Traffic Sign Recognition

Build a Traffic Sign Recognition Project

The goals/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

The project implementation can be found in Traffic sign classifier.ipynb file. It consists of five steps.

Step 0: Load The Data

In this steps, the provided data is loaded using the `pickle` library. The images have labels to recognize what they represent. The labels are numbers, but there is a .csv file containing the mapping between the labels and a text name of the image to make it more human-friendly.

Step 1 : Dataset Summary & Exploration

Here the data set is explored. First, show some general numbers about it:

- Number of training examples = 34799
- Number of testing examples = 12630
- Number of validation examples = 6960
- Image data shape = (32, 32, 3)
- Number of classes = 43

We have 43 different traffic signs. Some random images are shown:

Step 2: Design and Test a Model architecture

Pre-processing

Neural networks work better if the input(feature) distributions have mean zero. A suggested way to have that normalization was to operate on each pixel by applying: (pixel - 128)/128. There are a lot of different preprocessing we could do to improve the image qualities, but I decided to go just for gray scaling the images.
The final pre-processing used consist of two steps:

- Converting the images to gray scale.
- Transforming the pixel values to the range [-1, 1] by subtracting 128 and then divide it by 128.

Model architecture

My final model consisted of the following layers:

Input - The LeNet architecture accepts a 32x32xC image as input, where C is the number of color channels. Since images are grayscale, C is 1 in this case.

- Layer 1 - Convolutional. The output shape should be 28x28x6.
- Activation - Relu.
- Pooling - The output shape should be 14x14x6.
- Layer 2 - Convolutional. The output shape should be 10x10x16.
- Activation - Relu.
- Pooling - The output shape should be 5x5x16.
- Flatten - Flatten the output shape of the final pooling layer such that it's 1D instead of 3D. The easiest way to do is by using `tf.contrib.layers.flatten`.
- Layer 3- Fully Connected. This should have 120 outputs.
- Activation - Relu.
- Droput.
- Layer 4- Fully Connected. This should have 84 outputs.
- Activation - Relu.
- Layer 5- Fully Connected (Logits). This should have 10 outputs.

Output - Return the result of the 2nd fully connected layer.

This architecture was originally inspired by the famous LeNet, but then modified during trial-and-error. The introduction of dropout helps to stabilize the training process.

Train, Validate and Test the Model

I started training with 10 epochs with batch size of 128 and used Adam optimizer. The final settings used were:

1. batch size: 100

2. epochs: 50

3. learning rate: 0.0009

4. mu: 0

5. sigma: 0.1

6. keep probability: 0.5

I started with pre-defined architectures (LeNet and the Sermanet/LeCun model) and almost all of the tweaking from there was a process of trial and error. My guesses were educated through the Tensorflow and LeNet labs. I kept a log of my experiments, which helped finding hyperparameters to get a good level of validation accuracy.

The final model has:

- **Validation Accuracy**: 99.0 %
- **Test Accuracy**: 94.1 %

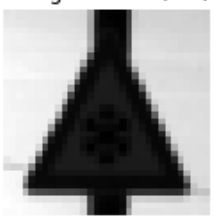
Step 3: Test a Model on New images

In this step, five new images found on the Web are classified. First, the images are loaded and presented:

I downloaded them manually from Google, this is why they were all different sizes. When using them for predictions, I used CV2 to resize them to 32x32 pixels. All the signs photos look like they should not be a challenge for the model to classify. None of them were taken in low light conditions. Neither do they have significant distortions.

Four out of the five images were classified correctly. Accuracy of predictions for new images was 0.80. This is lower than the accuracy for the test set of 0.941.

Here are the softmax probabilities for them are:

input	top guess: 11 (100%)	2nd guess: 30 (0%)	3rd guess: 21 (0%)
			
input	top guess: 1 (100%)	2nd guess: 2 (0%)	3rd guess: 5 (0%)
			
input	top guess: 26 (84%)	2nd guess: 18 (7%)	3rd guess: 25 (6%)
			
input	top guess: 12 (100%)	2nd guess: 7 (0%)	3rd guess: 40 (0%)
			
input	top guess: 25 (100%)	2nd guess: 30 (0%)	3rd guess: 21 (0%)