Preprocessing

For preprocessing I shuffled the data so it would be random. I also normalized it so that it would weight all features equally. I tried greyscaling the images but I didn't get good results with that and some images might have been recognizable because of their color such as the yield sign which is consistently yellow etc. So I took that part out.

Model Architecture

I began by implementing the same architecture from the LeNet Lab. However, after I added in a training set to the existing validation set, I saw that it was overfitting the data. So I added dropout.

The layers are set up like this

1. 5x5 convolution (32x32x1 in, 28x28x6 out)
2. ReLu
3. 2x2 max pool (28x28x6 in, 14x14x6 out)
4. 5x5 convolution (14x14x6 in, 10x10x16 out)
5. ReLu
6. 2x2 max pool (10x10x16 in, 5x5x6 out)
7. Flatten (400 out)
8. Fully connected (400 in, 120 out)
9. ReLu
10. Dropout
11. Fully connected (120 in, 84 out)
12. ReLu
13. Fully connected (84 in, 43 out)

Model Training

I based my model on the LeNet labs model. I tried several learning rates such as .001, .002 also added dropout to the second hidden layer but that made the validation accuracy worse. I also tried various numbers for dropout such as .3, .5 and .7 and found the best result in terms of getting the validation accuracy to be over 93% while still classifying some of the images correctly to be a rate of .5. I used 30 as the number of epochs although originally it was 10 but I saw accuracy was continuing to increase so I increased it to 20 and then to 30. I also increased it to 40 but then didn't see any further improvements so I keep it at 30.

Solution Approach

At first I didn't have a training set just a validation set but then I added in the training set and then could see if was overfitting the model. So I added dropout to the second layer and then added it to the first layer to see if that made things better. But adding to the first layer made the validation accuracy much worse so I just left it in the second layer.

Acquiring New Images

At first the model wasn't classifying any correctly. Then I decided to try with some other images I found online that I cropped myself which were speed limits. But I didn't find that it accurately predicted them. It correctly put them in the speed limit part but got the wrong speed limit consistently. The new speed limit one might have been difficult to classify because to my eyes I couldn't see what the number was just that it was a speed limit. Some of the other ones were more obvious to me such as the yield sign which is a pretty unique looking image and perhaps that's why it was the first one that was correctly classified. After that the right turn only image was correctly classified. I would have thought that it would be easy to classify the stop sign but it wasn't. Something interesting that I noticed was that sometimes it would predict the same label two or three times. That was a very bad classification because the signs aren't similar and aren't in the same category of images.

Performance on New Images

My test accuracy was 40%. I think if I had tried with more images it would have gotten a better accuracy because I think it has a difficult time accurately classifying speed limits. I tried with several speed limits and none was classified correctly although they were classified as speed limits. So I think perhaps it would have been better to try with ten images or more and see which ones it had an easier time classifying.

Model Certainty - Softmax Probabilities

Although 40% of the images were classified correctly. For one of the images that it classified correctly it was sure of it at 40.8%. It was sure of the other one at 16.1% which is a low certainty but it still got it right. Interestingly enough there was a certainty of 26.6% for the speed limit one which it got wrong.