Traffic Sign Recognition

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](https://review.udacity.com/#!/rubrics/481/view) individually and describe how I addressed each point in my implementation.

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 numpy library to calculate summary statistics of the traffic

signs data set:

```
Number of training examples = 139196
Number of validating examples = 4410
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43
```

2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. I'm selecting random image from dataset and show it by matplotlib.pyplot

index = random.randint(0, len(X_train))

image = X_train[index].squeeze()

2. 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.

I’ve normalized the data set by code:

```
np.divide(np.subtract(np.cast[float](X_train_norm), 128), 128)
```

As the result I have float variable with -1 to 1 values. I didn’t add any more preprocessing, as I had good results with that. I’ve also tried with grayscale, but it has worse results for my network.

I decided to generate additional data because sometimes “It isn’t who has the better code, but who has more data to train”. I’ve extended the dataset by new images that are rotated from base set.

```
for j in range(n_ran) :
  for i in range(len(X_new)):
    M=cv2.getRotationMatrix2D((16, 16), 5*random.randint(-3, 3), 1)
    img[0, :, :, 0] = cv2.warpAffine(X_train[i, :, :, 0], M, (32, 32))
    img[0, :, :, 1] = cv2.warpAffine(X_train[i, :, :, 1], M, (32, 32))
    img[0, :, :, 2] = cv2.warpAffine(X_train[i, :, :, 2], M, (32, 32))
    X_new[i] = np.asarray( img[0], dtype="uint8" )
```

As the result I have 139196 training examples. That help to increase accuracy without overfitting.

The difference between the original data set and the augmented data set is the following It can be rotated by -15 to 15 deg.

I’ve tired also with translate, but it hasn’t good results.

3. 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:

4. 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 ....

Layer 1 -> truncated normal from 32x32x3 to 28x28x24, conv2d with weights and bias, next RELU and max pool to 14x14x24

Layer 2 -> truncated normal from 14x14x24 to 10x10x64, conv2d with weights and bias, next RELU and max pool to 5x5x64

Layer 3 -> flattens the output to 1600 values vector fully connected to 480 values, then relu

Layer 4 -> fully connected layer 480 to 336, with relu activation

Layer 5 -> changes the 336 vector to 43 – the number of signs classes.

5. 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:

* validation set accuracy of 0.956

* test set accuracy of 0.950

I've choose the LeNet architecture from the laboratory.

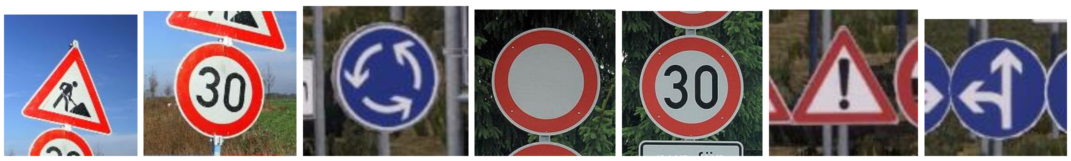
It should be a good choice, as it also is working with recognition of images

I've achieved tes Accuracy of 0.95, what is bigger than 0.93 – base on that I can say that it fulfil the requirements

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 (actually I've choose 7) German traffic signs that I found on the web:



The first image might be difficult to classify because It is rotated speed limit

The second image might be difficult to classify as it is a round and blue, that wasn't much represented in training set

The third – "No vehicles" Is hard, as there is a lot of similar signs – this is the base of all restriction signs.

The fourth sign is a speed limit of 30 again, but with different background

The fifth is the general causion sifn, similar to traffic lights,

The drive straight or left sign has other blue signs in the background, that makes the sign with worse visability

The seventh sign is a road construction rotated to the right.

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:

```
[25  1 40 15  1 18 37]
```

That means:

Road work -> Road work

Speed limit (30km/h) -> Speed limit (30km/h)

Roundabout mandatory -> Roundabout mandatory

No vechiles -> No vechiles

Speed limit (30km/h) -> Speed limit (30km/h)

General caution -> General caution

Go straight or left -> Go straight or left

The model was able to correctly guess 7 of the 7 traffic signs, which gives an accuracy of 100%. But the test set of 7 examples is rather small. The test set of 12630 examples gave 95%.

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.

Every time the CNN is pretty sure about detection with probability of 1. That can be interpreted as the easy test set.

```
TopKV2(values=array([[  1.00000000e+00,   1.26739472e-26,   5.23413688e-28,
          9.72767895e-29,   9.32159403e-31],
       [  1.00000000e+00,   1.19940026e-19,   5.50809032e-24,
          4.72120957e-26,   1.10047065e-29],
       [  1.00000000e+00,   4.33751117e-22,   1.10661960e-24,
          1.28249092e-26,   9.22443695e-28],
       [  1.00000000e+00,   1.69402589e-10,   2.87003012e-20,
          5.94620803e-21,   7.25897436e-23],
       [  1.00000000e+00,   1.54833683e-19,   4.47641077e-22,
          4.58079789e-23,   1.50835613e-26],
       [  1.00000000e+00,   1.78081266e-27,   1.21680028e-29,
          4.08261714e-30,   1.66061316e-30],
       [  1.00000000e+00,   2.42061464e-23,   1.36234006e-23,
          6.39298753e-30,   7.75101932e-31]], dtype=float32), indices=array([[25, 3
0, 22,  5, 24],
       [ 1,  0,  4,  2,  8],
       [40, 37, 33, 39, 41],
       [15,  2,  1,  8,  9],
       [ 1,  4,  0,  2, 18],
       [18,  4, 27, 26,  1],
       [37, 35, 26, 34, 39]]))
```

(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?

Unfortunately I had a problem with access to the variables of the CNN.