Title: Traffic Sign Segmentation & Classification

Members: Henry Mound & Ben Vandenbosch

Summary:
We created a script (create_dataset.py) that vectorizes each datapoint, consisting of two features, and outputs them to a file. This script reads in from 'signDatabasePublicFramesOnly/allAnnotations.csv' and parses the following information for each row in the csv: the path to the image, the (x1, x2, y1, y2) coordinates of where the traffic sign occurs in the image, and the type of sign in the image. The script then proceeds to load the image using OpenCV, crop the image to the specified area including the traffic sign, and resizes the image to a specified size (we are using 224px x 224px for now). Finally, the script serializes a string version of the processed image as well as the traffic sign label (assigned an ID 0 through 46). Using TensorFlow, this data is then encoded and outputted as a line in our dataset so that it can be read by our classifier.

The TensorFlow script , build_model.py, is still essentially an adapted version of a tutorial found on YouTube/GitHub. It reads the data encoded by the create_dataset script and creates/trains a CNN. We plan to begin experimentation with hyperparameters now that we have built our interface between the data and the TensorFlow and tailor code to meet our needs (you can see the commented out experiments/work in progress at the ned of the file). Though our classification rates after running for a few minutes are very low right now, that would be expected based on our dataset and how long it takes to train such a model. Lastly, we have begun to research and look at image segmentation for the first part of our pipeline that we plan to build.

Results:

Dataset Summary:

Signs	**Examples**
addedLane	294
curveLeft	37
curveRight	50
dip	35
doNotEnter	23
doNotPass	9
intersection	2
keepRight	331
laneEnds	210

merge	266
noLeftTurn	47
noRightTurn	26
pedestrianCrossing	1085
ampSpeedAdvisory20	11
rampSpeedAdvisory35	5
rampSpeedAdvisory40	3
rampSpeedAdvisory45	29
rampSpeedAdvisory50	16
rampSpeedAdvisoryUrdbl	3
rightLaneMustTurn	77
roundabout	53
school	133
schoolSpeedLimit25	105
signalAhead	925
slow	34
speedLimit15	11
speedLimit25	349
speedLimit30	140
speedLimit35	538
speedLimit40	73
speedLimit45	141
speedLimit50	48
speedLimit55	2
speedLimit65	74
speedLimitUrdbl	132
stop	1821
stopAhead	168
thruMergeLeft	5
thruMergeRight	7
thruTrafficMergeLeft	19
truckSpeedLimit55	60

turnLeft	32
turnRight	92
yield'	236
yieldAhead	57
zoneAhead25	21
zoneAhead45	20

Initial Results from Model:
Our initial classification accuracy is less than 1 percent,

Sample Output (After a few minutes of running):

```
Classification accuracy: 0.25%
{'accuracy': 0.0025461488, 'loss': 36.5961, 'global_step': 0}
```

Problems:
The major problem thus far has been learning how to integrate with and use TensorFlow. Now that we have figured it out (or at least enough to begin working with it meaningfully), we will begin to adjust hyperparameters and test different network architectures to figure out what works best with our model. As can be seen in the current codebase, our current architecture is based on a tutorial we found on YouTube/GitHub and has not been adapted specifically to our needs. With the data structured and interface to TensorFlow solved, we can turn our focus to the ML concepts.

Secondly, our network has an abysmal classification rate right now, partially due to the fact that there are 47 categories, some of which only have a handful of examples. To solve this, we will likely begin to recategorize, dropping some of the categories with only a few examples and lumping some subcategories into larger categories (for example, use all speed limit signs as one category rather than breaking them down by the actual limit).

Hours:
Henry: 7 hours
Ben: 7 hours
Code: GitHub Repository (https://github.com/henrymound/ml_final_project/)