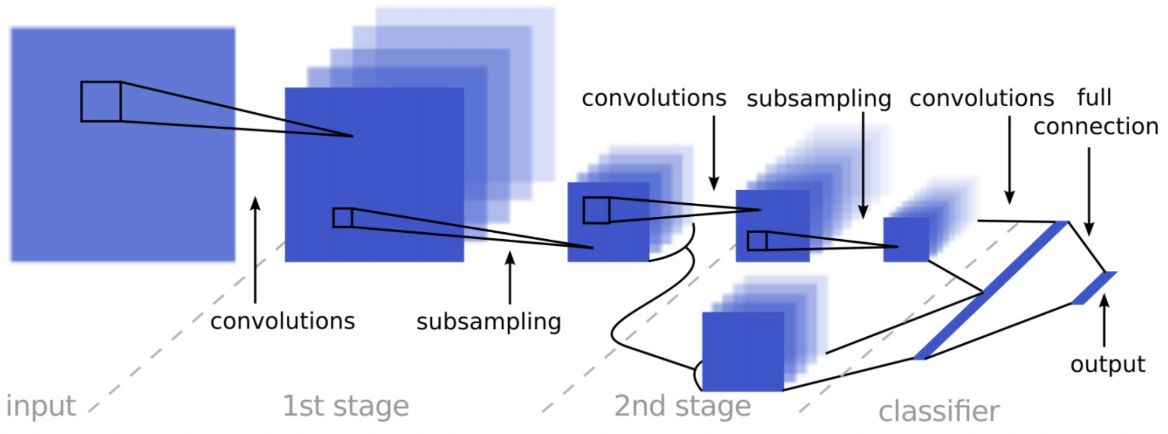
Traffic sign classifier write up

1) I used numpy to find summary statistics
    a. size of training set is: 27839 samples
    b. size of validation set is: 6960
    c. size of test set is: 12630
    d. shape of traffic sign image is: (32,32,3)
    e. number of unique classes in the data set is: 43
2) Here is how I visualized the data set. I displayed the first instance of every class. Then I created a histogram of the training data, then I created a histogram of the validation data. Since they look the same we can continue. I did notice that there were more representations from a few classes than others.

Design and test architecture

1) I preprocessed the image by grayscale and normalizing. This is because the project prompted me to try it. Upon discussing with the TA, it seems like putting it in grayscale doesn't do much. It might reduce the time it takes to train the system, but it isn't a noticeable difference. Another thing was normalizing the data. At first I didn't understand why my values weren't normalizing properly. I later discovered that the data set is in UTF-8 which is limited to the values 0 -255. Since all pixels are already in that range, normalizing isn't necessary either. I still implemented it for completion sake.
2) Here is my model architecture (modified LeNet)



Here are the layers:
Input (32x32x1)
1st stage 5x5 2d convolution (28x28x6)
ReLu
1st stage 2x2 max pool subsampling (14x14x6)
2nd stage 5x5 2d convolution (10x10x16)
ReLu
2nd stage 2x2 max pool subsampling (5x5x16)
1st stage flattener (1176)

2nd stage flattener (400)
Contacted fully connected layer (1576)
Dropout layer at p=50%
Output (43)

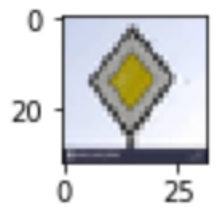
3) I trained the model with Adam optimizer. I used 45 epochs and a learning rate of 0.0009. I used a batch size of 128.

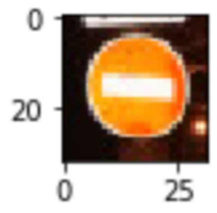
    Here are my final results
    Training set accuracy
    Validation set accuracy = 0.991
    Test set accuracy = 0.931

    At first I tried the basic LeNet architecture, but would only get a test set accuracy of 0.88. I wanted to do better so I read the paper that was presented with the project. The architecture seemed simple enough so I tried it. I got a tremendous boost in performance. I then decided to lower the learning rate to 0.0009, increase the number of EPOCHS to 45 and let it run while I do some other activity. Once I saw the numbers, I was content with the performance of my model and moved on.

4) Test model on new image

    Here are the five images I found on the web





Some of the images are off center and are in different lighting, but overall, they should work well with my model.

a. My model was able to guess 5 out of 5 of the signs correctly, giving me an accuracy of 100%. This is better than the error for the test set (only 93.1%).
b. If we look at the output of the softmax function, we can see that my model correctly predicts the signs with almost 100% certainty!

Image 1

Prob prediction sign

1.00E+00	12	Priority road
4.00E-10	5	Speed limit (80km/h)
1.99E-10	13	Yield
2.97E-11	35	Ahead only
1.86E-12	3	Speed limit (60km/h)

Image 2

prob	prediction	sign
1.00E+00	17	No entry
5.20E-14	14	Stop
1.77E-15	1	Speed limit (30km/h)
6.68E-16	33	Turn right ahead
1.74E-17	0	Speed limit (20km/h)

Image 3

prob	prediction	sign
1.00E+00	1	Speed limit (30km/h)
2.26E-06	2	Speed limit (50km/h)
1.84E-07	0	Speed limit (20km/h)
3.28E-10	4	Speed limit (70km/h)
3.76E-11	5	Speed limit (80km/h)

Image 4

prob	prediction	sign
1.00E+00	37	Go straight or left
5.53E-11	2	Speed limit (50km/h)
5.94E-12	35	Ahead only
3.14E-13	1	Speed limit (30km/h)
2.39E-14	8	Speed limit (120km/h)

Image 5

prob	prediction	sign
1.00E+00	18	General caution
9.08E-15	11	Right-of-way at the next intersection
1.11E-15	26	Traffic signals
4.81E-17	27	Pedestrians
1.87E-23	1	Speed limit (30km/h)