Project 2. Traffic Sign Recognition.
Writeup.

Goal of the project: Build a Traffic Sign Recognition classifier

The steps of this project are the following:

- Load the 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

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

Writeup

Provide a Writeup, 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.

Current document itself represents a Writeup. Also here is a link to the project code implementation and all required resources for testing it.

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.

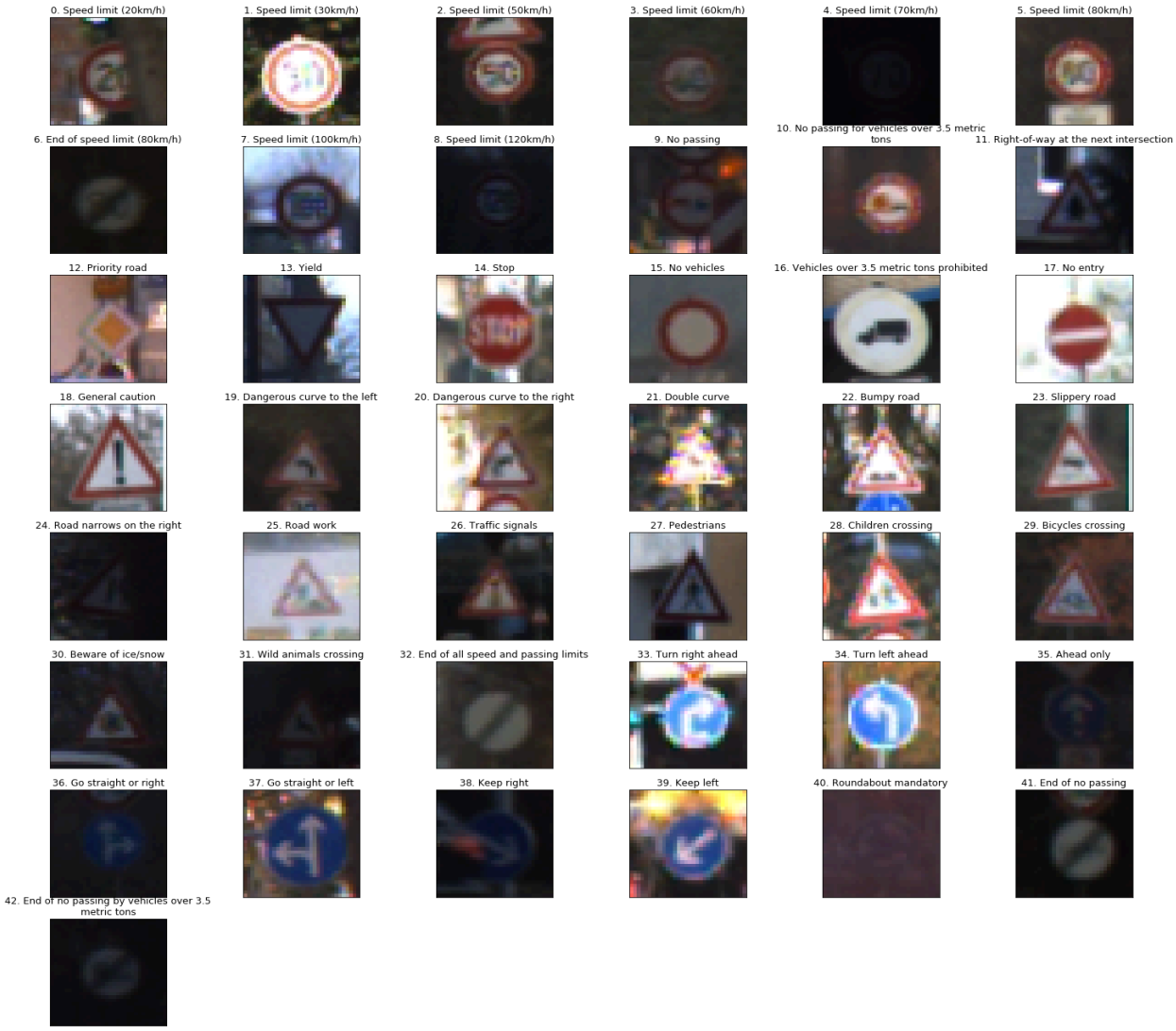
I used the pandas library to calculate summary statistics of the traffic signs data set:

- The size of training set is `34799 images`
- The size of the validation set is `4410 images`
- The size of test set is `12630`
- The shape of a traffic sign image is `32x32x3 (32 pixels height/width x 3 (rgb) channels)`
- 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 (implemented with means of **matplotlib.pyplot** library), which is split into the histograms showing the number of images of certain classes of traffic signs in training, validation and testing sets . Vertical axle of each histogram represents the number of images of certain class in given set. Horizontal axle shows the numerical class IDs of corresponding traffic sign (in ascending order. 43 in total)

Below is also shown the collection of samples of all classes of traffic signs represented in the data set. Classes of images in a table are sorted by numerical ID of the class. All IDs of the classes together with corresponding traffic sign names (labels) are stored in a dictionary document signnames.csv. Each class of traffic sign is represented by the first occurred image belonging to this class in a validation data set. (with this in mind it is expected that all 43 classes of traffic signs are represented in this dataset)



Design and Test a Model Architecture

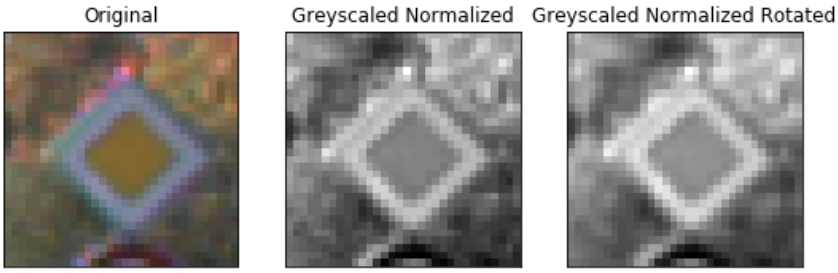
1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described 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 in all 3 data sets (training, validation, testing) to greyscale form, because this will decrease the amount of required time for training the model and amount or required memory (2d matrix instead of 3d) without significant loss in training/validation/testing accuracy, since in my opinion for classification of such images as traffic signs, color does not bring much value, only shape and content do.
- Also I have decided to normalize the pixel values in the images in all 3 data sets in order to achieve zero mean and equal variance across pixels of each dataset. This should speed up the training process. For the normalization of images following technic was used: I have calculated the mean of pixel value across whole training data set and used it as the normalization factor for all 3 datasets. Normalization of pixel values in a data sets was done using following formula:

```
normalizaed_pixel_value = (pixel_value – normalization_factor) / normalization_factor
```

- Additionally following preprocessing technics were applied only to training data set:
    - All images of greyscaled normalized training dataset were rotated on small random angle (each image to it's own angle) in a range of [-5 : 5) degrees and resulting rotated set was added to initial (preprocessed) training set. Labels (class IDs of images) of training set were duplicated to match the updated size of feature set. I experimented with various rotation degree ranges from -/+ 5 to -/+30. The range [-5 : 5) degrees seems like show the best gain for classification accuracy.
    - Additionally all images in the training set are shuffled for each epoch of training to minimize the bias that might be caused by (predefined) order of images (by some criteria) in the initial training set. Just as a preventive measure.

Here is an example of a traffic sign image transformation steps that is applied to every image of training data 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 consists of the following layers:

Layer	Description	Input	Output
Convolution 5x5	stride: 1x1, padding: valid, activation: ReLU	32x32x1	28x28x6
Max pooling	stride:2x2, window: 2x2	28x28x6	14x14x6
Convolution 5x5	stride: 1x1, padding: valid, activation: ReLU	14x14x6	10x10x16
Max pooling	stride: 2x2, window: 2x2	10x10x16	5x5x16
Flatten	transforms 3D matrix into 1D array	5x5x16	400
Fully connected	connects all inputs to all outputs, activation: ReLU, dropout 25%	400	120
Fully Connected	connects all inputs to all outputs, activation: ReLU, dropout 25%	120	84
Fully Connected	connects all inputs to all outputs	84	43

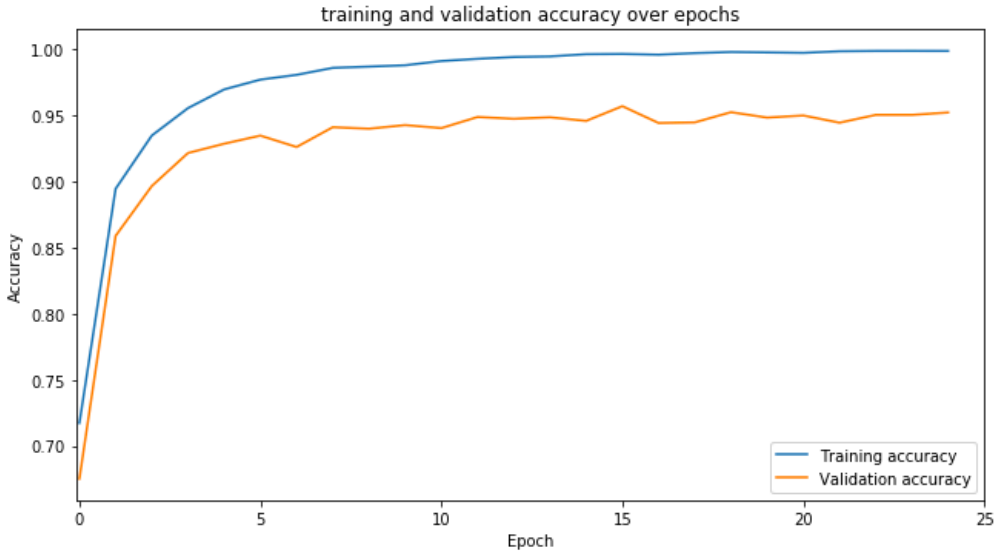
3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

To train the model, I used used 25 epochs, with batch size of 128 and a learning rate of 0.0003. mu and sigma for random generator of initial weights and biases were equal 0 and 0,1 accordingly. As the optimizer I decided to use AdamOptimizer, which seems to show better performance than Gradient Descent Optimizer.

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 implementation of training model is based on LeNet-5 architecture, with further incremental improvement (iterative approach). With original training dataset and 'default' values of hyperparameters (10 epochs, 0.001 learning rate, 128 batch size) it gave me the validation accuracy of about 90%. After the preprocessing of whole dataset and feeding it into the model I got about 92% accuracy on validation set. After augumentation of training dataset and still using the same hyperparameters I got about 94% accuracy on validation set. To find the best combination of hyperparameters, I tried several values for number of epochs (15, 25, 50, 100) with combination of following values values for learning rate accordingly: (0.0005, 0.0003, 0.0002, 0.0001). The batch size I decided to keep unchanged. The best results for accuracy (99.8% on training dataset, 95.2% on validation dataset and 93.7% on the test dataset) I got with 25 epochs and

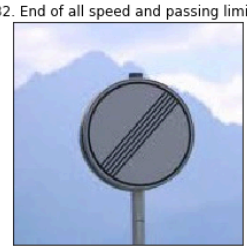
learning rate 0.0003. Also among several tested values for dropout 20%, 25%, 30%, the 25% has shown the best results for model accuracy. Taking into account a difference of more than 4% between accuracy on training set and validation set, I believe that there is a slight overfitting of the model. Having this in mind as further steps for improvement I think it would be worth playing with a sizes of training and validation datasets (e.g. increase the validation set size by taking a part of images from training set) to achieve smaller difference between training and validation accuracy.
Following is the visualization of how the accuracy of the model was improving over the epochs of training.



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 first and second images may be difficult to be classified correctly because of inconsistent background of displayed traffic signs. Third image should become definitely a challenge for the classifier, since the parts of traffic sign have different saturation and very inconsistent background in general. Forth image is also going to become a tough task for my classification model, since the original shape of traffic sign is distorted. Fifth image may bring some difficulty during classification because of very similar hue of the background and traffic 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.

Here are the results of the prediction:

File name	Real Sign type	Predicted Sign Type
27_pedestrians.jpg	Pedestrians	Right-of-way at the next intersection
32_end_of_all_limits.jpg	End of all speed and passing limits	End of all speed and passing limits
01_speed_limit_30.jpg	Speed limit (30km/h)	Speed limit (30km/h)
13_yeld.jpg	Yield	Yield
14_stop.jpg	Stop	Stop

The model was able to correctly predict 4 of the 5 traffic signs, which gives an accuracy of 80%. This result is smaller than accuracy achieved on the initial test set, but can naturally be explained by very small size of web downloaded test dataset. Increasing the dataset size should approximate the classification accuracy to previously received 93% result.

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.

The code for making predictions on my final model is located in 14th cell of the Ipython notebook. The code for visualizing the results of predictions (table above) islocated in the 16th cell of the notebook. Also here are the top 5 probabilities predicted by my trained model for each of 5 images from the web (code that renders below report is located in 17th cell of the notebook)

```
Top 5 predictions for image in file '27_pedestrians.jpg':
        Right-of-way at the next intersection: 87.26%
        Pedestrians: 12.74%
        General caution: 0.00%
        Double curve: 0.00%
        Beware of ice/snow: 0.00%

Top 5 predictions for image in file '32_end_of_all_limits.jpg':
        End of all speed and passing limits: 66.35%
        Speed limit (60km/h): 23.45%
        Priority road: 8.65%
        Ahead only: 0.74%
        Children crossing: 0.33%

Top 5 predictions for image in file '01_speed_limit_30.jpg':
        Speed limit (30km/h): 100.00%
        Speed limit (50km/h): 0.00%
        Speed limit (60km/h): 0.00%
        Keep right: 0.00%
        Roundabout mandatory: 0.00%

Top 5 predictions for image in file '13_yeld.jpg':
        Yield: 100.00%
        No passing for vehicles over 3.5 metric tons: 0.00%
        Ahead only: 0.00%
        No vehicles: 0.00%
        Speed limit (80km/h): 0.00%

Top 5 predictions for image in file '14_stop.jpg':
        Stop: 100.00%
        Go straight or right: 0.00%
        Speed limit (60km/h): 0.00%
        Keep right: 0.00%
        Yield: 0.00%
```

As expected, model was not able to properly classify the distorted ‘pedestrians‘ traffic sign, although the prediction probability for correct classification of this image was a second from the top (12.74%). Surprisingly classifier did extremely well (100% probability) with the recognition of ‘STOP’ traffic sign, which had a big saturation drop in the middle of the sign image. Same for ‘Yield‘ and ‘Speed limit 30km/h’ images – despite the inconsistent background (and slight distortion in case of ‘Yield‘ sign), model was able to classify these traffic signs correctly with 100% probability. And as it was expected model had some ‘doubts’ while classifying the ‘End of all limits’ traffic sign image – correct prediction got only 66.35%, and rest 33.65% of total probability model distributed among 4 other types of traffic signs, which shows that model was pretty uncertain about this sign in general.