13, 'Yield') : (0.000)
   Guess (9, 'No passing') : (0.000)
   Guess (33, 'Turn right ahead') : (0.000)
   Guess (34, 'Turn left ahead') : (0.000)

   For the fourth image, the model is highly assertive that this is a 'No vehicles' sign (probability of 95.5%) and the result is accurate. The top five soft max probabilities were:

   Guess (15, 'No vehicles') : (0.955)
   Guess (12, 'Priority road') : (0.030)
   Guess (38, 'Keep right') : (0.007)

Guess (14, 'Stop') : (0.004)
Guess (36, 'Go straight or right') : (0.001)

For the fifth image, the model is highly assertive that this is a 'Go straight or left' sign (probability of 97.1%) and the result is accurate. The top five soft max probabilities were:

Guess (37, 'Go straight or left') : (0.971)
Guess (40, 'Roundabout mandatory') : (0.018)
Guess (4, 'Speed limit (70km/h)') : (0.011)
Guess (39, 'Keep left') : (0.000)
Guess (1, 'Speed limit (30km/h)') : (0.000)

For the sixth image, the model is sure that this is a 'General caution' sign (probability of 100%) and the result is accurate. The top five soft max probabilities were:

Guess (18, 'General caution') : (1.000)
Guess (26, 'Traffic signals') : (0.000)
Guess (27, 'Pedestrians') : (0.000)
Guess (40, 'Roundabout mandatory') : (0.000)
Guess (38, 'Keep right') : (0.000)