My project goes according to the following,

Loading the pickled data (Cell 1):

Where I loaded the files (train, valid and test) features and labels.

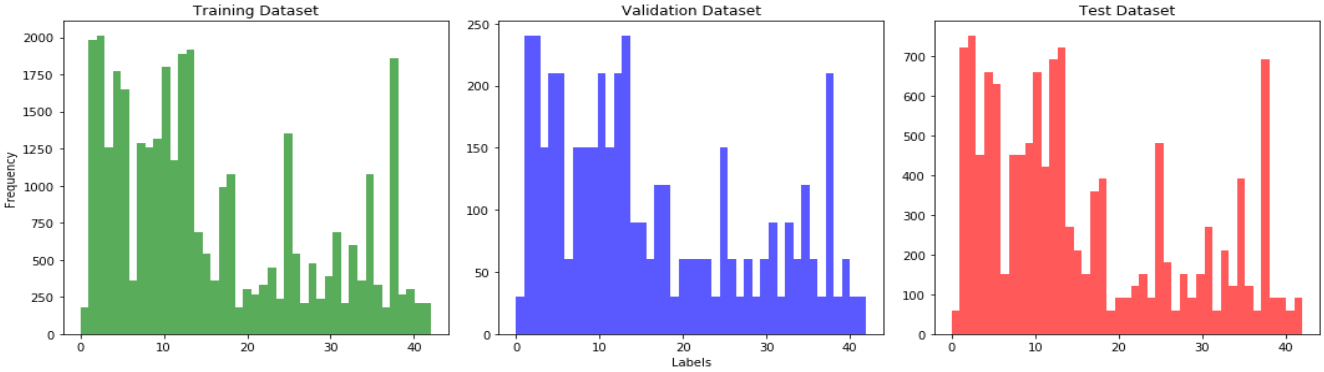
Show Basic summery of the data (Cell 2):

Where I printed very basic info about the dataset.

- Number of training and testing images
- Number of classes
- Image size

A Random Sample Visualization (Cell 3&4):

Where I plotted a histogram showing the frequency of each class in training, validation and testing data.



Then plotted a sample of the 5 highest frequencies as well.

Then, a random image from each class.

Preprocessing (Cell 5):

In my submission preprocessing consists of two steps:

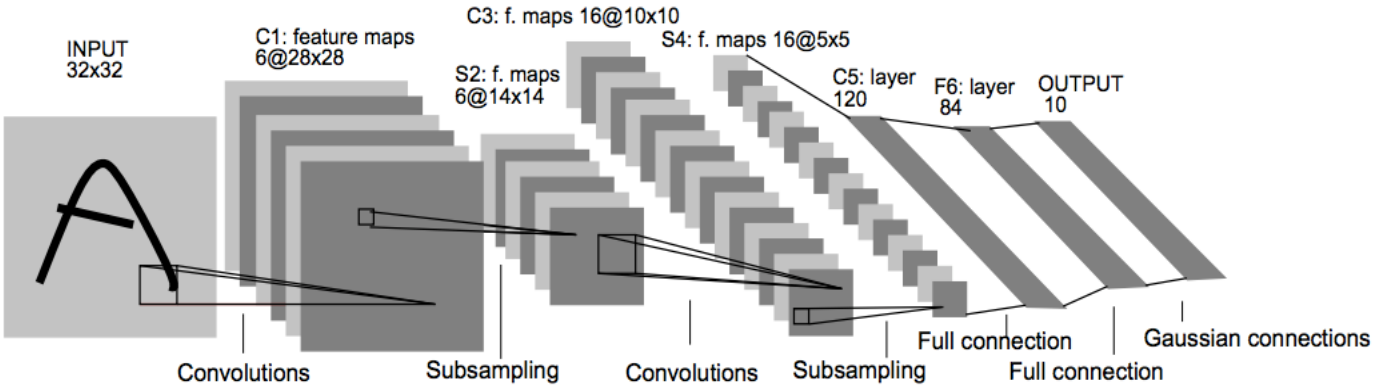
1- Convert to Grayscale, it helps to reduce training time of the network.
2- Normalize data to the range (-1:1), this was done by subtracting 128 and dividing by 128 as suggested in the lessons to provide around 0 mean.

Training, validation and Testing:

I used SciKit Learn train_test_split function to create a validation portion of the training set, it was 20% of the dataset that are used as data validation.

Model Architecture:

My model architecture was the original lenet architecture:



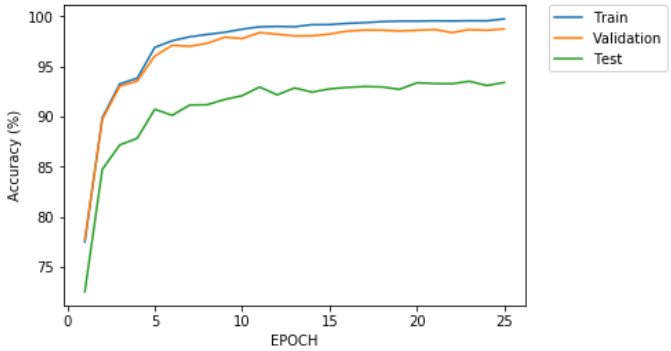
Layer No.	Type	Input	Output	Activation
1	Convolutional	32x32x1	28x28x6	Relu
	Pooling (max_pool)	28x28x6	14x14x6	
2	Convolutional	14x14x6	10x10x16	Relu
	Pooling (max_pool)	10x10x16	5x5x16	
	Flatten	5x5x16	400	
3	Fully Connected	400	120	Relu
	Dropout			
4	Fully Connected	120	84	Relu
	Dropout			
5	Fully Connected	84	43	

Training the model:

I used Adam Optimizer as I found in some articles that it gives a lower training loss than other optimizers and requires little parameters tuning. link1, link2

I tried many times on parameters specially batch size, epochs and learning rate with the same Mu, Segma and dropout keep probability.
After many times of changing the parameters as some of them written in the table, I choose the selected set of variables with the rest of parameters to be:
Mu = 0, Sigma = 0.1, Dropout keep probability = 0.5
The result was this curve:

Epochs	Batch Size	Learning Rate	Accuracy
100	60	0.001	98.9%
20	32	0.001	98.5%
20	64	0.001	98.5%
20	128	0.001	98.1%
20	256	0.001	97.8%
25	256	0.001	97.9%
25	64	0.001	98.8%
25	32	0.0001	93.5%
25	32	0.001	98.9%
25	32	0.002	98.1%



My approach:

I started with the implementation of LeNet as the same one in Lesson 13 in the course and it turned out to work well, then I played around with the above parameters.

Epochs:

I started with 100 and submitted my first time with this without noticing that the accuracy didn't change after about 30 epochs, after playing around with the value again I observed that above 25 the accuracy seems to change little or never increases so I settled on 25.

Batch Size:

I started with 60 in my first submission and then after the review and some reading, I understood that it should be a power of 2, so I changed it and noticed that it is better to be low so I settled on 32.

Learning Rate:

I tried large numbers that gave me a low accuracy compared to low ones, so I lowered the number till I reached 0.001 after lowering the number less than 0.001 the network didn't reach the same accuracy given number of epochs so number of epochs must be greater, hence longer training time.

So, I choose the learning rate of 0.001, Epochs of 25 and Batch size of 32.

The new images test:

I searched for 10 new images resized then to 32x32 and tested them.
In the new images the model was challenged by some difficulties such as:

- Image contains part of another sign in images # 1 and 7.
- Image being a little bit elongated (scaled) like the image # 2.
- Image being tilted like the images # 4 and 8.
- Image with low brightness like images # 1 and 10.

The model got 60% of them right.

Using the model's SoftMax probabilities:

I used k=3 to see the top 3 most probable classes, and this was the output.