Traffic Sign Recognition

Writeup

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following: * Load the data set (see below for links to the project data set) * Explore, summarize and visualize the data set * Design, train and test a model architecture * Use the model to make predictions on new images * Analyze the softmax probabilities of the new images * Summarize the results with a written report

Rubric Points

The output of the Python Jupyter Notebook is saved as HTML – filename in the github is "Traffic_Sign_Classifier.html". It is based on the same name Jupyter Notebook that has all the code cells executed.

Load The Data Set

I loaded the data set in code cell 1. They were classified into three categories: Train, Valid, Test. The 'test' data set won't be used till I was done with getting an acceptable result from the 'valid' data set. I also tried to combine the train and valid data set to create a more random set of data – but this iteration did not result in significant improvement in the accuracy so I abandoned this approach.

Data Set Summary & Exploration

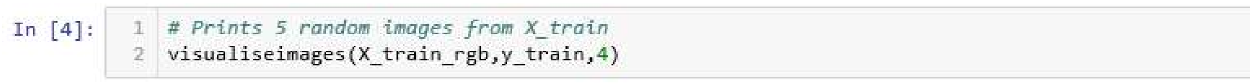
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

Refer to code cells 1 and 2 – this is where I explored the data set.

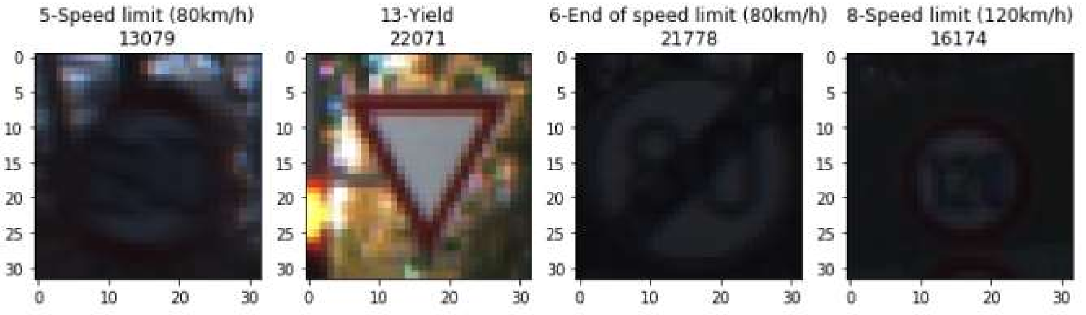
- The size of training set is 34799 samples
- The size of the validation set is 4410 samples
- The size of test set is 12630 samples
- The shape of a traffic sign image is 32x32x3 (RGB image)
- The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

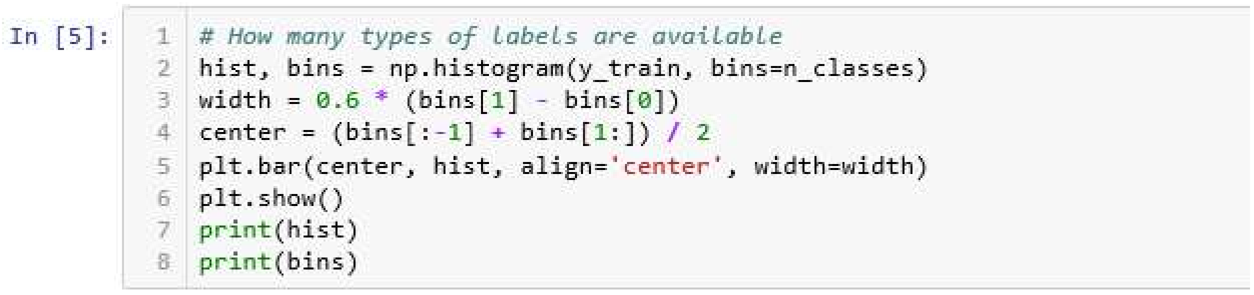
In the code cell 3, I wrote a 'visualiseimages' function which receives a image array and a variable specifying how many images need to be displayed. This function is used repeatedly throughout the next code cells to keep tabs on the progress. Here is an example output from this:



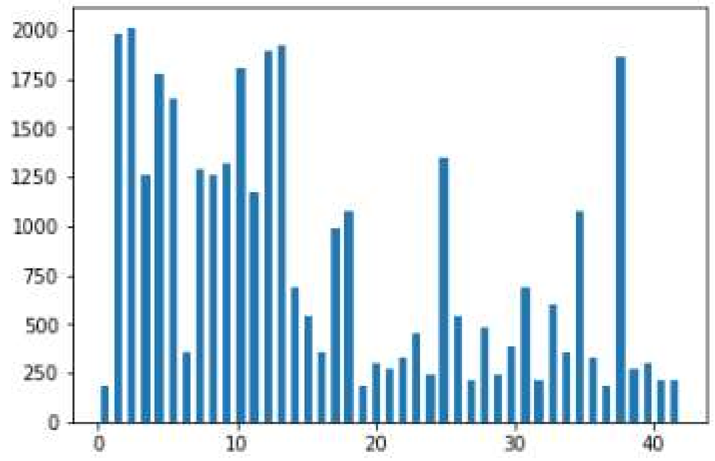
```
In [4]:
# Prints 5 random images from X_train
visualiseimages(X_train_rgb,y_train,4)
```



In order to visualize the distribution of the classes amongst the training data set, I created a histogram (refer code cell 5). This clearly shows that all the 43 classes don't have similarly representative population of images in each of them:



```
In [5]:
# How many types of labels are available
hist, bins = np.histogram(y_train, bins=n_classes)
width = 0.6 * (bins[1] - bins[0])
center = (bins[:-1] + bins[1:]) / 2
plt.bar(center, hist, align='center', width=width)
plt.show()
print(hist)
print(bins)
```



```
[ 180 1980 2010 1260 1770 1650  360 1290 1260 1320 1800 1170 1890 1920  690
  540  360  990 1080  180  300  270  330  450  240 1350  540  210  480  240
  390  690  210  599  360 1080  330  180 1860  270  300  210  210]
```

Design and Test a Model Architecture

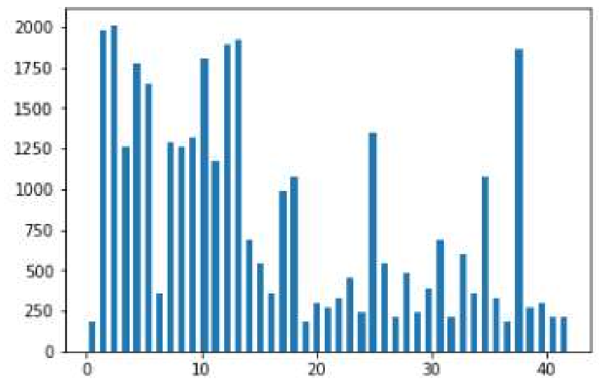
1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

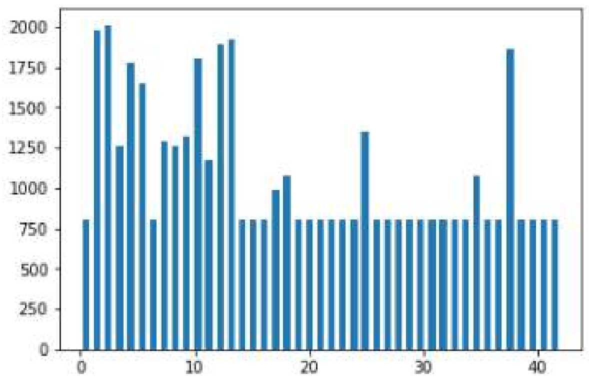
As the average number of samples per class(y_train) are 802, I decided that each class that has number of samples less than 800 need to be concatenated with some data to make it more representative. What kind of data to concatenate? :

No point concatenating the same kind of data. So I will experiment with pasting the same data set as the class but only slightly modifying the images - shift the image a few pixels, add a gaussian blurr, darken/brighten image. Do one of these randomly.

Refer code cells 8,9,10 where I wrote functions that either brighten, shift or blurr a given image in a random manner. The randomness ensures that learning is improved. When I tested the model without randomness and with randomness, I obtained about 2-3% improvement in accuracy. Although this also instills a variability in the final output every time you run the mode. This can be seen in the markdown cell after code cell 26.

Code cell 12 is where I ran the training image data set through my concatenation algorithm. Here are the results (comparing it to the histogram above):



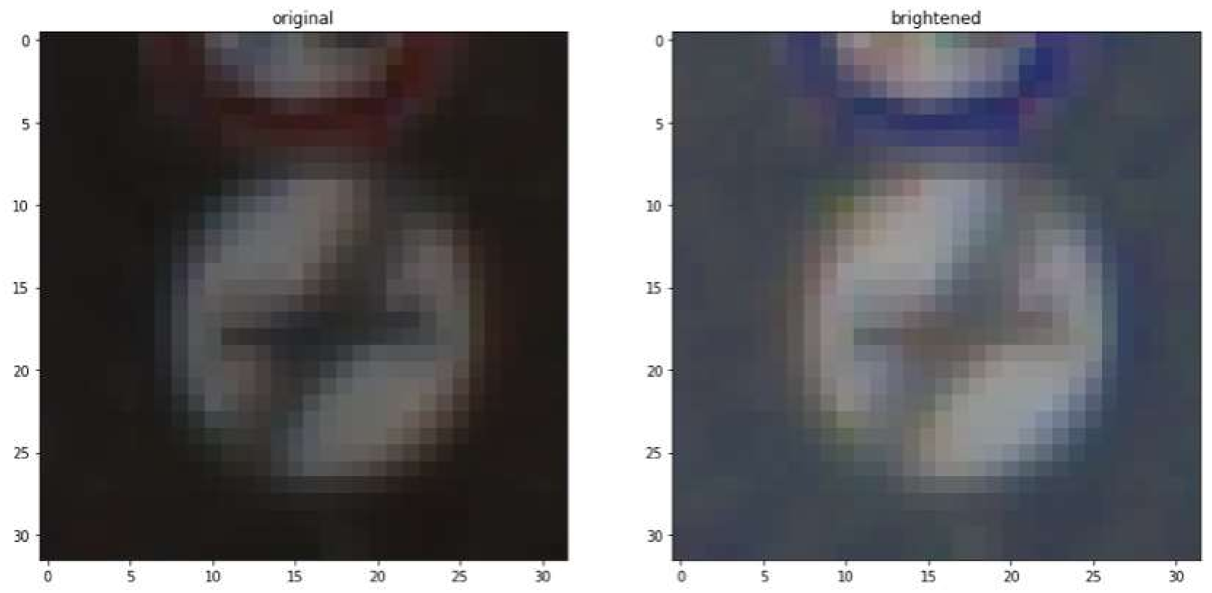


As can be seen, the bins which had underrepresentation of training images are now filled up to 800 count with randomly modified images.

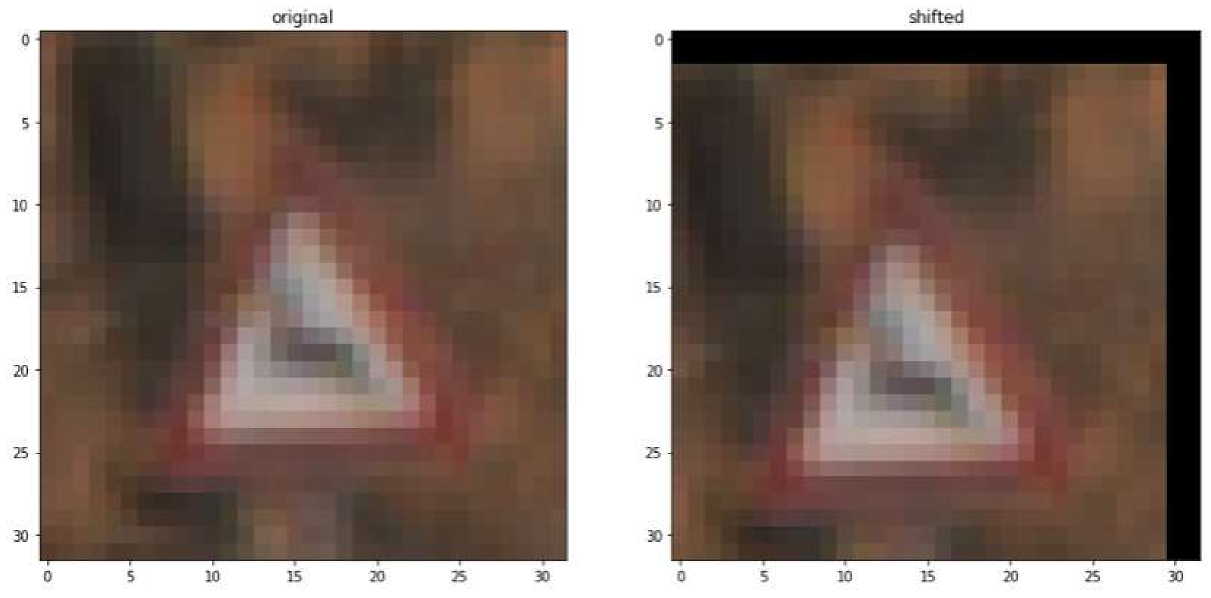
After this the total training data set increased to **46480 from original 34799**

Here is an example of augmented images that were added to the training data set:

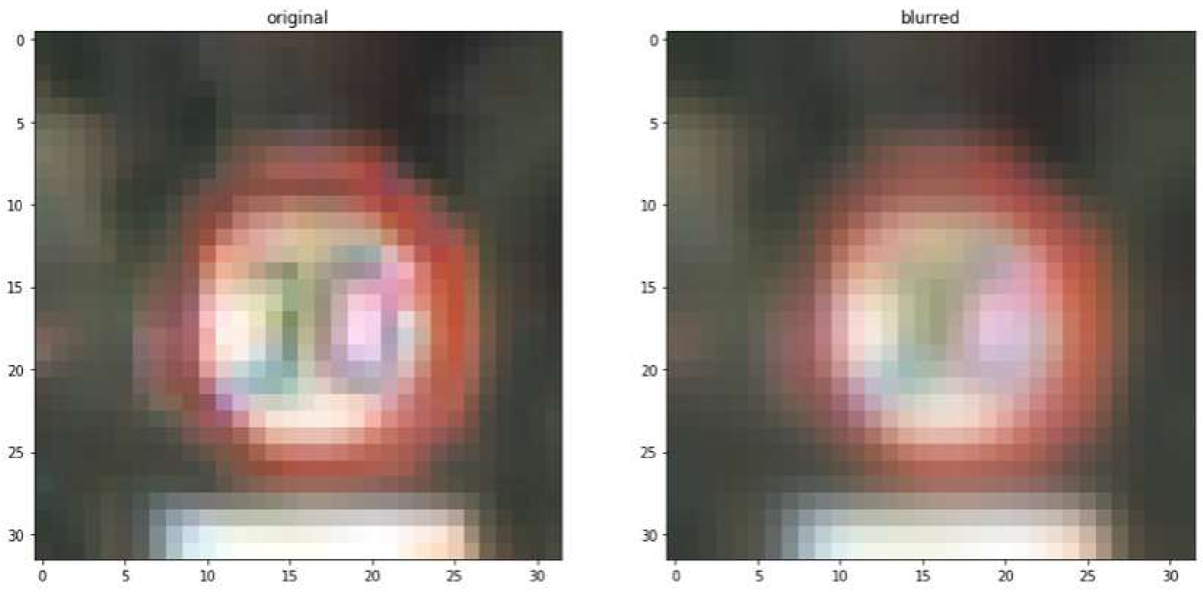
Brightening the image (randomly chosen brightening factor – I convert the image to HSV space using CV2 and then change the V value):



Shifting the image (randomly chosen dx,dy value and then warpAffine function):



Blurr the image (Gaussian blurr using a random sized Kernel)



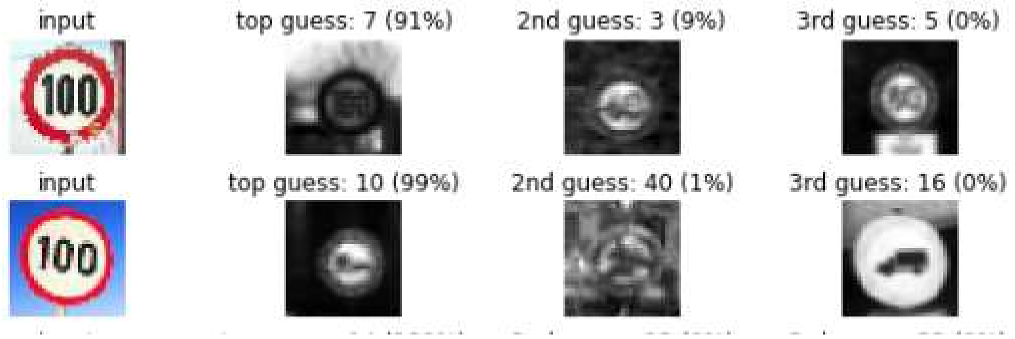
In the final algorithm to augment the training data instead of just putting the image through just one function, I randomly choose whether to:

Shift(Brighten(image))

Blurr(shift(image))

Brighten(Blurr(image))

After completing the project where I put the images I found through the trained network, I observed an image of the 100 kmph was slightly tilted. This was not recognized by the network. I realized that in the above three combinations, I should have included a tilt function as well.



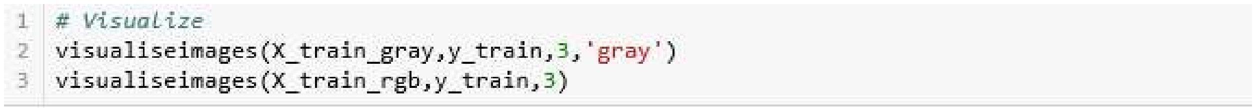
As can be seen here, the first 100 kmph sign is recognized but the second one which is tilted is not.

After augmenting the training image data there are two further steps I took:

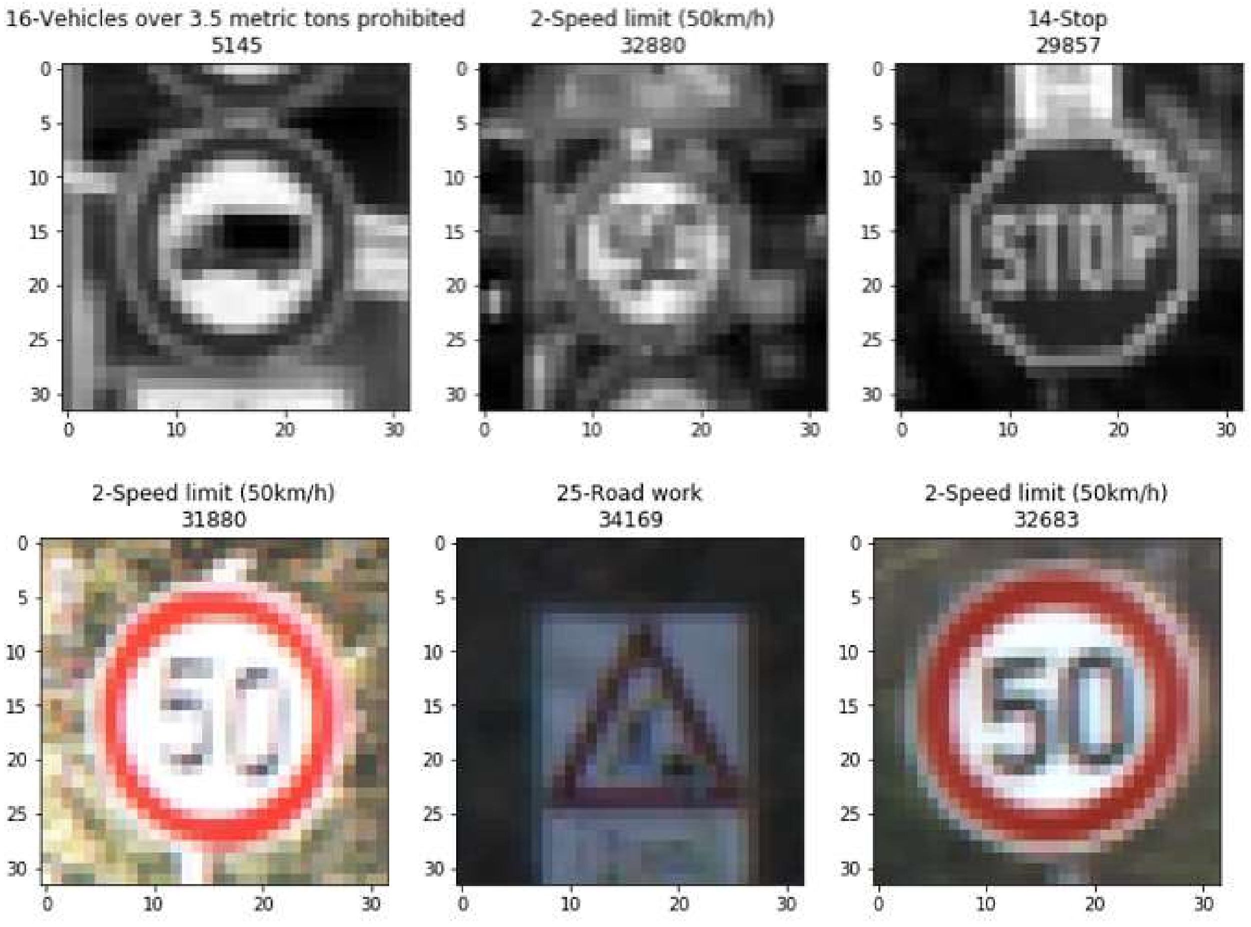
- Convert to gray scale – this helps in reducing the training time as the image content (class) is mainly in how the data is distributed in the array rather than the color space data

- convert the gray scale image to a normalized image in the range (-1, 1): This was done to better organize the data to a smaller mean (changed from ~84 to ~-0.3 for a grayscale image)

Here are some examples of converted images (after and before conversion):



```
# Visualize
visualiseimages(X_train_gray,y_train,3,'gray')
visualiseimages(X_train_rgb,y_train,3)
```



The final training data set size was (46480, 32, 32, 1)

2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

My final model consisted of the following layers:

Layer	Input	Output
1. Convolution 5 x 5	32x32x1 – Original image	28x28x 6
2. ReLu	28x28x6	28x28x6
3. Max pooling 2x2	28x28x6	14x14x6
4. Convolution 5 x 5	14x14x6	10x10x16
5. ReLu	10x10x16	10x10x16
6. Max pooling 2x2	10x10x16	5x5x16
7. Flatten-1	5x5x16	400
8. Convolution on step 6	5x5x16	1x1x400
9. ReLu on step 9	1x1x400	1x1x400
10. Flatten-2	1x1x400	400
11. Concatenate 7 and 10	400, 400	800
12. Dropout (prob:0.5)	800	800
13. Fullu connected	800	43

3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

The provided pipeline was used: This consisted of sending the original image to the LeNet network to obtain the logits. These logits are then sent to workout the cross entropy using the tensorflow softmax and the one hot encoded classes. The cross entropy is used to calculate ht loss (using reduce_mean) and finally a Adams optimizer is used to minimize the loss.

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.

My final model results were: training set accuracy of 95.9% validation set accuracy of 95.4% test set accuracy of 94.2%

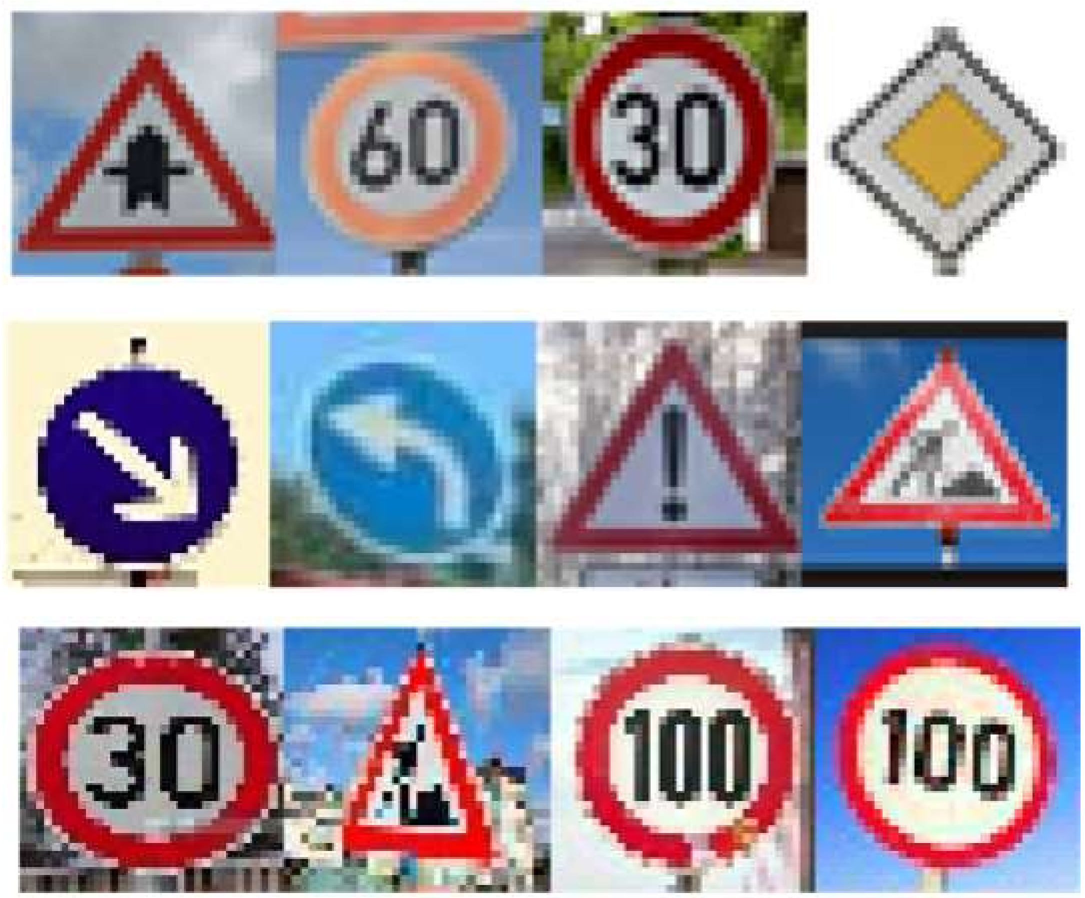
The details of the iterations that I conducted are shown in the markdown cell after code cell 24. In short, initially I started off with the stock LeNet function that was provided without any image modification or dataset augmentation. I got 87.9% accuracy on the validation set. I ran the training step by step as shown in the markdown cell by incorporating normalization, gray scale conversion, changing epochs, training rate, adding concatenate and dropout layer to the LeNet function. Finally I changed the image modification functions to include randomness which provided me additional 2-3% accuracy improvement.

After this I basically played with epochs, dropout rates, batch size and learning rate to achieve final result of 95% (best I got was 97% at epoch 30 in a 60 batch run). By the time of submission my Amazon Web services g2.2xlarge GPU limit increase had not been approved, so iterations were slowwwwww. This limited my ability to some extent. But even with available resources, I achieved a test dataset accuracy of 95.4%

Test a Model on New Images

1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

Here are German traffic signs that I found on the web:

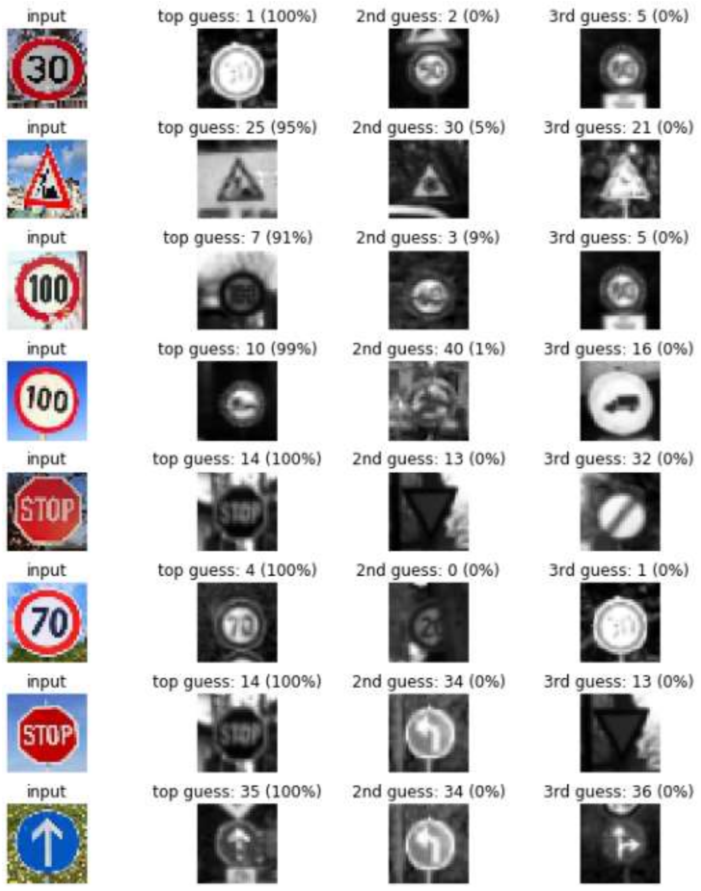


There are two data sets of 8 images each. In both cases the network model was not able to recognize the images which were tilted (60 kph in first set and 100 kph in second set).

My plan now is to include a tilting function in the image modification set which I use at the time of data set augmentation. As I ran out of time and my training speed was very slow, I will be doing this as a continuous improvement.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

Here are the results of the prediction:



The model was able to correctly guess 7 of the 8 traffic signs, which gives an accuracy of 87.5%. If the tilting image modification is included, it will be interesting to see if the 100kph tilted image will also be included to get me to 100%.

3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

The code for making predictions on my final model is located in cell 34 of the python notebook.

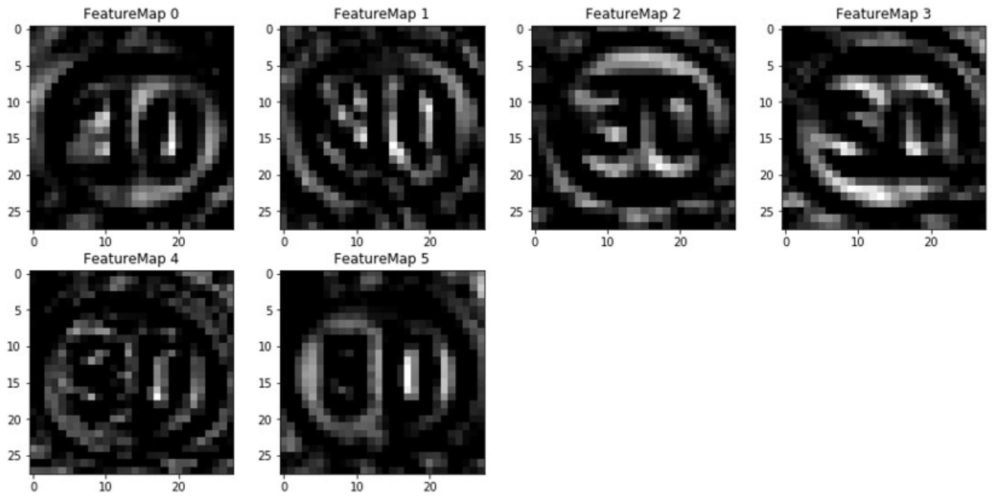
Out of 8 images 7 were correctly recognized and softmax probabilities as shown above are 100% for top 5.

(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

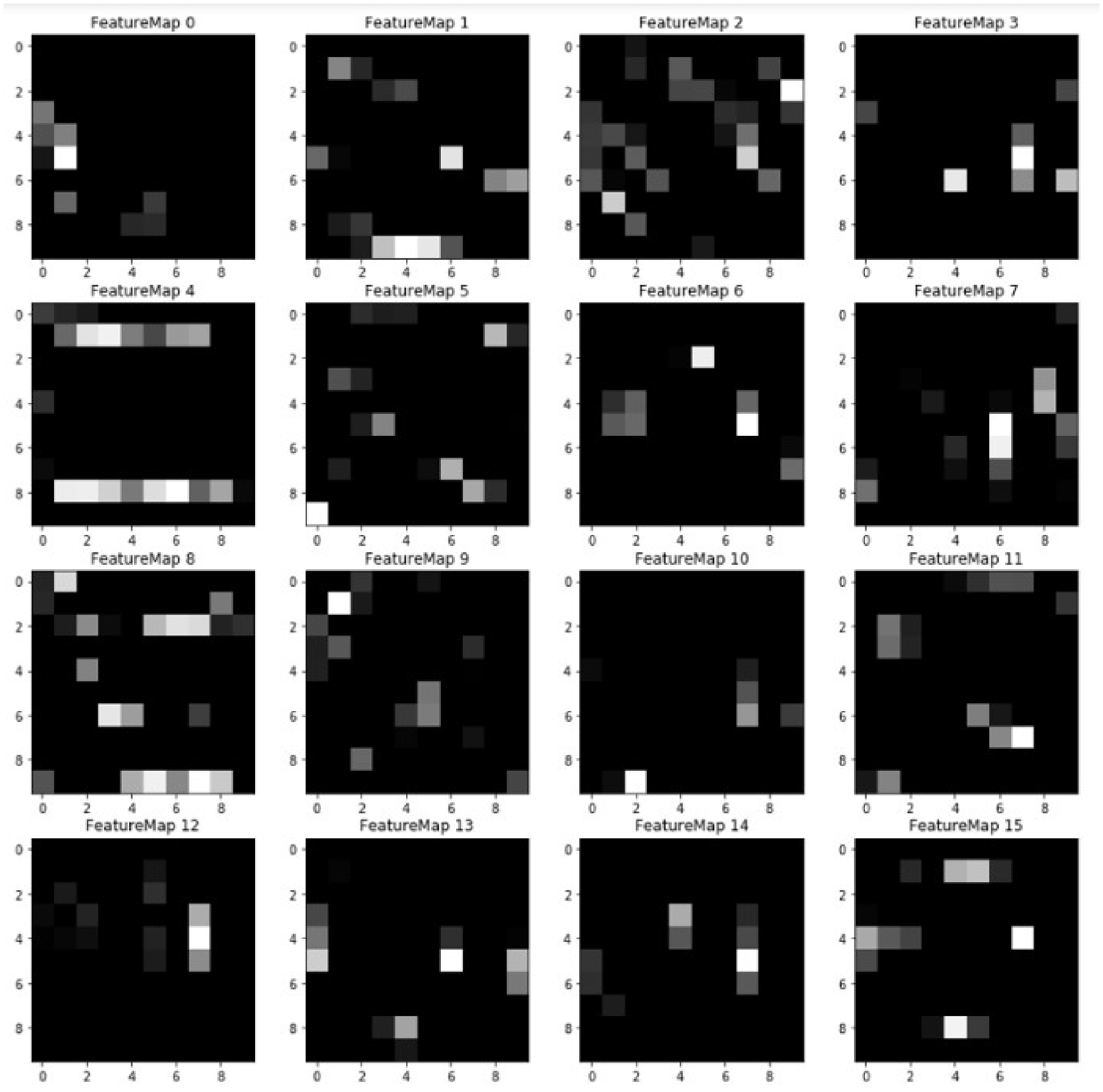
1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?

Original image:

Layer – 1 (output of 6 layers):



Layer – 2 (16 layer output):



I am showing the result of the first and second convolution layer whose output is 6 and 16 layers respectively. The image on top is the original image sent to the outputFeatureMap function.

From the outputs of the two layers, its really hard to tell which features eventually influence the decisions. In the layer-1 I can see that it is detecting the edges of the features. I am currently reading a few papers to get a better understanding into how features influence decisions. For example:

https://arxiv.org/pdf/1507.02313.pdf