Traffic Sign Classification Writeup:

Dataset Exploration

1) Dataset Summary

There were 2 separate datasets. The first download consisted of train.p and test.p. This dataset was then updated to train.p/test.p/valid.p. These datasets are not the same.

2) Exploratory Visualization

Exploratory Visualization of dataset statistics:

Created histograms for valid.p/test.p/train.p showing the number of examples/class. There are varying samples per class.

Exploratory visualizaiton of preprocessing steps:

Visualized test images to see the effect of image processing and observerd the following:

1) the images are dark. Used np.histogram to fix this with default=False. There are 2 settings for default; True and false. Plotted the histogram for default=False. For true the peak shifts closer to 0.

2) to create jittered images as specified in the paper which used translation of [-2,+2] pixels and rotation [-15,+15] degrees. The paper used the Y channel; substituted grayscale conversion per project instructions.

Design and Test a Model Architecture

Preprocessing:

There are 2 options; feed the complete 32x32 scaled images as provided into the model or do a processed grayscale of 32x32x1 with added rotated/translated/brightness adjuste images.

Model Architecture:

Model Training:

Solution Approach:

Doing nothing and inputting a normalized gray scale gets accuracy of .935. This is an ok metric but it is better to generate loss/validaiton graphs to show the degree of overfitting. The loss will increase as the degree of overfitting increases. This can be used to tune and modify the network.

Test a Model on New Images

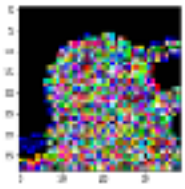
Acquiring New Images
Performance on New Images
Model Certainty - Softmax Probabilities

1) float64 is default after image processing w/opencv. Convert to float32 to match declaration of x and y in tensorflow input variables.
2) skimage vs. opencv. skimage not work b/c of scaling images to 0-1 vs. 0-255. Can expand range to get this to work.
    skimage code test:

```
# X_train; add rotated, translated, brightness, contrast.
#what is layout. Get all images of class0 first
from skimage.transform import rotate

def rotImage(image,label):
    img=[]
    labels=[]
    rot_img = skimage.transform.rotate(image, 15, resize=True,
                        center=None, order=1, mode='edge', cval=0, clip=False,
preserve_range=True)
    img.append(rot_img)
    labels.append(label)
    rot_img1 = skimage.transform.rotate(image, -15, resize=True,
                        center=None, order=1, mode='edge', cval=0, clip=False,
preserve_range=True)
    img.append(rot_img1)
    labels.append(label)
    return img,labels

#rot_images, labels = rotImage(X_train[0],0)
#plt.imshow(rot)

rot = []
labels=[]
num = 0
for i in range(0,43):
    for idx, img in enumerate (X_train):
        if y_train[idx]==i:
            #print (idx)
            r,l = rotImage(img,i)
            rot.extend(r)
            labels.extend(l)
```

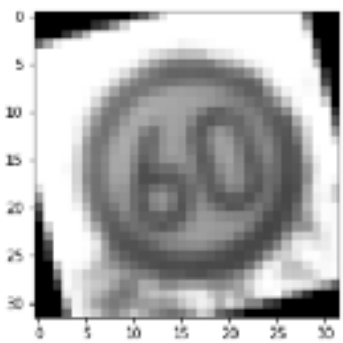
```
            num += 1
    print (len(rot), len(labels),num)
#print (len(rot), len(labels),num)
print(labels[10000])
plt.imshow(rot[10000])
```

This is the distorted 20km/h street sign with the skimage transform parameters. If we use default we don't get this distortion. Was an attempt at testing effect of parameters.



This shows a nondistorted skimage version. note the range of output between 0-1. Note: our normalziation contains 0 which is bad.

```
#0-1 displays same as 0-255. Range is messed up using skimage. Fix by expanding range and compare to opencv
rot_np = np.array(rot)
print(np.amin(rot_np[10000]), np.amax(rot_np[10000]))

print(rot_np.shape)
rot_gray=rgb_gray(rot_np)
plt.imshow(rot_gray[10000],cmap='gray')
print(np.amin(rot_gray[10000]), np.amax(rot_gray[10000]))
print(np.amin(X_train[0]), np.amax(X_train[0]))

#rot_histo = apply_hist(rot_gray)
#plt.imshow(rot_histo[10000])


0.0 1.0
(69598, 32, 32, 3)
0.0 1.0
19 113
```



3) Experiment on opecv rotation. In color, the rotation produces black edge artifacts as the rotation angle increases. If you convert to grayscale this artifact is still there but is less visible.

Displays the progression showing the rotation, crop then resize. The rotation shifts the image. To remove the black edge artifacts you have to crop which increases the size; then expand back to 32x32 after the crop and reduce scale by .9 to negate the magnification effect. This was done by eyeball test. Not measured.

4) The output of TF it not deterministic. Follows a distribution. Run 10x and plot distribution. if this is a normal distribution we can use mean to describe the accuracy.
5) The dataset was changed from train.p/test.p to train.p/test.p/valid.p and the contents were different. Test if the same images are in both. They are not the same.
6) Histogram test using np.histogram.
7) folow dropout from paper https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf [.9,X,]