#**Traffic Sign Recognition**

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

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one.

Data Set Summary & Exploration

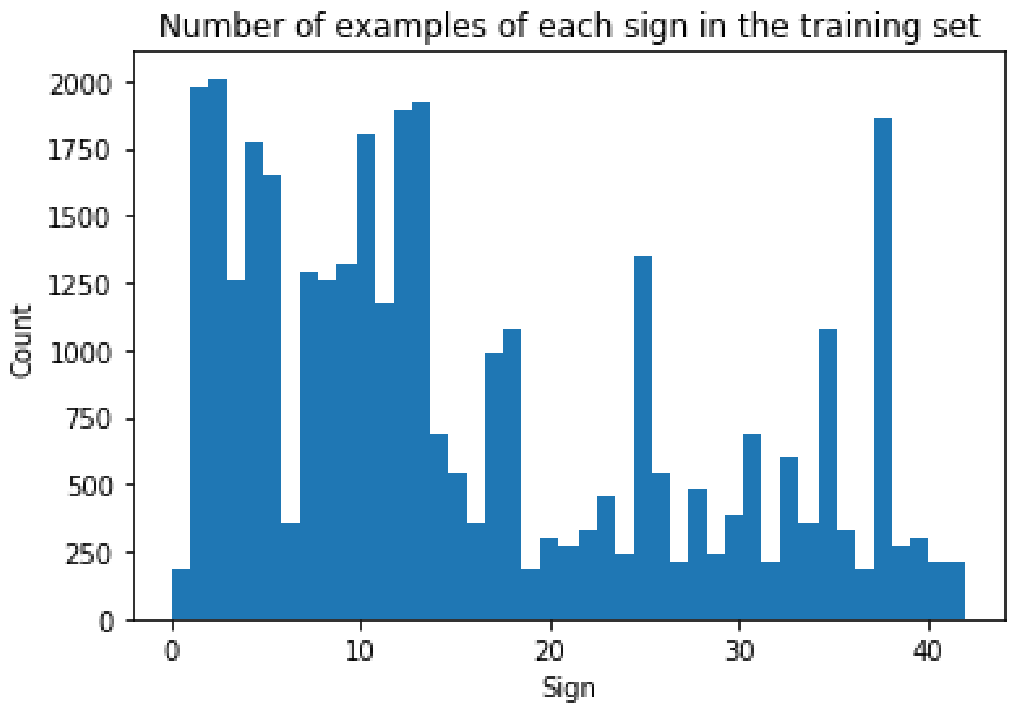
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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

- The size of training set is 34,799
- The size of the validation set is 4,410
- The size of test set is 12,630
- The shape of a traffic sign image is (32, 32, 3)
- The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

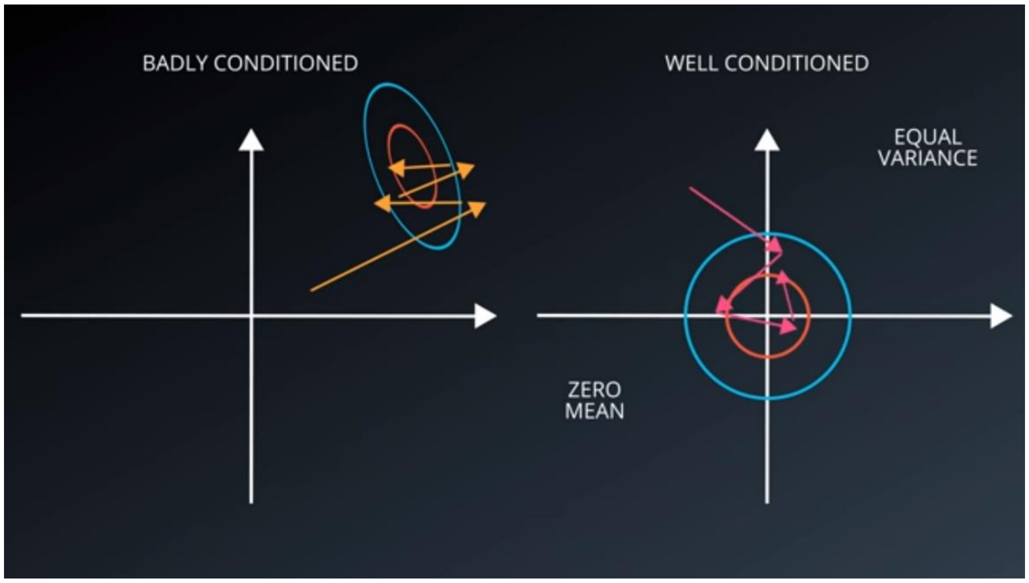
Here is an exploratory visualization of the data set. It is a bar chart showing the split of 43 classes. We can clearly observe that some classes have counts as low as <200.



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?

I normalized the image data because we want to have **numerical stability** i.e. we want our variables to have zero mean and equal variance whenever possible. Also, there are good mathematical reasons to keep the values you compute roughly around a mean of zero and equal variance when you are doing optimization. A **badly conditioned problem** means the optimizer has to do a lot of searching to go and find a good solution. A **well conditioned problem** makes it a lot easier for the optimizer to do its job as depicted in below figure.

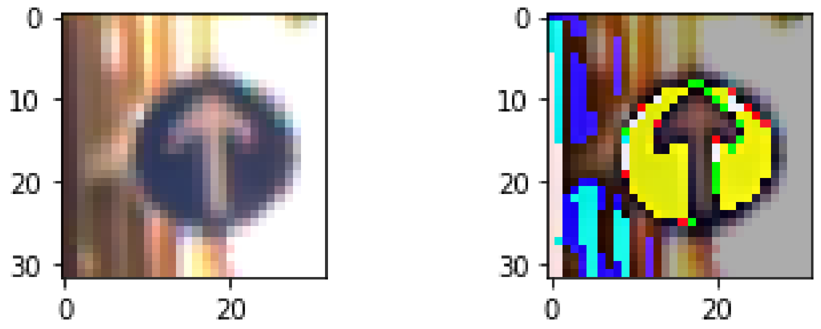


I used min-max normalization to normalize the features.

X_train = (X_train - X_train.mean()) / (np.max(X_train) - np.min(X_train))

It doesn't change the content of the image but it makes it much easier for the optimization to proceed numerically.

Here is an example of a traffic sign image before and after normalizing.



I also generated fake dataset by transforming the images through rotation, translation & applying affine transform. However, this didn't help much with improving the accuracy of the model so I removed it from the training set.

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.

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x10
RELU	
Max pooling	1x1 stride, outputs 26x26x10
Convolution 3x3	1x1 stride, valid padding, outputs 24x24x15
RELU	
Max pooling	2x2 stride, outputs 12x12x15
Convolution 3x3	1x1 stride, valid padding, outputs 10x10x30
RELU	
Max pooling	2x2 stride, outputs 5x5x30
Fully connected layer 1	outputs 800
Fully connected layer 2	outputs 400
Fully connected layer 3	outputs 200
Output layer	Outputs 43

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.

To train the model, I used:

Adam Optimizer - Naturally performs step size annealing unlike SGD where we generally decrease step size after some epochs. Uses exponential decay that consists of not just considering the average (first moment), but also the variance (second moment) of the previous steps.

- **Parameters used:**
  - Epochs = 20

- Batch size = 128
- Rate = 0.001

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:

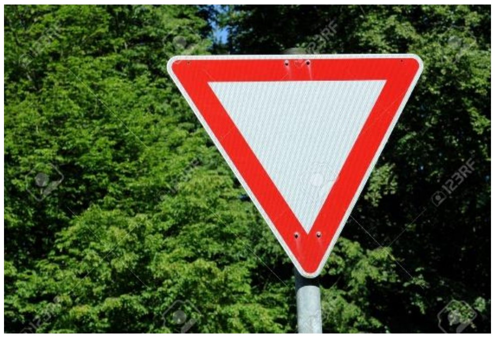
- Validation set accuracy of 96.5%
- test set accuracy of 94.9%

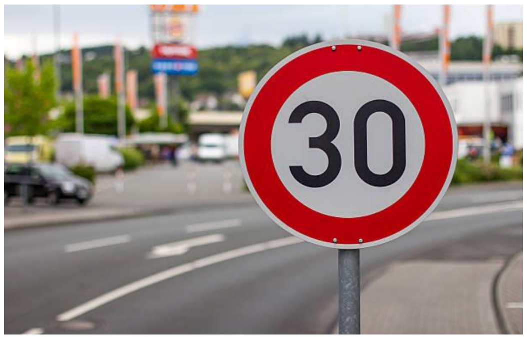
Used the Le Net architecture as a base as it's known for recognizing complex patterns in an image. With this I was able to achieve validation accuracy of 93%. Subsequently, I increased the complexity of the architecture by inserting in 1 additional layer of convolution, max pooling & fully connected layer respectively which helped in training the model and improving it's accuracy (see above section for details of architecture)

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.

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





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

Here are the results of the prediction:

Image	Prediction
Yield	Yield
Speed limit (30km/h)	Speed limit (30km/h)
Priority road	Priority road
No entry	No entry
Bumpy road	Bumpy road

The model was able to correctly guess all 5 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the test set of 94.9%

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.

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

All the predictions are done with almost certainty i.e. ~100% probability. The top five soft max probabilities were

Image	Prediction	Top 5 Softmax probability
Yield	Yield	[1. , 0. , 0. , 0. 0.]
Speed limit (30km/h)	Speed limit (30km/h)	[1. , 0. , 0. , 0. 0.]
Priority road	Priority road	[1. , 0. , 0. , 0. 0.]
No entry	No entry	[1. , 0. , 0. , 0. 0.]
Bumpy road	Bumpy road	[1. , 0. , 0. , 0. 0.]