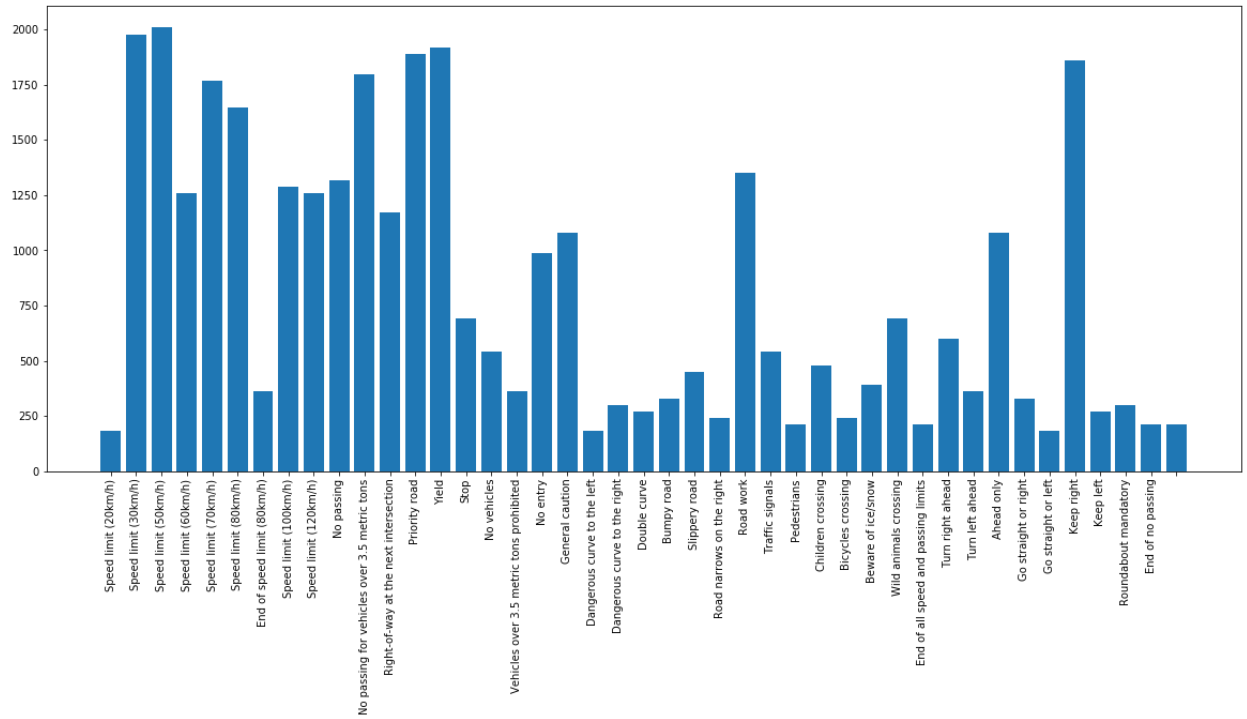
Meredith Meyer

Project 3: Traffic Sign Classifier

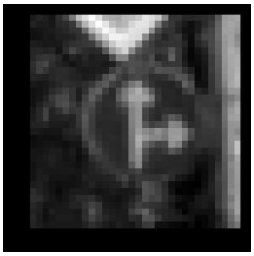
Data Set Exploration

To summarize the data set, I used both python built-in functions and the numpy shape attribute. The length of the y label arrays is equal to the number of samples, so I calculated the sample for each case (train, validation, test) that way. The shape of all the images are the same, so I used the first image in the test sample and used numpy shape to find that. The number of classes is the range of possible y values +1. A lot of focus was placed on the distribution of data used during training, so for visualization of the data I plotted a bar graph showing the frequency of each sign type within the training data set:



Design & Test of Model

First, I ran some pre-processing steps on the data. I gray-scaled each image, which eliminated the depth. I then applied an approximate normalization by applying (pixel - 128)/128 to each image. Because gray-scaling eliminated the depth of the data, I reshaped the images to have a depth of 1. See an example of image before and after gray-scaling:



Meredith Meyer

For the network, I used the framework of the LeNet model, altering the final size to account for the 43 classes in this traffic sign case. To achieve increased accuracy of the model, I primarily focused on the number of epochs, batch size, and learning rate.

In general I had trouble getting the accuracy above 90% and holding. Even with many epochs, the accuracy did not seem to increase and stabilize. I believe this to be a result of over-fitting. Therefore, I experimented with adding a dropout layer to the model, and increasing batch size. In general, I had the best results with a dropout rate of 0.003-0.005, 50-100 epochs, and a sample size of 200.

After many, many iterations, I was able to achieve an accuracy for the validation set of over 93%. My final model used 30 epochs with batch sizes of 150, and 40% dropout layer, and a learning rate of 0.003.

Supplemental Images

I chose 5 German traffic sign images from the internet. My model had trouble classifying the signs from the internet- my test accuracy was usually 20%. Additionally, the probabilities for the classifications were low. I would guess that it is associated with the low quality of the images, or warping in the preprocessing steps. Ideally, I'd like to investigate this further.