#**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

[//]: # (Image References)

[image1]: ./examples/visualization.jpg "Visualization"
[image2]: ./examples/grayscale.jpg "Grayscaling"
[image3]: ./examples/random_noise.jpg "Random Noise"
[image4]: ./examples/placeholder.png "Traffic Sign 1"
[image5]: ./examples/placeholder.png "Traffic Sign 2"
[image6]: ./examples/placeholder.png "Traffic Sign 3"
[image7]: ./examples/placeholder.png "Traffic Sign 4"
[image8]: ./examples/placeholder.png "Traffic Sign 5"

Rubric Points
###Here I will consider the [rubric points](https://review.udacity.com/#!/rubrics/481/view) 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](https://github.com/udacity/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb)

###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 12630
* The shape of a traffic sign image is 32x32x3
* 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. It is a bar chart showing how the data ...

![distribution][distribution.png]

###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.)

I did not convert to grayscale to avoid losing any information. But normalized the data converting to floating point

####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 network has essentially the same structure as LeNet from the lab. The only differences were the following:
First, I modified the first layer to accept depth-3 (color) images instead of depth-1 images.
Also added a dropout layer after each activation (relu) layer

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Layer 1: Convolution 5x5	1x1 stride, same padding, outputs 28x28x6
RELU	
Dropout	tunable parameter

Max pooling	2x2 stride, outputs 14x14x6
Layer 2: Convolution 5x5	1x1 stride, same padding, outputs 10x10x16
RELU	
Dropout	tunable parameter
Max pooling	2x2 stride, outputs 5x5x16
Flatten	Input 5x5x16, output 400
Layer 3: Fully connected	Input 400, output 120
RELU	
Dropout	tunable parameter
Layer 4: Fully connected	Input 120, output 84
RELU	
Dropout	tunable parameter
Layer 5: Fully connected	Input 84, output 43 (labels)

The output of the above layers was the logits. After which I computed the loss function supplied to the optimizer, I took the cross entropy of softmax(logits) with the one-hot-encoded labels. The loss was defined to be the average of the cross entropy across the batch.
The dropout parameter was identical for all dropout

####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 Adams optimizer and had below parameters:
EPOCHS = 40
BATCH_SIZE = 128
rate = 0.001
dropout = .75

Changed dropout to 1.0 for validation and testing

####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 99.9%
* validation set accuracy of 94.1%
* test set accuracy of 93.4%

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?

If a well known architecture was chosen:
* What architecture was chosen?
* Why did you believe it would be relevant to the traffic sign application?
* How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

I began with the LeNet architecture from the lab. LeNet proved very effective at classifying black-and-white handwritten digits, so it was plausible to expect that it could identify street sign images, which are of similar overall complexity.
I first modified LeNet to accept an input of color depth 3 and yield an output of 43 classes instead of 10. As a "zero order optimization" I tried simply dialing up the number of training epochs to 40 to see where the validation accuracy would plateau (or potentially peak and decline due to overfitting). Unfortunately, I never observed validation set accuracy of greater than 94%, although the accuracy on the test set climbed to 99.9%ish. This led me to believe that the network was mildly overfitting.
To combat overfitting, I implemented a dropout layer after each relu activation, by adding a tunable hyperparameter. 0.75 value gave good validation accuracy.

###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:

![alt text][bicyclecrossing] ![alt text][childrencrossing] ![alt text][nopassing]
![alt text][roadwork]

The first image might be difficult to classify because the curves of bicycle may be confused with the slippery sign which also has a triangle shaped sign

####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:

Image	Prediction
bicyclecrossing	slippery road
no passing	no passing
go straight or right	go straight or right
road work	Right-of-way at the next intersection
children crossing	Beware of ice/snow

The model was able to correctly guess 2 of the 5 traffic signs, which gives an accuracy of 40%.

####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 the 11th cell of the Ipython notebook.

For the no passing, straight or right, the model is very certain.
For others, the network is less certain of its true sign.