#Traffic Sign Recognition

##Writeup Template

###You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

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.

####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.

###Data Set Summary & Exploration

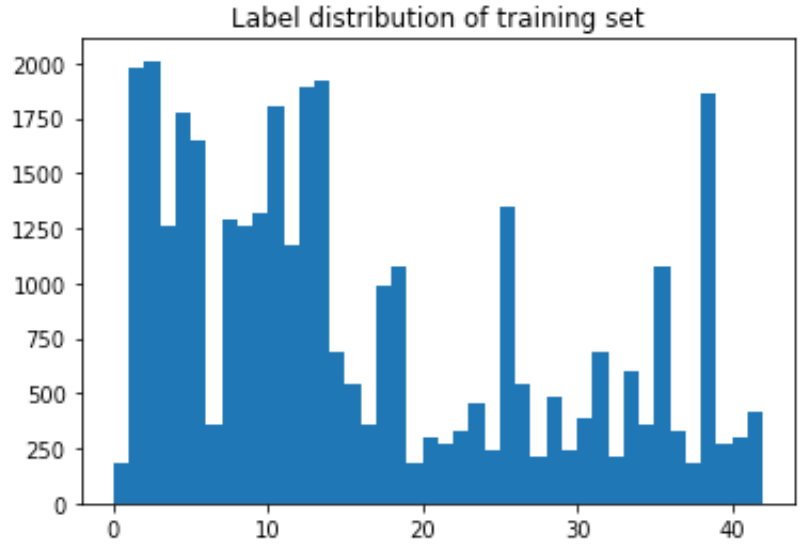
####1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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

- **The size of training set is : 34799 . After adding data to training set to achieve a more evenly distribution of number of pics in each class, the size of training set is 46714**
- **The size of the validation set is : 4410**
- **The size of test set is : 12630**
- **The shape of a traffic sign image is ?**
- **The number of unique classes/labels in the data set is ?**

####2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data distribution



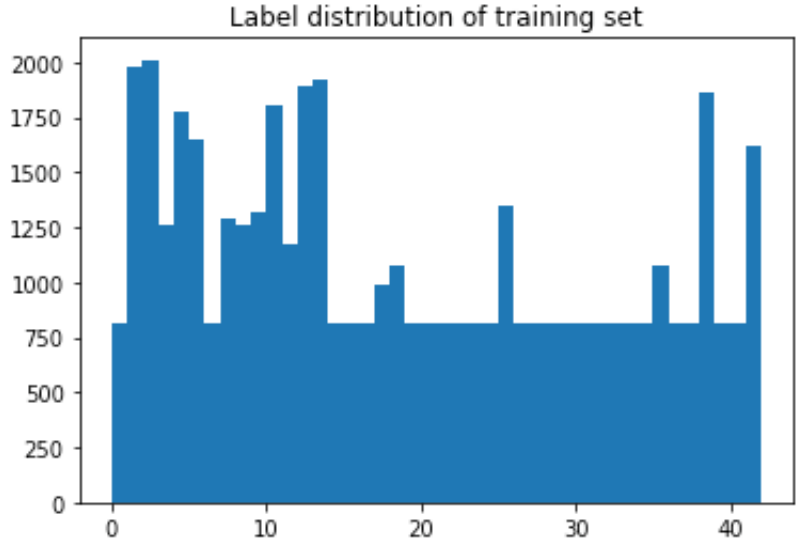
###Design and Test a Model Architecture

####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 a first step, I decided to convert the images to grayscale , and add 1 extra dimension using newaxis because I need 32*32*1 image
As a last step, I normalized the image data because it's better for Neural Network to work with value that not too large. I choose to normalize data such that it's between [-0.9, 0.9]. I think that works giving our neural network hyper parameter is mu = 0 and sigma = 0.1 So I want my data to have both negative and positive value

I decided to generate additional data because the number of images in each class are not evenly distributed in training set, as shown above

To add more data to the the data set, I used the following techniques : I compute desired number of pictures per class by taking average across different classes. For those classes with number of pics less than the desired number, I added more pics into the class by randomly select 1 pic and slightly modify it by rotating it. After adding more data, the distribution of training data look like this. Adding more data like this does improve my validation accuracy. Without adding data, my validation accuracy hover around 92% , using the same model architecture



####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:
When I applied original LeNet architecture after pre-process the images

and adding more data into training set, I noticed that the model seems to be a bit overfitting (Training accuracy is much higher than Validation Accuracy. Validation accuracy is around 88% for all epochs). So I add dropout with keeping prob = 0.7. Drop out help my validation & test accuracy get into 90% level.
After trying average pool instead of max pool, my accuracy of validation and test are improve d and reach 93%
I think my model could be improved future by using L2 Regularization, because I can see my Training Accuracy is quite high compared to validation accuracy, which means my model is still a bit over-fitting

####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.

Learning Rate = 0.004. I tested all value between 0.001 and 0.005. 0.003 and 0.004 works best.
Batch Size = 128, Epochs = 10
Hyper parameter:mu = 0, sigma = 0.1
Optimizer = AdamOptimizer

####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...

EPOCH 1 ...
Training Accuracy = 0.893
Validation Accuracy = 0.843

EPOCH 2 ...
Training Accuracy = 0.961
Validation Accuracy = 0.895
```

```
EPOCH 3 ...
Training Accuracy = 0.977
Validation Accuracy = 0.902

EPOCH 4 ...
Training Accuracy = 0.981
Validation Accuracy = 0.912

EPOCH 5 ...
Training Accuracy = 0.984
Validation Accuracy = 0.925

EPOCH 6 ...
Training Accuracy = 0.990
Validation Accuracy = 0.925

EPOCH 7 ...
Training Accuracy = 0.989
Validation Accuracy = 0.929

EPOCH 8 ...
Training Accuracy = 0.986
Validation Accuracy = 0.922

EPOCH 9 ...
Training Accuracy = 0.991
Validation Accuracy = 0.933

EPOCH 10 ...
Training Accuracy = 0.993
Validation Accuracy = 0.939

Test Accuracy = 0.927
```

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?
- What were some problems with the initial architecture?
- How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

- Which parameters were tuned? How were they adjusted and why?
- What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?

Answer:
1. LeNet original model is tried first. Validation accuracy is around 87-88%
2. Try Drop out, that improve to 89-90%
3. Adding more data given uneven distribution of number of pics in each class in original training data. That improve validation accuracy to 91-92%
4. Try using average pooling instead of max pooling, achieve accuracy 93%

###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: Read my Jupyter Notebook for image plot
They are 5 traffic signs in order: Stop, Traffic Light, Children Crossing, Yield,Speed Limit 30 km/h

####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 4 of the 5 traffic signs, which gives an accuracy of 80%. The last pic is a speed limit 30 km/h sign and the classifier fail. Classifier predict it’s a speed limit 50 km/h. The Speed Limit 30km/h softmax is 2nd highest for the last pic. Maybe softmax is not the best activate function to evaluate? (I don’t know but the architecture is set up using softmax cross entropy to measure loss so it’s supposed to work)

Compare to Test Set accuracy (92.%), the new prediction accuracy is lower. It might due to overfitting in my training data (as mentioned above, L2 Regularization may help).

####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)

```
TopKV2(values=array([[  9.99838471e-01,   9.71260961e-05,   2.3141901
7e-05,
          1.70083003e-05,   1.47641504e-05],
       [  9.48050737e-01,   5.19492626e-02,   1.89372532e-10,
          7.01867753e-16,   3.18499770e-24],
       [  9.99913692e-01,   5.29925455e-05,   2.03488926e-05,
          4.41085376e-06,   2.56094313e-06],
       [  1.00000000e+00,   2.80177747e-12,   3.73093565e-13,
          8.35991229e-14,   3.48192002e-14],
       [  6.12002730e-01,   3.72724831e-01,   1.52428094e-02,
          2.60035395e-05,   3.47842206e-06]], dtype=float32), indices
=array([[14,  2, 15, 38, 33],
       [26, 18, 37, 24, 40],
       [28, 36, 30, 24, 26],
       [13, 12, 35, 33, 26],
       [ 2,  1,  5,  4,  3]], dtype=int32))
Image  0 predicted classed: [14  2 15 38 33]
 and predcted class name: ['Stop', 'Speed limit (50km/h)', 'No vehicl
es', 'Keep right', 'Turn right ahead']

Image  1 predicted classed: [26 18 37 24 40]
 and predcted class name: ['Traffic signals', 'General caution', 'Go
straight or left', 'Road narrows on the right', 'Roundabout mandatory
']

Image  2 predicted classed: [28 36 30 24 26]
 and predcted class name: ['Children crossing', 'Go straight or right
', 'Beware of ice/snow', 'Road narrows on the right', 'Traffic signal
s']

Image  3 predicted classed: [13 12 35 33 26]
 and predcted class name: ['Yield', 'Priority road', 'Ahead only', 'T
urn right ahead', 'Traffic signals']

Image  4 predicted classed: [2 1 5 4 3]
 and predcted class name: ['Speed limit (50km/h)', 'Speed limit (30km
/h)', 'Speed limit (80km/h)', 'Speed limit (70km/h)', 'Speed limit (6
0km/h)']
```