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
Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

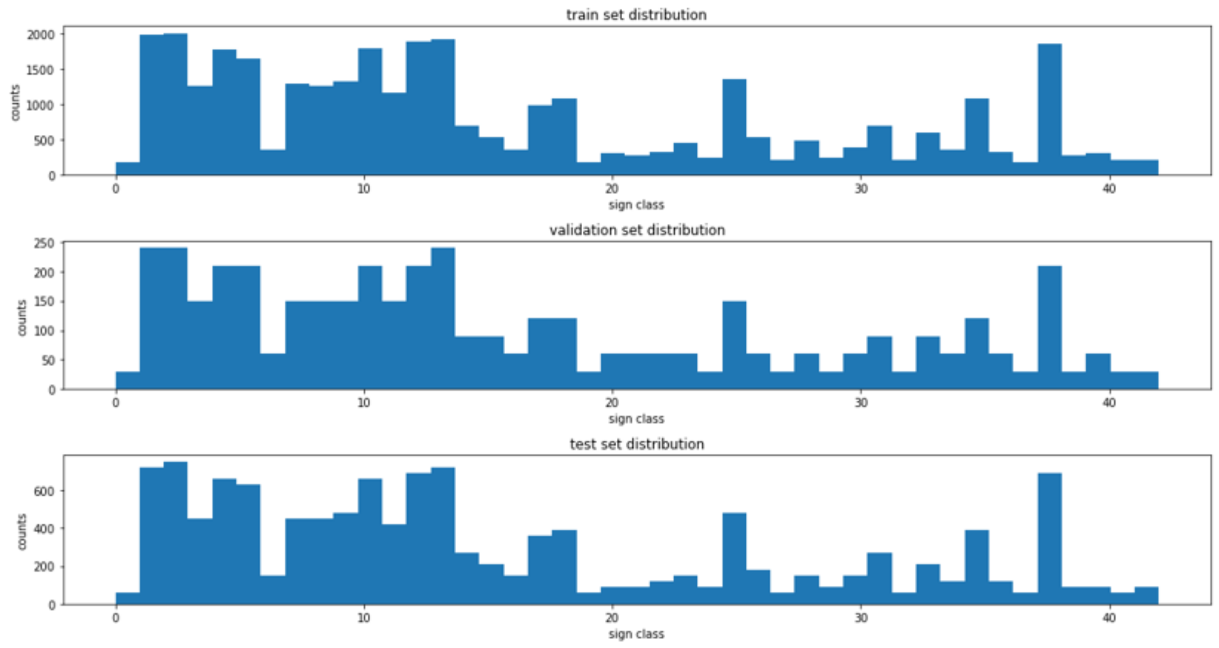
Writeup / README
1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.
You're reading it! and here is a link to my project code

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.
I used the pandas library to calculate summary statistics of the traffic signs data set:

- The size of training set is ? -- 34799
- The size of the validation set is ? --4410
- The size of test set is ? -- 2630
- The shape of a traffic sign image is ? -- (32 , 32 , 3)
- The number of unique classes/labels in the data set is ? --43

2. Include an exploratory visualization of the dataset.
Here is an exploratory visualization of the data set.

- The counts distribution of original train set, validation set and test set:



- Samples of some test set images:

Design and Test a Model Architecture

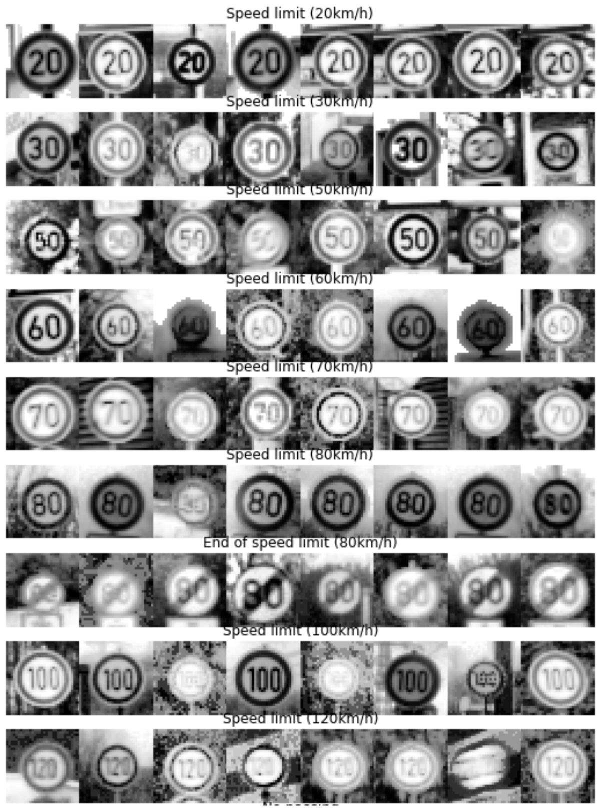
1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques

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.)

The steps I preprocessed one image:

- Convert RGB to Gray scale image since sample images are under different light and weather, some of them are very dark which is not like the original color.
- Equalize the gray scale image to adjust the contrast ratio and try to strengthen the edges of the traffic signs
- Normalize the equalized image to make it easier to converge during training
- Code:

```
def preprocess_one_img(img):
    #Convert RGB to Gray
    gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)

    #Equalize
    equalized_img = cv2.equalizeHist(gray_img)

    #normalize
    min_p, max_p = np.min(equalized_img), np.max(equalized_img)
    normalized_img = ((equalized_img - min_p)/(max_p - min_p))*2 - 1

    return np.expand_dims(normalized_img, axis = 2)
```



I decided to generate additional data because the count distribution of the trainset is not equal, some sign classes have more than 2000 images while come classes have only about 100 images.

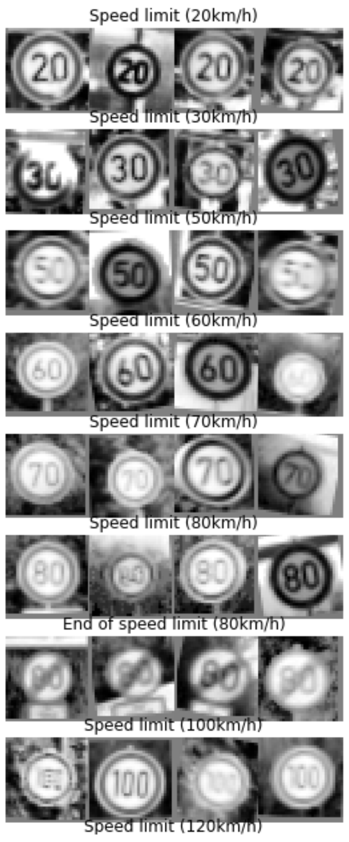
The main techs used on image augmentation are:

- Random bright
- Rotation
- Translate

```
def image_augmentation(img, eff=0.4, angle=12, pixels=2):
  #random bright
  eff = eff + np.random.random()
  img = img*eff

  #rotate
  angle = np.random.randint(-angle, angle)
  M = cv2.getRotationMatrix2D((16,16), angle, 1)
  img = cv2.warpAffine(src=img, M=M, dsize=(32,32))

  #translate
```

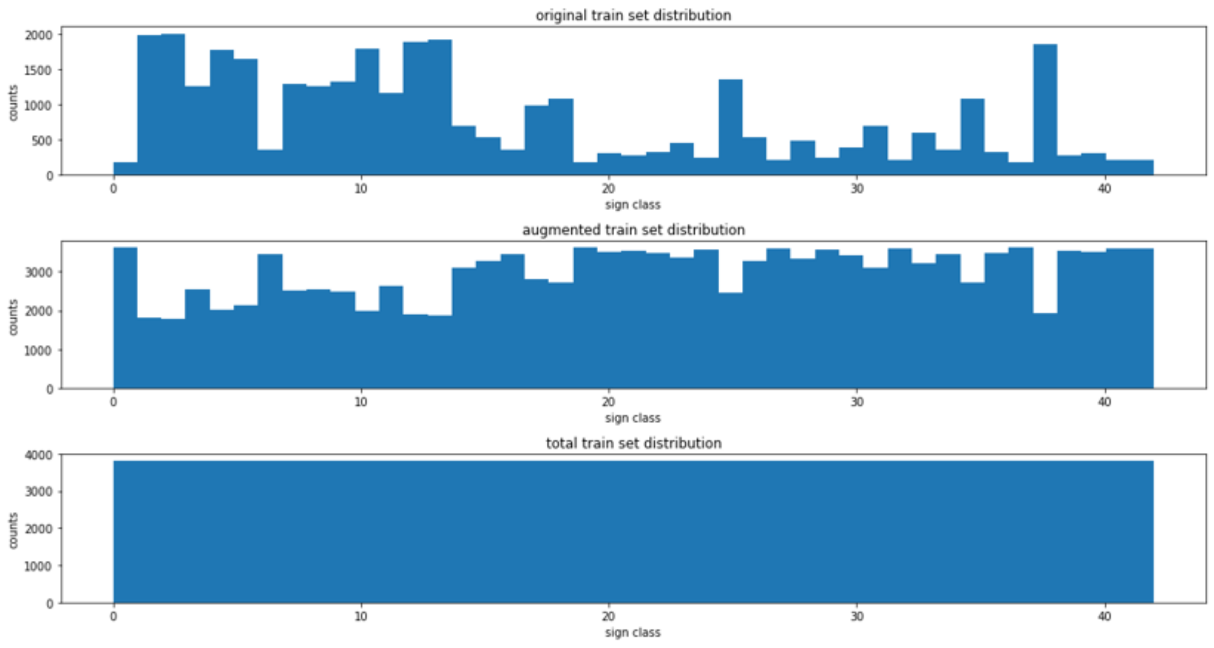
```
    px = np.random.choice(range(-pixels, pixels))
    py = np.random.choice(range(-pixels, pixels))
    M = np.float32([[1,0,px],[0,1,py]])
    img = cv2.warpAffine(src=img, M=M, dsize=(32,32))

    #normalize
    min_p, max_p = np.min(img), np.max(img)
    img = ((img-min_p)/(max_p-min_p))*2 - 1

    #expand dims
    img = np.expand_dims(img, axis=2)

    return img
```

Samples for augmented images:



Each class generate 3800-counts images to make the total counts for each class are equal.

The counts distribution for original train set, augmented images and total train set:



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	Description
Input	32*32*3 RGB Image
Conv1:5*5	1*1 stride, valid padding, output:28*28*6
relu1	
pool1	2*2 stride, outputs: 14*14*6
Conv2:5*5	1*1 stride, valid padding, output:10*10*16
relu2	
pool2	2*2 stride, outputs: 5*5*16
flatten	Output: 400
Dropout1	
fully	Input:400, output:120
Relu3	
Dropout2	
output	Input 120, output 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.

To train the model, I used an ...

```
#hyper parameters

batch_size = 32

keep_prob = .5

epochs = 100

patience = 8

initial_lr = 0.0005
```

I use exponential decay learning rate so that the network could train fast in the beginning and slow when close to the minmum loss.

```
learning_rate = tf.train.exponential_decay(
    learning_rate=initial_lr,
    global_step=global_step,
    decay_steps=ceil(n_train / batch_size),  # Decay every epoch
    decay_rate=0.95,
    staircase=True).
```

I use cross entropy and add softmax for the loss and use AdamOptimizer which is easy to adjust the parameters.

```
    cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=one_hot_y)

    loss_operation = tf.reduce_mean(cross_entropy)

    optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)

    training_operation = optimizer.minimize(loss_operation)
```

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 ? -- 0.982

- validation set accuracy of ? -- 0.974
- test set accuracy of ? -- 0.948

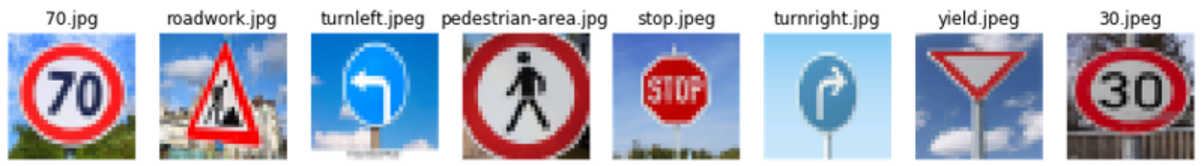
Iterative approach:

1. First I used the LeNet structure since it is used to classifier 10 handwritting numbers, the conv layer are very strong to extract features from the images, however the trainning results are not good at the beginning
2. Then, I add augmented images, the trainning accuary is much better than validation accuary which means the network is overfitting.
3. And then, I remove the second fully connection layer since there are 43 classes, change the input of the output layer to 120 directly, the trainning accuracy dropped which indicate there is underfitting
4. And then, I removed the dropout layers after the conv layers
5. And then, I adjust the total number of the augmented training set of each class, and the eff, angles and learning rate, batch sizes

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.

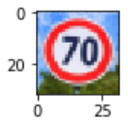
Here are five German traffic signs that I found on the web:



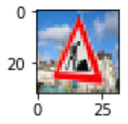
The fourth image might be difficult to classify because the image count is only about 100 in the orignal trainning set and most of them are in dark environment

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).

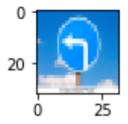
Here are the predictions:



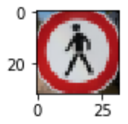
```
Actual: Speed limit (70km/h)
Predicted: Speed limit (70km/h)
```



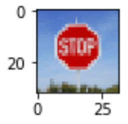
```
Actual: Road work
Predicted: Road work
```



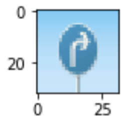
```
Actual: Turn left ahead
Predicted: Turn left ahead
```



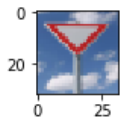
```
Actual: Pedestrians
Predicted: General caution
```



```
Actual: Stop
Predicted: Stop
```



```
Actual: Turn right ahead
Predicted: Turn right ahead
```



```
Actual: Yield
Predicted: Yield
```

```
Actual: Speed limit (30km/h)
Predicted: Speed limit (30km/h)
```

The model was able to correctly guess 7 of the 8 traffic signs, which gives an accuracy of 87.50%. It is not as good as the test set accuracy I guess it is because the total number is too small and the images downloaded from web site have different distribution with the trainning set images

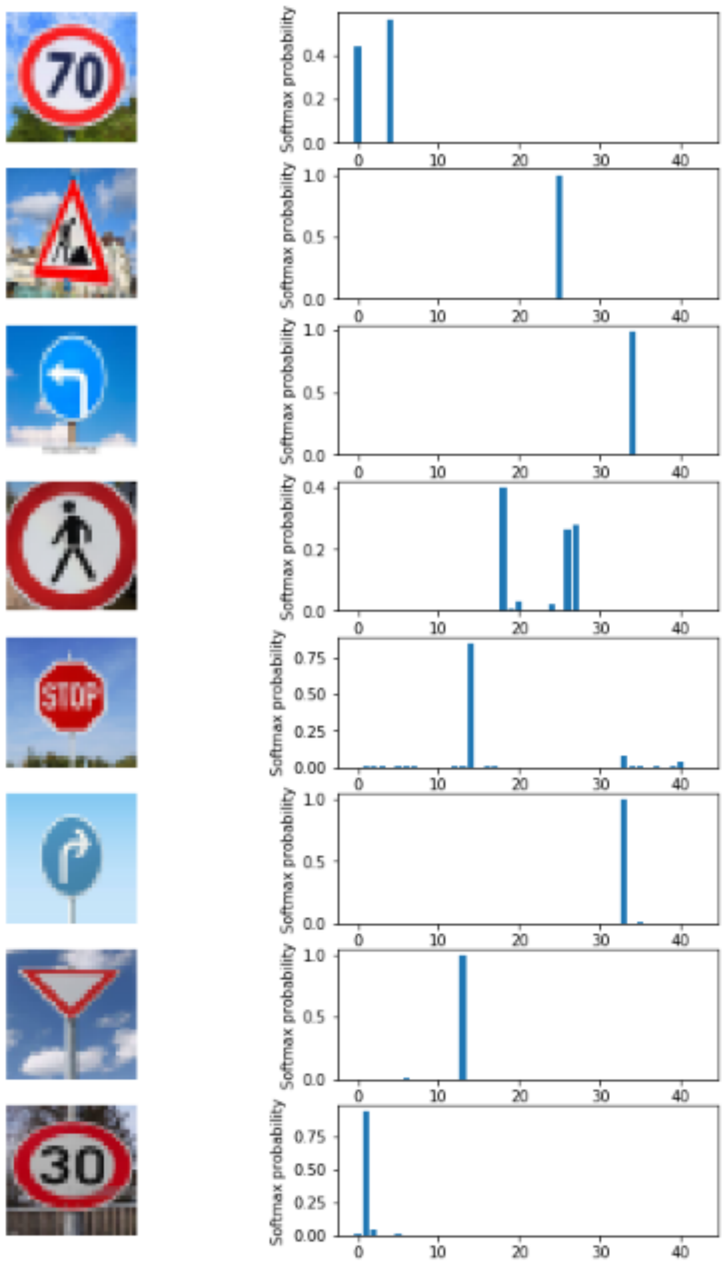
3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions"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

Here are the Top5 probilities:

The top 1 probility are very obviously for second, third, fifth, sixth, senveth, eighth images while the top 1 probility of first and fourth images are close to the top2/top3 problities.

The first image is speed limit -70 which is very close to speed limit 20 on the image.

The fourth image pederstrian has only about 100 images on the original training set and most of them are in very dark environment.

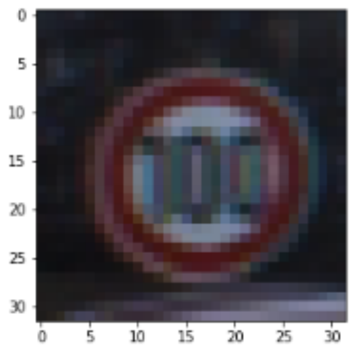


(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

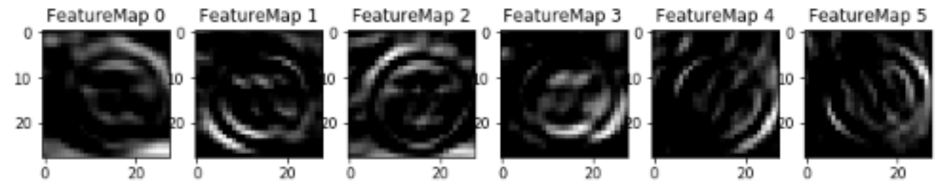
1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?

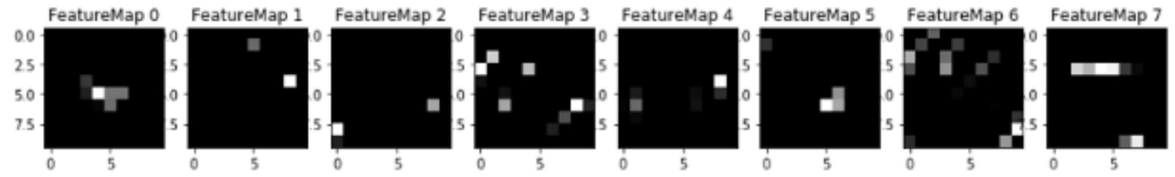
The circle and the number 1, 0, 0

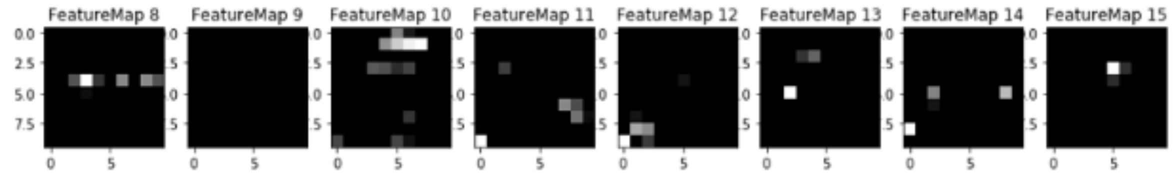
```
Feature maps for Speed limit (100km/h)
```



```
First convolutional layer
Second convolutional layer
```


FeatureMap 0
FeatureMap 1
FeatureMap 2
FeatureMap 3
FeatureMap 4
FeatureMap 5


FeatureMap 0
FeatureMap 1
FeatureMap 2
FeatureMap 3
FeatureMap 4
FeatureMap 5
FeatureMap 6
FeatureMap 7


FeatureMap 8
FeatureMap 9
FeatureMap 10
FeatureMap 11
FeatureMap 12
FeatureMap 13
FeatureMap 14
FeatureMap 15